Operational Optimisation of Water Distribution Networks

Manuel López-Ibáñez

November 2009

A thesis submitted in partial fulfilment of the requirements of
Edinburgh Napier University,
for the award of Doctor of Philosophy

Abstract

Water distribution networks are a fundamental part of any modern city and their daily operations constitute a significant expenditure in terms of energy and maintenance costs. Careful scheduling of pump operations may lead to significant energy savings and prevent wear and tear. By means of computer simulation, an optimal schedule of pumps can be found by an optimisation algorithm. The subject of this thesis is the study of pump scheduling as an optimisation problem.

New representations of pump schedules are investigated for restricting the number of potential schedules. Recombination and mutation operators are proposed, in order to use the new representations in evolutionary algorithms. These new representations are empirically compared to traditional representations using different network instances, one of them being a large and complex network from UK. By means of the new representations, the evolutionary algorithm developed during this thesis finds new best-known solutions for both networks.

Pump scheduling as the multi-objective problem of minimising energy and maintenance costs in terms of Pareto optimality is also investigated in this thesis. Two alternative surrogate measures of maintenance cost are considered: the minimisation of the number of pump switches and the maximisation of the shortest idle time. A single run of the multi-objective evolutionary algorithm obtains pump schedules with lower electrical cost and lower number of pump switches than those found in the literature. Alternatively, schedules with very long idle times may be found with slightly higher electrical cost.

Finally, ant colony optimisation is also adapted to the pump scheduling problem. Both Ant System and Max-Min Ant System are tested. Max-Min Ant System, in particular, outperforms all other algorithms in the large real-world network instance and obtains competitive results in the smallest test network. Computation time is further reduced by parallel simulation of pump schedules.

Contents

Chapter 1

Introduction

A consequence of rising energy prices is to ration the available energy for various tasks. In water distribution systems, water often needs to be pumped to a higher elevation with adequate pressure. For example, hydraulic pumps transport water from a treatment plant into an elevated storage tank. From an elevated tank, water falls by gravity to reach with adequate pressure nodes in the water network where water is consumed. Hydraulic pumps consume most of the energy required to operate a water distribution system. Therefore, optimising pump operations may lead to significant reductions in energy expenditure. During the last decade many studies have proposed various automatic systems for the optimal scheduling of pump operations with the aim of saving energy and reducing operating and maintenance costs. The present day energy requirements are higher: larger and more complex water networks need to be handled. Moreover, the problem formulation involves various constraints to maintain service levels besides minimisation of operational and maintenance costs. Due to high variability and uncertainty involved with the prediction of demands, better results are expected to be obtained by using real-time optimisation. For real-time optimisation, the formulated model (including simulation model) and the chosen optimiser must be able to produce results in real-time. Therefore, despite the advances in computing and hydraulics, the problem seems as challenging today as it was a decade ago.

As an *optimisation problem*, pump scheduling is difficult to solve because of its large search space, high computing requirements and the nonlinear and discontinuous nature of real-world networks. As an *engineering problem*, automatic pump scheduling is an arduous task due to the diversity and complexity of real-world water distribution systems. An engineer developing a system to automatically optimise pump operations in a real-world network would need to address a multitude of problems: collect data about the physical network; build and calibrate according to this data a hydraulic model of the network and a demand forecast model; and define appropriate performance criteria and constraints in order to offer an adequate, reliable and competent service to customers. All these tasks are prerequisites to develop the optimisation system itself and they may well determine the design of the optimisation system. As a *research topic*, automatic pump scheduling is challenging. Because of its strongly practical nature, it is difficult to provide a general and consistent formulation of the problem. Moreover, each instance of the pump scheduling problem is motivated by a different application with unique peculiarities and requirements. Therefore, it is not surprising that each instance is often solved by a mostly specific approach that may not be suitable for a different instance of the problem. The end result is that most research on pump scheduling is never contrasted on multiple network instances, or compared against alternative techniques.

The goal of this thesis is to address the problem from a more general perspective in order to develop techniques that may be applied to diverse instances of the problem. Our goal is, as well, to assess the robustness of these techniques when applied to different instances of the problem. Finally, another further goal is to determine the actual effectiveness of a particular technique in comparison to alternative techniques by means of sound and thorough experimental analysis. The aforementioned complexity of the pump scheduling problem can be overcome by dividing it into subproblems that may be independently addressed. Therefore, in this thesis we will focus primarily on the optimisation algorithm and on the characteristics of the problem that affect it.

An optimisation algorithm generates potential solutions that are, hopefully, optimal (or a good approximation to the optimal) with respect to the objectives of the problem and feasible with respect to its constraints. Hence, we will discuss the formulation of the objectives and constraints of the pump scheduling problem at length. Another relevant aspect is the representation of the solutions for the problem, i.e., the precise definition, meaning and domain of the decision variables for which the algorithm tries to find appropriate values. In the pump scheduling problem, a solution is a schedule of the pumps and there exist several alternatives for defining a pump schedule. Each of them has its own advantages and disadvantages. A particular representation strongly determines how an optimisation algorithm handles a problem. Therefore, new representations may provide new and innovative methods to address the problem. Finally, the optimisation algorithm itself is the main subject of this thesis. Although inventing a radically new approach might be tempting, there exist already several well-known, widely tested and successful optimisation techniques. Evolutionary Algorithms and Ant Colony Optimisation are two general-purpose optimisation techniques that have been applied to numerous problems. We aim to provide a comprehensive experimental analysis of both techniques in comparison with results provided in the literature. We hope to produce state-of-the-art results for the network instances considered in our study.

1.1 Aims of the Thesis

The main aims of this thesis are to:

1. Study the pump scheduling problem formulation in order to develop a general definition of its objectives and constraints.
2. Review the most important representations of pump schedules and develop alternatives that may enhance the optimisation algorithm and provide additional benefits.
3. Investigate the application of various optimisation techniques.
4. Provide a sound and exhaustive experimental analysis of the above techniques applied to different network instances in order to identify best settings, best representations and best techniques.
5. Based on the previous findings, to produce state-of-the-art results for publicly available network instances.

1.2 Outline of the Thesis

This thesis is comprised of 7 chapters:

Chapter 1 corresponds to this introduction.

Chapter 2 describes the pump scheduling problem in very general terms. Objectives and the most popular constraints are discussed. In addition, the recent literature on the problem is reviewed. We present a general schema of the systems proposed in this thesis. The hydraulic simulator utilised in this work, EPANET, is described and we enumerate the improvements incorporated to it. Finally, this chapter introduces the two test networks that will be used in the experiments throughout this thesis.

Chapter 3 discusses the representation of pump schedules. First, the two traditional representations, namely, the binary representation and level-controlled triggers, are reviewed. Then, two new representations based on the concept of time-controlled triggers are proposed. The advantages of the new representations are explored in detail.

Chapter 4 explores the application of Evolutionary Algorithms to the pump scheduling problem. First, a simple evolutionary algorithm (SEA) is defined. Second, evolutionary operators (recombination and mutation) are reviewed for the traditional representations. New recombination and mutation operators are proposed for the new representations based on time-controlled triggers. Next, a thorough experimental analysis by means of statistical tools investigates the effect of the parameters of SEA and their interactions. The experimental analysis identifies the best *configurations* of SEA for each representation and each test network. Then, these configurations are compared with each other and with results available in the literature. Further experiments analyse the effect of the constraint on the number of pump switches and the effect of the length of time intervals.

Chapter 5 investigates the formulation of pump scheduling as a multi-objective problem in terms of Pareto optimality. Two different multi-objective variants are proposed in this chapter. First, the minimisation of energy cost and number of pump switches is considered. Here, the number of pump switches is a measure for maintenance costs. Second, we introduce the shortest idle time in a pump schedule as an alternative surrogate measure of maintenance costs. Hence, we propose a multi-objective formulation that minimises energy cost and maximises the shortest idle time. Experiments are carried out by means of a multi-objective evolutionary algorithm (SPEA2) and analysed using state-of-the-art methods. The results are compared with the single-objective SEA proposed in the previous chapter and with the results from the literature.

Chapter 6 proposes the application of Ant Colony Optimisation (ACO) to the pump scheduling problem. The pump scheduling problem is adapted to the ACO framework by means of the time-controlled triggers representation. First, the approach is empirically tested by using the Ant System algorithm. Next, a more advanced ACO algorithm, Max-Min Ant System, is utilised to solve the pump scheduling problem. Finally, ACO is empirically compared with both SEA and the results available in the literature. In addition, the computation effort required by ACO in the pump scheduling problem is investigated. As a result of our findings, we propose a parallel evaluation of pump schedules by using a thread-safe variant of EPANET in order to significantly reduce the computation time.

Chapter 7 summarises the main conclusions and contributions of this work, enumerates publications arising from this thesis and offers ideas to further extend our investigation.

In addition to the main body, several appendixes provide expanded information that is of interest for future reference:

Appendix A lists the results of SEA obtained in the experiments of Chapter 4.

Appendix B lists, in a similar way, the results of ACO obtained in the experiments of Chapter 6.

Appendix C describes the format of the experimental data, which is publicly available for further analysis and comparison. All our algorithms utilise a common output format that is both easy to understand by humans and easy to parse by computer programs. We encourage researchers to utilise a similar output format to facilitate the comparison of future optimisation algorithms for the pump scheduling problem.

Appendix D enumerates in detail our modifications to the EPANET Toolkit library. Our improvements include new functions and features, computation time optimisations and bug fixes. The source code of our implementation is publicly available.

Appendix E introduces a new thread-safe variant of EPANET Toolkit to be used by parallel applications. First, the limitations of the original EPANET for parallel computing are discussed. Next, the implementation of a thread-safe variant is described. Finally, a random search algorithm that evaluates pump schedules in parallel is linked with the thread-safe library for the purpose of testing the benefits of multiple CPUs. In Chapter 6, this thread-safe version of EPANET is combined with a parallel ACO algorithm in order to reduce the computation time of ACO.

Appendix F reproduces a complete EPANET input file corresponding to the Vanzyl network as an example of instance of the pump scheduling problem. Although some versions of this input file are available in the literature, the reproduced input file is more complete.

1.3 Contributions of this Thesis

The contributions from this thesis (see Section 7.2 for a detailed list) are based on a general view of pump scheduling as an optimisation problem. Hence, a formal definition of the pump scheduling problem is proposed, which may be easily adapted to many particular network instances. To this end, different alternatives for the most utilised constraints, e.g., constraint on volume deficit and number of pump switches, are examined.

The main contributions of this thesis are new proposals on how to approach the problem using existing techniques. These techniques are evolutionary algorithms (EA), ant colony optimisation (ACO) and multi-objective evolutionary algorithms (MOEA). In particular, ACO has never been applied before to the pump scheduling problem. The results presented in this thesis suggest that ACO is a competitive approach. Two multi-objective formulations of pump scheduling are tested in this thesis. The first one is the minimisation of pump switches in addition to the minimisation of electrical cost. This formulation has been considered before in the literature. However, in this thesis, an experimental analysis with various representations and evolutionary operators is carried out by means of up-to-date analysis methods. The second multi-objective formulation is proposed for the first time by this thesis. In this formulation, the second objective is maximisation of the shortest idle time as surrogate measure of maintenance costs.

Another significant contribution is two new representations of pump schedules and their corresponding evolutionary operators. These representations are based on the concept of time-controlled triggers, and they allow to limit the search to those solutions that satisfy the pump switches constraint. To the best of our knowledge, this thesis provides the most comprehensive experimental analysis of different representations and their associated evolutionary operators available in the literature. The conclusions of the analysis suggest that time-controlled triggers achieve superior results in comparison to traditional representations such as the binary representation or level-controlled triggers.

The experimental analysis performed in this thesis also explores other characteristics of the pump scheduling problem. For example, a higher limit on the number of pump switches provides more flexibility when scheduling pump operations, and intuitively, this flexibility should lead to reduced electrical cost. However, the experimental results presented in this thesis indicate that schedules with low electrical cost often have a moderate number of pump switches. In fact, restricting the search to those schedules with a low number of pump switches often leads to lower electrical cost. Another conclusion from the experimental analysis is that reducing the minimum time interval between pump switches allows to find lower cost schedules. However, the search space is also increased, and the algorithm requires more computation time to find those lower cost schedules. On the other hand, a setting of the minimum time interval larger than one hour produces noticeable worse results.

A further contribution is a new thread-safe variant of EPANET. This new version of EPANET enables the parallel evaluation of different pump schedules for the same network instance. By means of this thread-safe EPANET, ant colony optimisation may take advantage of parallel processors to significantly reduce the total computation time required by a single run, without altering the final solution found by the algorithm.

Chapter 2

The Pump Scheduling Problem

In any modern city, when we turn on a tap at home we expect potable water to come out. Moreover, we also expect an adequate flow of water at sufficient pressure. In order to satisfy our expectations, sufficient volume of water must be transported from a source of potable water, e.g., a treatment plant, to demand points (consumers) through a network of pipes. Although water can be transported by gravity, more often it must be pumped in order to reach higher elevations with sufficient pressure. However, pumps cannot be activated whenever we turn on a tap and people do not consume water uniformly throughout the day: within a single day there are periods of high and low consumption. Therefore, water must be stored in tanks at a higher elevation, so that it can be supplied whenever there is a higher demand. These elements define a life-cycle of the water within the networks of our city: water is pumped from the source of potable water, filling storage tanks, and water is consumed at some nodes of the network, emptying the tanks.

Reducing the energy consumption of water distribution networks has never had more significance than in the present day. The greatest energy savings can be obtained by careful scheduling of the operation of pumps. Moreover, the cost of operating pumps in a water distribution network represents a significant fraction of the total expenditure incurred in the operational management of water distribution networks worldwide. Pumps consume large amounts of electrical energy for pumping water from source to storage tanks and/or demand nodes. In addition, they eventually need to be repaired and replaced, resulting in maintenance costs. Therefore, the goal of the pump scheduling problem is to minimise the total operational cost, which includes pumping cost and pump maintenance cost, while guaranteeing a competent network service. In most cases, a competent network service is equivalent to supplying water to consumers at adequate pressures and achieving full recovery of tank levels by the end of operating period.

2.1 The Cost of Pumping Water

The main goal in the pump scheduling problem is to minimise the cost of supplying water, while keeping within physical and operational constraints (Ormsbee & Lansey, 1994), by means of scheduling daily pump operations. There are two types of costs associated with the operation of pumps: energy costs and maintenance costs. The energy cost may be composed of an *energy consumption charge* (£/kWh), i.e., the cost of electric energy consumed during a time interval, and a *demand charge* (£/kW), i.e., the cost associated with the maximum amount of power consumed within a billing period (Walski *et al.*, 2003). Maintenance costs are mainly associated with the wear and tear of pumps, which will result in future repair or even replacement of damaged pumps.

Let the operations of N^{p} pumps be scheduled over a time period, e.g., 24 hours. This *scheduling period* is divided into a number of time intervals (N^{T}). A certain pump schedule S represents which pumps operate during which time interval. The total cost of energy is calculated as:

$$\begin{aligned} C_{\mathrm{E}}(S) &= \sum_{n=1}^{N^{\mathrm{p}}} (\text{demand charge} + \text{consumption charge}) \\ &= \sum_{n=1}^{N^{\mathrm{p}}} \left[P_{\mathrm{d}}\, E_{\mathrm{d}}(n) + \sum_{t=0}^{N^{\mathrm{T}}} P_{\mathrm{c}}(t)\, E_{\mathrm{c}}(n,t)\, S(n,t) \right] \end{aligned} \tag{2.1}$$

where N^{p} = number of pumps
N^{T} = number of time intervals in a pump scheduling period
$S(n,t)$ = duration for which pump n is operating during time interval t (hour)
$P_{\mathrm{c}}(t)$ = energy consumption tariff during time interval t (£/kWh)
$E_{\mathrm{c}}(n,t)$ = energy consumption rate of pump n during time interval t (kWh/h)
P_{d} = demand charge (£/kW)
$E_{\mathrm{d}}(n)$ = maximum electric power consumption of pump n (kW)

The energy consumption rate of a pump depends on the flow through the pump, head supplied by the pump, and the efficiency at which it operates, during a particular time interval (Walski *et al.*, 2003):

$$E_{\mathrm{c}}(n,t) = \frac{0.01019 \cdot Q(n,t) \cdot h(n,t)}{e(n,t)} \tag{2.2}$$

where $Q(n,t) =$ flow rate through pump n during time interval t (l/s)
$h(n,t) =$ total dynamic head (TDH) supplied by pump n during time interval t (m)
$e(n,t) =$ overall (wire-to-water) efficiency of pump n during time interval t (%)

The electrical power required to drive the pump is calculated as:

$$E_{\mathrm{d}}(n) = \frac{0.0098 \cdot Q^{\max}(n) \cdot h(n)}{e(n)} \tag{2.3}$$

where $Q^{\max}(n) =$ peak flow rate through pump n (l/s)
$h(n) =$ total dynamic head (TDH) supplied by pump n (m)
$e(n) =$ overall (wire-to-water) efficiency of pump n (%)

The demand charge is normally applied to the maximum power demand (kW) over a billing period (typically a month) longer than the scheduling period (typically one day). Nonetheless, maximum demand policies can be modelled within a scheduling period by calculating the corresponding penalty cost to the maximum power use over the scheduling period (McCormick & Powell, 2003a).

On the other hand, maintenance costs are difficult to quantify. Instead, they are estimated using a surrogate measure, such as the number of pump switches. A *pump switch* is defined as the action of turning on a pump that was not operating during the previous time interval (Lansey & Awumah, 1994). Frequent switching causes wear and tear of pumps, which, in turn, increases maintenance costs. Thus, the general practice is to minimise the number of pump switches resulting in lower maintenance costs. Most of the approaches in the literature consider energy costs as the most important objective and the number of pump switches is added as a constraint of the problem.

2.2 Constraints of the Pump Scheduling Problem

In order to be useful in practice, feasible schedules must satisfy certain constraints. These constraints include *hydraulic constraints*, also called implicit system constraints, which define the hydraulic equilibrium state of the system, e.g., Conservation of Mass at each node and Conservation of Energy around each loop in the network. On the other hand, *implicit bound constraints* represent system performance criteria. They include constraints on junction pressures, pipe flow rates or velocities, and tank water levels. Implicit bound constraints may also include constraints on pump operation switches. Frequent switching a pump on and off results in maintenance costs due to increasing wear on the pump, thus constraining the number of pump switches limits future maintenance costs.

Constraints on tanks water levels typically include minimum and maximum limits on tank levels, and balance between supply and demand from tanks. Minimum and maximum tank limits may be explicit constraints of the problem or can be implicitly enforced by a hydraulic simulator. Balance between water supplied and consumed from tanks is achieved by ensuring that tanks recover their levels by the end of scheduling period. Balancing supply and demand also allows to apply a similar pump schedule to the next scheduling period, since consumer demands and network conditions are very similar in consecutive periods.

For a perfect periodicity of the operations, the volume of water in each tank at the end of the scheduling period must be equal to its initial volume (Cohen, 1982). However, this is a very strict constraint. A relaxed formulation allows the final volume of water at each tank to be different from its initial volume and simply requires that the total volume of water pumped into the network is the same as the amount consumed (Goldman & Mays, 2000). Even though this condition suffices to ensure the balance between water supply and demand, it permits that at the end of the scheduling period water may be stored at a lower elevation than at the start. This difference in elevation implies a loss of energy in the system and prevents periodicity. Although the situation may correct itself with time and become periodic, this is not guaranteed and subsequent scheduling periods will require more pumping, increasing costs. An alternative formulation that does not produce such loss of energy imposes a constraint stipulating that the final volume in a tank should not be lower than its initial volume (Mäckle, Savic & Walters, 1995; van Zyl, Savic & Walters, 2004).

Following this latter definition, we formulate the constraint on the balance between supply and demand as follows. *Tank volume deficit* (ΔV_k) is defined as the difference in percentage between the initial volume ($V_{k,S}$) and the final volume ($V_{k,E}$) of water in a tank k (2.4a). A negative volume deficit represents a surplus of water in the tank. However, it is not assumed that this surplus compensates the loss of water in a different tank. The *volume deficit tolerance* (ΔV^{tol}) is a parameter of the problem formulation that defines the volume deficit that is allowed (2.4b). Only values that are higher than ΔV^{tol} are accumulated to calculate the *total volume deficit* (ΔV) of a particular schedule, which must be zero in a feasible solution (2.4c).

$$\Delta V_k = 100 \cdot \frac{V_{k,S} - V_{k,E}}{V_{k,S}} \tag{2.4a}$$

$$\Delta V'_k = \begin{cases} \Delta V_k & \text{if } \Delta V_k > \Delta V^{\text{tol}}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4b}$$

$$\Delta V = \sum_{k=1}^{N^{\text{t}}} \Delta V'_k \; = \; 0 \tag{2.4c}$$

where N^{t} is the number of tanks in the network.

The parameter ΔV^{tol} allows to model different problem instances. In certain situations, operators may decide to permit a certain amount of deficit to be able to obtain lower costs. A positive value of ΔV^{tol} would mean that feasible schedules may generate a volume deficit of up to ΔV^{tol} percentage of the total volume of each storage tank. It is not difficult to imagine a situation where the requirement is that final tanks levels must be higher than initial levels by at least a specified volume. This formulation can be modelled by defining a negative volume deficit tolerance ($\Delta V^{\text{tol}} < 0$). A straightforward extension may contemplate a different volume tolerance for each storage tank ΔV_k^{tol}, allowing maximum flexibility. Moreover, a multi-objective formulation may consider ΔV^{tol} not as a constraint, but as an objective value that must be minimised. Normally $\Delta V^{\text{tol}} = 0$, that is, initial tank levels must be completely recovered at the end of the scheduling period. This is the setting used throughout this thesis.

In addition to balancing supply and demand, a reliable network service must supply water to consumers at adequate pressures. Therefore, the optimisation model must satisfy minimum pressure constraints at demand nodes:

$$H_{k,t} \geq H_k^{\min} \tag{2.5}$$

where $H_{k,t}$ is the head supplied at node k during time period t and $H_k^{\min}$ is the minimum head required at node k. In particular, we accumulate the violations of the above constraints into a single pressure deficit constraint:

$$\begin{aligned} \Delta H_{k,t} &= \begin{cases} \dfrac{H_k^{\min} - H_{k,t}}{H_k^{\min}} & \text{if } H_{k,t} < H_k^{\min}, \\ 0 & \text{otherwise.} \end{cases} \\ \Delta H &= \sum_{t=1}^{N^{\text{T}}} \sum_{k=1}^{N^{\text{d}}} w_k \Delta H_{k,t} \quad = 0 \end{aligned} \tag{2.6}$$

where N^{T} is the number of time periods, N^{d} is the number of demand nodes, and w_k is a weight of the relative importance of pressure violations in node k. This weight allows to model problem instances where some customer nodes (hospitals, critical industries) have a higher importance than others. In this thesis we do not further explore this possibility, and, hence, we set $w_k = 1, \forall k = 1, \dots, N^{\text{d}}$.

Finally, an additional constraint on the number of pump switches of a schedule is often incorporated to the problem formulation in order to reduce maintenance costs. As mentioned earlier, frequently switching a pump on and off causes wear and tear (Lansey & Awumah, 1994). However, our goal is not limiting the total number of pump switches (N^{sw}), but actually limiting the number of switches of each pump (N^{sw}_p), where:

$$N^{\text{sw}} = \sum_{p=1}^{N^{\text{p}}} N^{\text{sw}}_p$$

The difference is that by limiting N^{sw}, a schedule may still contain a pump with a excessive number of switches, whereas this cannot occur if the constraint is applied to N^{sw}_p. Thus, to strictly limit the number of switches per pump to a specified value, the following constraint is considered:

$$N^{\text{sw}}_p = SW \qquad \forall p \in \{1, \dots, N^{\text{p}}\} \tag{2.7}$$

where SW is a constant to be specified and it is the number of switches allowed per pump during the scheduling period. Schedules with a lower number of pump switches are normally also acceptable, and thus, the previous constraint may be relaxed as follows:

$$N^{\text{sw}}_p \leq SW \qquad \forall p \in \{1, \dots, N^{\text{p}}\} \tag{2.8}$$

The above constraints are the most frequently used constraints. Additional constraints, such as limits on source flows or velocity constraints, may be incorporated to the problem formulation depending on particular requisites of a network and the optimisation approach. For example, when a hydraulic simulator is used to evaluate pump schedules, the simulator may issue *warnings* for specific undesirable situations. Such warnings indicate that the schedule is problematic and should not be considered a feasible solution to the problem.[1] Therefore, an additional constraint is added that requires feasible solutions to generate no simulation warnings.

[1] These warnings are discussed in Section 2.4.1 for the particular hydraulic simulator used in this study.

2.3 Review of Previous Approaches

The *optimal control policy* is defined as the schedule of pump operations that will result in the lowest total operating cost for a given set of hydraulic and implicit bound constraints (Ormsbee & Reddy, 1995). In very general terms, a system that automatically controls the operation of pumps in order to obtain an optimal control policy is made up of three main components: a hydraulic network model, a demand forecast model, and an optimal control model. A calibrated *hydraulic network model* is used to calculate the response of the water distribution system to different operational policies. A *demand forecast model* is used to predict water demand during scheduling period from historical data. These demands are incorporated into the hydraulic network model. The optimal control model, which we will refer as the *optimisation algorithm* thereafter, generates optimal control policies by minimising an objective, usually electricity cost, subject to a number of operational constraints.

The automatic scheduling of pump operations in water distribution systems to minimise the costs of supplying water is not a new problem. Ormsbee & Lansey (1994) carried out a review of the state of the art of algorithms for this problem. Following a similar review method, we summarise more recent works in Table 2.1.

We first classify the various approaches according to the type of system addressed by the model: the number of storage tanks (*column 2*) and pumping locations (*column 3*), either individual pumps or pump stations. Additionally, we identify the three main components in an operation control system: the hydraulic network model (*column 4*), the demand forecast model, and the optimisation algorithm (*column 5*). Modern approaches typically use distributed demand model or do not depend on a particular model, so we do not take the demand model into account in our review. Finally, another relevant aspect is whether the approach uses *explicit* or *implicit* representation of pump schedules (*column 6*). All these elements are discussed in more detail in the following sections.

2.3.1 Hydraulic Network Models

Each candidate schedule of the pumps must be evaluated in order to calculate its associated costs and assess its feasibility with respect to the problem constraints. Since testing potential pump schedules on the real system would be impractical, some type of mathematical model of the water distribution system is required. Mass-balance models, regression models, simplified hydraulics and full hydraulic simulation are potential techniques for modelling network hydraulics.

A *mass-balance model* simplifies a single tank system based on some assumptions. First, the volume of flow into the system must be equal to the daily demand plus the difference in water level in the tank. Mass-balance models assume, as well, that some combination of pumps exists that will be able to generate the desired variation of water level in the tank. Finally, some constraints, such as the pressure-head required to achieve flow into the tank and minimum pressure constraints at demand nodes, are either neglected or considered to be satisfied if water level in tanks is above certain elevation.

Regression models are defined by a set of nonlinear equations that is constructed from the response of a given network over a particular range of demands. This approach results in more accuracy than mass-balance models while still being very fast to evaluate. However, regression models are very sensitive to the data used to construct the model. Significant changes in either the network or the demand distribution may lead to erroneous results. In such case, the model must be built again from newly collected data.

In *simplified hydraulics*, network hydraulics are approximated using a highly skeletonised network model, where the effect of several components are related in a single equation. In few particular cases, a simple model of linear hydraulic equations suffices to represent the system hydraulics. Nonetheless, even when this is possible, extensive system analysis is necessary to accurately represent the real system.

Finally, a *full hydraulic simulation* solves hydraulic equations (conservation of mass and conservation of energy) to model the nonlinear dynamics of a water distribution system. Simulation models generally require more data to formulate, and require a significant amount of work to calibrate. On the other hand, simulation models are robust both in terms of system changes and demand variations. While mass-balance or regression models would require significant modifications to account for changes in the system response, a well-calibrated simulation model would be able to provide the hydraulic response of the modified system.

The computational effort to evaluate a single pump schedule is much higher in the case of simulation models than for mass-balance and regression models. Nevertheless, computation power has increased so dramatically that full hydraulic simulations are much faster than 10 years ago. This is reflected in the fact that while up to 1994 most optimisation models made use of mass-balance or regression models and a few considered simplified hydraulics (Ormsbee & Lansey, 1994), hydraulic simulation has become the usual practice in newer works, as can be seen in column 4 of Table 2.1. A clear contribution to this trend is the number of hydraulic simulators that have been made publicly available in the recent years, with the notable example of EPANET (Rossman, 1994, 1999), which is the simulator used in this work.[2]

[2] Section 2.4.1 discusses the characteristics of EPANET, how it has been applied to the present work, and enhancements implemented during the course of our research.

2.3.2 Demand Forecast Models

Not only an accurate hydraulic model of the real system is required in order to develop an optimal pump schedule. We must also know the system demands in advance. Nowadays, most distribution systems have the potential to gather data on system demands. This data is used to statistically estimate typical system demands in order to construct a forecast model. Depending on the accuracy of both the spatial and temporal data available and on the particular hydraulic model, either a lumped, proportional or distributed demand forecast model may be used.

Mass-balance hydraulic models typically use a *lumped model*, which aggregates system demands into a single value. Regression-based hydraulic models normally consider *proportional demand models*, which varies a single demand pattern proportionally to the total demand. Finally, in a full network simulation model, the total system demand is distributed both temporally and spatially among several network demand nodes. This *distributed demand model* is the most accurate.

Although a forecast model is an essential part of the system, the choice of a particular model does not depend directly on the optimisation algorithm but on the data available and the hydraulic model. Most modern approaches use network simulation, which implies a distributed demand model. Therefore, we do not take into account demand forecast in our review and we do not further discuss demand forecast in this work.

2.3.3 Representation of Pump Schedules

Another relevant aspect is the representation of pump schedules within the optimisation algorithm (column 6 of Table 2.1). This representation can be either *explicit*, by directly specifying the status of each pump (Boulos *et al.*, 2001; Goldman & Mays, 2000; Mäckle, Savic & Walters, 1995; McCormick & Powell, 2003b, 2004; Pezeshk & Helweg, 1996; Sakarya & Mays, 2000; Savic, Walters & Schwab, 1997; Wegley, Eusuff & Lansey, 2000); or *implicit*, by defining the operations of pumps in terms of properties of other elements of the network (Atkinson *et al.*, 2000; Dandy & Gibbs, 2003; Kazantzis *et al.*, 2002; van Zyl, Savic & Walters, 2004).

The two most widely used representations are explicit binary representation (Boulos *et al.*, 2001; Goldman & Mays, 2000; Mäckle, Savic & Walters, 1995; Savic, Walters & Schwab, 1997; Sotelo, von Lücken & Barán, 2002), and implicit representation based on tank-level triggers (Atkinson *et al.*, 2000; Dandy & Gibbs, 2003; van Zyl, Savic & Walters, 2004). The binary representation divides the scheduling period in smaller time intervals and encodes a pump schedule in a string of bits, each bit representing the status (on/off) of a pump during a time interval. On the other hand, tank-level triggers change the status of pumps when water on a tank reaches a certain elevation. They are typically used in pairs: the pump is turned off when water level goes above an upper trigger level and it is turned on when water level falls below a lower trigger level. A more detailed discussion about these representations will be carried out in Chapter 3, where new representations are also proposed.

In addition to these two widely used representations, other alternative representations have been proposed. McCormick & Powell (2003b, 2004) suggested a representation that, instead of encoding the status of individual pumps, encodes combinations of (groups of) pumps. In their proposal, these combinations were generated by grouping pumps with a significant hydraulic interaction between them. Decision variables were the proportions of each combination to be used at each time interval.

Sakarya & Mays (2000) tested a representation where for each pump p and each time interval t_i, the decision variable was the length of time $D_{p,i}$ that pump p operates during time interval t_i, where $0 \leq D_{p,i} \leq t_i$. As admitted by the authors, this approach may generate very short operating time during a single time interval and may cause an excessive number of pump switches.

Kazantzis *et al.* (2002) proposed two representations that combined implicit and explicit aspects. The first one, in addition to a single pair of level-controlled triggers for both peak and off-peak tariff periods, incorporated a pump stop time and a pump start time. The pump stop time turns off the pump independently of the actual water level of the tank, so the water in the tank is at minimum level at the end of the peak tariff period. The pump start time turns on the pump ignoring any level-controlled triggers, in order to fill the tank just before the start of the peak tariff period. In the second representation, the pump stop time was replaced by a switch time to revert the upper trigger level to the maximum water level of the tank, enabling the tank to be filled. The pump start time was still used to turn on the pump independently of that actual water level of the tank.

Finally, few other representations were proposed to deal with variable-speed pumps. For example, Wegley, Eusuff & Lansey (2000) extended the binary representation by replacing the binary alphabet $\{0, 1\}$ with the set of speeds of variable frequency drive pumps.

2.3.4 Optimisation Algorithms

As for the optimisation (or control) model, Ormsbee & Lansey (1994) reviewed earlier techniques, such as linear, non-linear, integer, dynamic and mixed programming. Although these techniques are still used, their usefulness is inherently limited when applied to complex water distribution networks. This limitation has led researchers to consider other optimisation techniques, such as Simulated Annealing (Goldman & Mays, 2000), Particle Swarm Optimisation (Wegley, Eusuff & Lansey, 2000) and Evolutionary Algorithms (Atkinson *et al.*, 2000; Boulos *et al.*, 2001; Dandy & Gibbs, 2003; Kazantzis *et al.*, 2002; Mäckle, Savic & Walters, 1995; Simpson *et al.*, 1999).

Mäckle, Savic & Walters (1995) developed an evolutionary algorithm to optimise the daily scheduling of four pumps within a network with a single tank. Schedules were represented by binary strings of 4×24 bits. The cost of each schedule was the sum of the electricity cost and penalties for constraint violations on minimum, maximum, and final tank levels. The fitness of each schedule was calculated as the inverse of the overall cost. Since only the outcome of a single run of the evolutionary algorithm is shown, it is difficult to assess the average performance of the algorithm.

Simpson *et al.* (1999) developed an evolutionary algorithm where the two decision variables are the tank level triggers (represented by a binary string) for all the pumps in a single tank network. Their goal was to minimise the total cost which is calculated from the unit cost per m^3 of water pumped plus penalty costs for violations of constraints: the upper trigger level should be higher than the lower trigger level; tank levels should be within specified ranges; and there must not be more than an average of 6 pump starts per hour. They did not provide any strong reason why the evolutionary algorithm performs better than full enumeration of the 1024 possible solutions.

Goldman & Mays (2000) used Simulated Annealing to optimise operation of pumps using a binary representation for 24 time periods of 1 hour. Their objective was to minimise the total energy cost. Penalties were added to the total cost in case of violation of constraints on nodal pressure head bounds, minimum and maximum free chlorine concentration, minimum tank levels, and if tank levels did not return at the end of the simulation to their starting elevation. They performed an extended period simulation using EPANET for each daily schedule of the pumps. Their test network consisted of three tanks and two pumps. They concluded that the main advantage of Simulated Annealing over nonlinear programming is its flexibility with respect to the problem formulation. A nonlinear programming approach may require modification if there are changes in the network description or water demand, whereas Simulated Annealing is able to handle many different network instances without requiring any modification. This conclusion applies, as well, to other metaheuristics, such as evolutionary algorithms and ant colony optimisation.

Wegley, Eusuff & Lansey (2000) studied Particle Swarm Optimisation for optimising the operation of pumps that can operate at various speeds (variable frequency drive, VFD). Their goal was to minimise the energy cost of pumping while keeping nodal pressure head, demand, and tank water levels within bounds. Violation of these constraints resulted in penalties that were added to the objective function. The decision variables were the VFD speeds at discrete periods. Hydraulic simulation was performed by EPANET. The paper only describes the model and does not perform any experimental study.

Atkinson *et al.* (2000) used an evolutionary algorithm for finding peak and off-peak tank level triggers of level-controlled pumps in a complex water network, obtaining savings in electrical cost of 30% with respect to schedules manually produced by system operators. Since they only considered constraints on minimum, maximum, and final tank levels, these were implemented directly as bounds of the trigger values. However, this does not guarantee that final water level of the tanks will be equal to their initial elevation. In addition, their approach requires manual post-processing of resulting schedules in order to avoid excessive pump switches.

Boulos *et al.* (2001) linked an evolutionary algorithm with a hydraulic network simulator (H_2ONET) to optimise the operation of three parallel pumps in a network with a single tank with respect to electricity cost. Penalties were added to the electricity cost in case of violation of constraints on junction node pressure, pipe velocity, tank water level and pump switches. They used a binary representation of 24 periods of one hour for each pump. The quality of their results cannot be established since only one single result is described and no comparison is performed with any other algorithm.

Kazantzis *et al.* (2002) followed a mixed approach combining trigger levels (implicit controllers) and trigger times (explicit controllers). Their GA optimised four decision variables: (*1*) reduced upper trigger level which keeps a low water level in the tank in order to reduce pumping head; (*2*) pump start time to fill the tank at the end of the off-peak tariff period; (*3*) time of the day when the reduced trigger level is set to the maximum level of the tank so it can be filled; and (*4*) the initial level in reservoir. EPANET was used for running hydraulic simulations to evaluate electrical cost. Constraint violations resulted in a penalty cost added to the electrical cost. However, no procedure was defined to prevent excessive pump switches so the reduced upper trigger level may activate a pump several times per hour. Their test network comprised only one group of pumps and one tank.

Local Search

Because small modifications may turn an infeasible schedule into a feasible one and even improve its objective function value, searching the local neighbourhood of a solution can be beneficial. Therefore, some researchers have studied hybrid methods that improve the efficiency of Evolutionary Algorithms by incorporating local search strategies.

Savic, Walters & Schwab (1997) followed an approach similar to Mäckle, Savic & Walters (1995), as described above. In addition, they incorporated two different local search procedures based on different neighbourhoods: (*1*) schedules differing from the given schedule in exactly the state of a single pump in a single time interval, and (*2*) schedules in which the same pump has a different state in two time periods and the rest of the schedule is identical. Both local searches improved the results obtained by the evolutionary algorithm.

Van Zyl, Savic & Walters (2004) optimised peak and off-peak trigger levels using a hybrid method which combines an evolutionary algorithm and hill climber search strategies. Therefore, each independently controlled group of pumps adds four real decision variables. The objective function was the total energy cost after a 24-hour simulation. Penalty costs were added for the number of pump switches and tank volume deficit. The tank volume penalty cost was calculated as the sum of volume deficit of all tanks at the end of the simulation with respect to the initial tank volume. The pump switches penalty cost was based on the total number of pump switches in a simulation. These penalties were multiplied by penalty unit costs found by trial-and-error. In addition, they considered that reducing the volume deficit was more important than reducing the number of pump switches and, to this end, a third penalty term was added to the objective function. They concluded that the hybrid approach obtained better results than the pure evolutionary algorithm in fewer function evaluations. However, this approach requires to find suitable penalty costs. Moreover, it produces short idle intervals between operating periods, what may damage the network due to sudden pressure fluctuations.

Multiple Objectives

All the above approaches considered a single objective (typically energy cost) and other desirable objectives (such as maintenance cost) were considered constraints and, in most cases, incorporated to the problem as penalties to the objective function. However, the use of penalties poses an important difficulty which is the definition of appropriate penalty costs. High penalty costs will direct the search away from the boundary of the feasible region, where the best solutions may be located. On the contrary, low penalty costs will produce infeasible solutions. Despite these difficulties, little research has been carried out in the application of multi-objective algorithms to the pump scheduling problem (Savic, Walters & Schwab, 1997; Sotelo, von Lücken & Barán, 2002).

Savic, Walters & Schwab (1997) extended the formulation of Mäckle, Savic & Walters (1995) described above to a bi-objective approach: minimisation of energy costs and minimisation of number of pump switches. They used a multi-objective evolutionary algorithm, Pareto optimal ranking from Goldberg (1989), combined with a local search. Constraint violations on tank water levels were incorporated into the electricity cost as penalties and, in addition, infeasible solutions were considered always worse than feasible ones. Their approach was not compared with any other previous approaches and they tested it on a simple network with one tank.

Sotelo, von Lücken & Barán (2002) defined the pump scheduling problem as the minimisation of four objectives: (*1*) electricity consumption cost, (*2*) maximum demand charge, (*3*) number of pump switches, and (*4*) difference between the initial and final levels of the tank. The state of the pumps (on/off) for 24 time periods of one hour was represented using a binary string. They tested several multi-objective evolutionary algorithms (SPEA, NSGA, NSGA2, CNSGA, NPGA and MOGA) incorporating a repair heuristic to them. This heuristic repairs solutions which violate restrictions on the maximum and minimum tank levels. Their network instance consisted on five parallel pumps connected to one tank by a single pipe. They concluded that SPEA showed the best overall performance. However, the metrics used in this work to assess performance have been identified as unreliable by Zitzler *et al.* (2003) and, thus, they may lead to wrong conclusions.

Water Quality

With regard to improving water quality by means of pump scheduling, Dandy & Gibbs (2003); Goldman & Mays (2000); Sakarya & Mays (2000) have studied imposing constraints on chlorine concentration that penalise the objective function if they are violated. Dandy & Gibbs (2003) used an evolutionary algorithm to find the optimal chlorine dosing rate (mg/L) while optimising peak and off-peak trigger levels. The goal of Sakarya & Mays (2000) was to minimise the deviations of the actual concentrations of disinfectant from the desired concentrations. Lastly, Prasad & Walters (2003) studied the minimisation of residence times (the age of water) by means of reconfiguring the water distribution network.

Table 2.1: Summary of optimisation approaches for pump scheduling.

Reference	**Tanks**	**Pumps**	**Hydraulic model**	**Optimisation algorithm**	**Represen-tation**	**Comments**
Mäckle, Savic & Walters (1995)	1	4	Regression model	Evolutionary Algorithm	Explicit	Binary representation. Minimisation of electricity cost plus penalties for constraint violations on minimum, maximum and final tank levels.
Ormsbee & Reddy (1995)	2	2	Hydraulic simulation	Nonlinear Heuristic	Explicit	Custom representation based on rank ordering of pump combinations and percentage of each time interval that each combination is used.
Nitivattananon, Sadowski & Quimpo (1996)	8	10	Mass-balance	Dynamic Programming	Explicit	The system is decomposed in space and time. Heuristic methods are used to discretise and arrange pump discharges.
Pezeshk & Helweg (1996)	none	32	Hydraulic simulation	Adaptive Search Optimisation	Explicit	Pumps are switched on or off depending on a combination of influence coefficients and pipe pressure readings.
Savic, Walters & Schwab (1997)	1	4	Regression model	Hybrid GA / MOEA	Explicit	Binary representation. Same problem instance as Mäckle, Savic & Walters (1995). Multiple objectives: minimisation of both energy cost and number of pump switches.
Andersen & Powell (1999)	15	20	Hydraulic simulation	Nonlinear programming	Explicit	Decision variable is number of active pumps at each pump station. A continuous solution obtained by projected gradient method is translated to a discrete pump schedule.

(continues . . .)

Table 2.1: Summary of optimisation approaches for pump scheduling (continued).

Reference	Tanks	Pumps	Hydraulic model	Optimisation algorithm	Represen-tation	Comments
Simpson *et al.* (1999)	1	1	EPANET	Evolutionary Algorithm	Implicit	Binary string encodes trigger tank levels. Total cost is calculated from the unit cost per kL of water pumped plus penalty costs for constraint violations.
Atkinson *et al.* (2000)	6	7	Hydraulic simulation	Evolutionary Algorithm	Implicit	Applied to the Richmond network. Decision variables are trigger tank levels. Manual post-processing is required to avoid excessive pump switches.
Goldman & Mays (2000)	3	2	EPANET	Simulated Annealing	Explicit	Binary representation. Water quality is considered by incorporating penalties for violation of chlorine concentration constraints.
Sakarya & Mays (2000)	1	1	EPANET	Nonlinear optimisation (GRG2)	Explicit	Decision variable is the length of time that each pump operates during each period. Water quality is taken into account.
Wegley, Eusuff & Lansey (2000)	none	none	EPANET	Particle Swarm Optimisation	Explicit	Decision variables are the variable frequency drive (VFD) pumps speeds at discrete periods.
Boulos *et al.* (2001)	1	3	Hydraulic simulator (H_2ONET)	Evolutionary Algorithm	Explicit	Binary representation. Penalties are added for violation of constraints on junction node pressures, pipe velocity, tank water level and number of pump switches.
Ertin *et al.* (2001)	1	3	Mass-balance	Dynamic Programming	Explicit	The optimisation problem is modelled as a Markov decision problem.

(continues ...)

Table 2.1: Summary of optimisation approaches for pump scheduling (continued).

Reference	Tanks	Pumps	Hydraulic model	Optimisation algorithm	Represen-tation	Comments
Kazantzis *et al.* (2002)	1	1	EPANET	Evolutionary Algorithm	Mixed	4 decision variables including pump start time and upper trigger level. The reduced upper trigger level may activate a pump several times per hour.
Sotelo, von Lücken & Barán (2002)	1	5	Mass-balance	MOEAs: SPEA, NSGA, NSGA2, and MOGA	Explicit	Binary representation. 4 objectives: electricity consumption cost, maximum demand charge, number of pump switches, and difference between the initial and final levels of the tank.
Dandy & Gibbs (2003)	1	1	EPANET	Evolutionary Algorithm	Implicit	Decision variables are upper and lower trigger levels for peak and off-peak tariff periods. Also, chlorine dosing rate is optimised (water quality).
McCormick & Powell (2003b)	10	35	Mass-balance	Progressive Mixed Integer programming	Explicit	Decision variables are proportion of each pump combination to be used in each time slice.
McCormick & Powell (2004)	10	35	EPANET	Simulated Annealing	Explicit	A linearised hydraulic model is iteratively recalibrated using the full model.
van Zyl, Savic & Walters (2004)	2–6	3–7	EPANET	Hybrid GA	Implicit	The same algorithm is applied to two networks: a custom network and the Richmond system.

2.4 Approach Followed in this Thesis

The optimisation approach developed in this thesis follows modern applications that combine an optimisation algorithm with a network simulation model. Most of them use full hydraulic simulation and distributed demand forecast model. Figure 2.1 illustrates this architecture of an optimisation algorithm for the pump scheduling problem. The input to the system is a network instance that includes both a complete description of the water distribution network and demand forecast for at least the scheduling period considered by the optimisation algorithm. An optimisation algorithm generates candidate solutions (pump schedules) which are evaluated by hydraulic simulation to calculate their objective values (e.g., electrical cost) and constraint violations. The optimisation algorithm utilises the evaluated solutions in some manner to generate new candidate solutions.

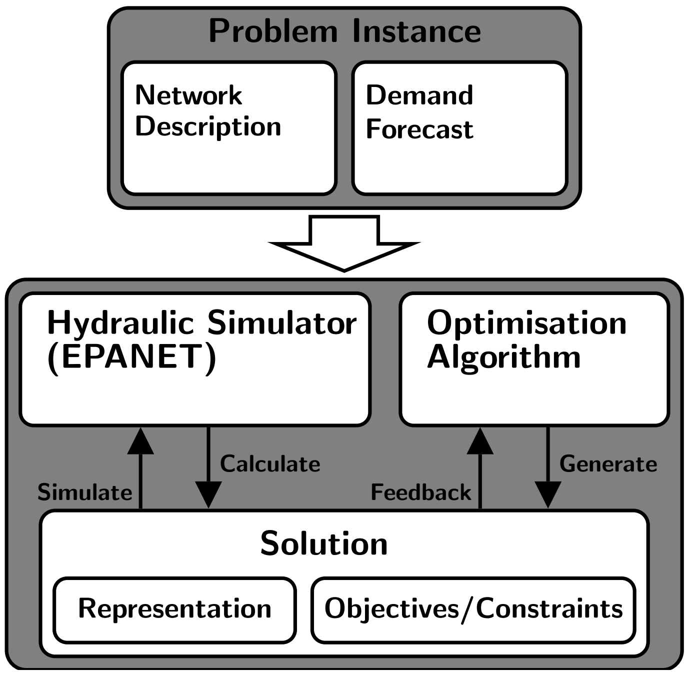


Figure 2.1: Schematic diagram of pump scheduling optimisation system.

In this architecture, the optimisation algorithm is fairly independent from the hydraulic simulator: modifications to either of them do not imply changes to the other. In fact, the same version of a hydraulic simulator is used by all the optimisation algorithms examined in this work. In the next section, we explain in more detail this hydraulic simulator and the improvements we have incorporated to it.

2.4.1 Evaluation of Schedules through Hydraulic Simulation

The hydraulic simulator used throughout this work is EPANET (Rossman, 1994, 1999) version 2.00.10. EPANET reads an input file (see Appendix F for an example) that contains the description of a network including water demand information. The operation of pumps may be triggered by using controls that depend on the time of the day, water levels in tanks or other elements. Alternatively, utilisation patterns may be assigned to pumps to specify their operation schedule. During a simulation, EPANET automatically handles hydraulic constraints and enforces maximum and minimum limits on tank water levels by closing the water flow from/into the tank. However, when water cannot flow out of a tank, pressure is reduced in demand nodes supplied by this tank, what may result in violations of pressure constraints.

The simulator may not be able to evaluate all potential schedules correctly. For example, when a pump cannot deliver sufficient head or flow, it will be forced to shut down or operate beyond the maximum rated flow. From an efficiency point of view, operating a pump at extreme points of the pump curve is undesirable. Shutting down the pump will introduce an extra pump switch that was not present in the original schedule. EPANET will issue a warning for these and other situations.[3] These warnings are not specific to EPANET but they rather indicate an inefficient operation of the pumps or a difficulty to evaluate a schedule. A schedule that cannot be evaluated correctly by the hydraulic simulator cannot be considered feasible, since it is very likely that the simulation has not produced a correct result with respect to objective values and constraint violations. Therefore, in our approach all such warnings were counted and taken into account in the

[3] EPANET issues warnings in six *special* situations (Rossman, 2000):

- A solution to the hydraulic equations was not found (System hydraulically unbalanced).
- A solution to the hydraulic equations was found after the status of all links was held fixed (System may be hydraulically unstable).
- One or more nodes with positive demands were disconnected from all supply sources (System disconnected).
- One or more pumps could not supply sufficient head or flow and they were forced to either shut down or operate beyond the maximum rated flow.
- One or more flow control valves could not deliver the required flow even when fully open.
- Water was not supplied to one or more junctions with positive demand (System had negative pressures).

constraint handling method.

We modified the original version of EPANET in order to fit our purposes better.[4] We added files to simplify building the library in the GNU/Linux operative system. In addition, in our version, the time of the day in seconds when a pump changes its status is saved in a vector, called the pump schedule vector, associated with each pump. The same function that updates the pump schedule vector also records the number of switches per pump. We also added new parameter codes to obtain the number of junctions, pumps and reservoirs in the network; the initial volume, current volume, maximum level and minimum level of a tank; the schedule vector of a pump; the time of the day at which the simulation period starts; and to assign an utilisation pattern to a pump. New functions were added to the toolkit to calculate the total volume of water leaked by emitters, the number of switches performed by a pump, total energy cost (including demand cost) of a simulation, spatial coordinates of a node, the shortest time interval during which a pump was not active during a simulation, and the number of warnings generated by a simulation. Other new functions allow to perform certain operations within the running program without constructing a new input file. In particular, we added functionality to iterate over tanks and pumps without knowing their corresponding name (identifier), node or link index, add a new pattern, add a new control, add a pair of level-controlled triggers and delete all rules.

In addition, we tried to reduce the computation time required by the hydraulic simulator by improving the code and using special compilation flags. We measured an average reduction of 13.7% on an Intel Pentium 4 CPU 2.66 GHz. Further modifications and fine-tuning of compilation flags showed a potential reduction in execution time of approximately 35%. However, these latter modifications affect the rounding of floating-point operations, producing slightly different results. In order to obtain reproducible results, we decided not to use these improvements in our experiments.

Finally, we also fixed a few problems in the code of EPANET that resulted in warnings from the compiler or even segmentation faults during execution. A detailed list of our modifications is reported in Appendix D on page 193.

2.5 Water Distribution Networks

In the rest of this work, optimisation algorithms will be tested on two water distribution networks: Vanzyl test network and Richmond network. The former is a network instance

[4] This modified version is available at `http://sbe.napier.ac.uk/~manuel/epanetlinux`

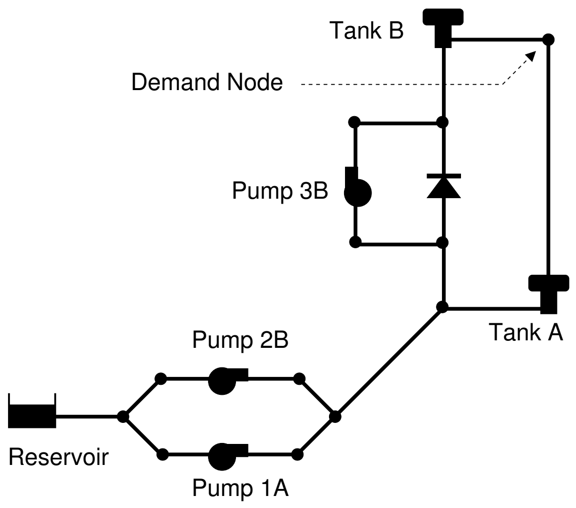


Figure 2.2: Vanzyl test network.

designed solely for benchmarking purposes, while Richmond is a medium-sized real-world network instance. The criteria to choose these two networks were the existence of earlier research, hence our results can be compared to previous optimisation algorithms, and the availability of the complete network descriptions, which allows researchers to replicate and extend our experiments. In the original formulation of these network instances, pump operations are triggered at certain water levels of the tanks, which is the common practice in UK. In this thesis, we modify the network instances in order to test representations of pump schedules that do not rely on level-controlled triggers.

2.5.1 Vanzyl test network

Van Zyl, Savic & Walters (2004) proposed this test network as a small yet complex benchmark network to fine-tune the parameters of a hybrid evolutionary algorithm. The layout of the network is shown in Fig. 2.2. It contains all the main elements of a typical water distribution network: a source of potable water (reservoir), three pumps, two tanks, and a check valve, which prevents water flowing backwards. Pumps 1A and 2B are identical pumps in parallel and, when neither of them is active, pump 3B transfers water from tank A to tank B. In case one or both pumps (1A and 2B) are running, pump 3B boosts the flow to tank B. Tank B has a higher elevation than tank A, and thus, water may flow by gravity from tank B to tank A through the pipes connected to the demand node.

In this instance the demand charge is taken to be zero and the water available at the reservoir is assumed to be infinite. The electricity cost is divided into two periods with a peak electricity tariff period from 7:00 to 24:00 and a off-peak tariff from 0:00 to 7:00. The demand pattern contains two peaks at 7:00 and 18:00. More details about the test instance are provided by van Zyl, Savic & Walters (2004). For convenience, a complete

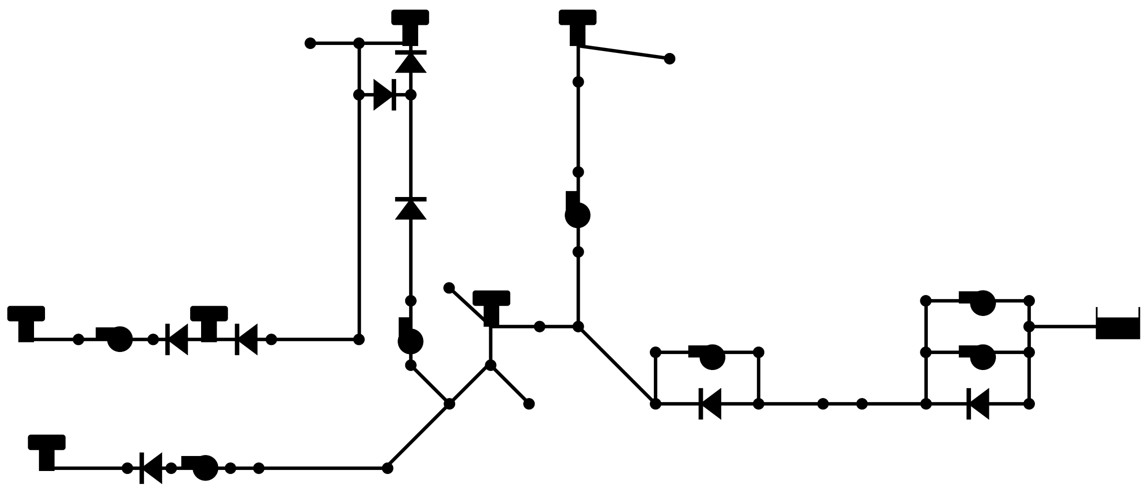


Figure 2.3: Simplified schematic representation of the Richmond network.

EPANET input file describing the network is given in Appendix F on page 204.

2.5.2 Richmond test network

Richmond water distribution system is a real system located in the United Kingdom. The network has 7 pumps, 6 tanks and one reservoir. Figure 2.3 shows a simplified schematic layout that gives an approximate idea of the connections between network elements. The actual network instance used in this work consists of 948 links and 836 nodes. Similarly to the Vanzyl network above, electricity consumption charge is divided into two periods with a peak electricity tariff period from 7:00 to 24:00 and there is no electrical demand charge. A complete description of the Richmond network is available online at `http://www.centres.ex.ac.uk/cws/`. This network was first studied by Atkinson *et al.* (2000), who applied an evolutionary algorithm to reduce the annual operation cost with respect to the original operational policies based on operator experience. Later, van Zyl, Savic & Walters (2004) modified the network definition so that all tanks were 95% full at the start of the peak electricity period (7:00 *am*). This latter formulation is the one we will use throughout this work.

2.6 Summary

This chapter has introduced the pump scheduling problem, its objectives and its constraints, through a general and formal definition, not tied to any specific application. The constraints most frequently considered in the literature have been discussed in detail. An extensive review of the literature on the pump scheduling problem has been carried out, classifying previous approaches according to several characteristics, such as the representation of schedules and optimisation technique considered by each of them. Our review has pointed out that, despite the wide range of approaches that have been proposed, the literature is lacking of comparative analysis of these approaches. In later chapters we will provide a comparative analysis of several techniques. Before that, in the next chapter, we will discuss in more detail the representation of schedules: we will examine existing representations and propose new alternative representations. Finally, this chapter has presented the hydraulic model, the test network instances and the common structure of the pump scheduling optimisation systems implemented in the following chapters.

Chapter 3

Representation of Pump Schedules

Pump control policies are usually defined in terms of their operations within a specified time horizon, typically 24 hours. Thus, a solution to the pump scheduling problem is any possible schedule of pumps for a predefined time period. A particular representation describes how a sequence of decision variables maps to a pump schedule. For the sake of clarity, we assume that the representation of a schedule is made up by the schedule of each single pump. Thus, the *sequence* of decision variables representing a solution to the pump scheduling problem has the following general form:

$$S = \{s^1, s^2, \ldots, s^{N^{\mathrm{p}}}\} \tag{3.1}$$

where each s^p corresponds to the *sequence* of decision variables representing the schedule of pump p. This formulation has the advantage that we can assume a single pump when discussing different representations, while extending the discussion to several pumps is straightforward.

A schedule can be represented either explicitly or implicitly. *Explicit* representations define schedules by directly specifying the status of each pump (Boulos *et al.*, 2001; Goldman & Mays, 2000; Mäckle, Savic & Walters, 1995; McCormick & Powell, 2003b, 2004; Pezeshk & Helweg, 1996; Sakarya & Mays, 2000; Savic, Walters & Schwab, 1997; Wegley, Eusuff & Lansey, 2000). Typically, *explicit* representations divide the operation period into several time intervals. For each time interval, a decision variable either indicates the status of the pump (active, idle or speed for variable frequency drive pumps) or the fraction of time a pump is operating during that time interval. On the other hand, *implicit* representations define the operations of pumps in terms of properties of other elements of the network (Atkinson *et al.*, 2000; Dandy & Gibbs, 2003; Kazantzis *et al.*, 2002; van Zyl, Savic & Walters, 2004). For example, water levels in tanks are often used to trigger operation of pumps. Hence, when using an implicit representation, the goal becomes one of determining optimal values of those surrogate variables.

In the following sections, the representations used during the rest of this thesis are explained in detail. First, two well-known and widely used representations are explained. One is the binary representation, an explicit representation in which the on/off status of a pump during a number of time intervals is encoded by a sequence of binary values. The other is an implicit representation, level-controlled triggers, where the on/off status of a pump depends on the water level of a tank. Next, we introduce two new explicit representations based on the concept of *time-controlled triggers*. Section 3.3 discusses how this concept is used to represent schedules in such a way that the constraint on the number of pump switches is implicitly satisfied.

Even though other representations have been suggested in the literature, as discussed earlier in our review of previous works (Section 2.3 on page 13), none of them has been shown to give a clear advantage over both the binary representation and level-controlled triggers in the general scheduling problem. In fact, they were typically developed to deal with specific requirements or limitations of the formulation of the problem, the optimisation algorithm or the simulation model. Moreover, some representations are extensions of either the binary representation or level-controlled triggers. Therefore, we focus on these two as a basis for comparison with the new proposed time-controlled triggers representation.

3.1 Binary Representation

The binary representation is the most commonly used explicit representation of pump schedules (Boulos *et al.*, 2001; Goldman & Mays, 2000; Mäckle, Savic & Walters, 1995; Savic, Walters & Schwab, 1997; Sotelo, von Lücken & Barán, 2002). In the binary representation, the scheduling period (T) is divided into a fixed number (N^{T}) of smaller intervals and a single bit is used to represent the status of a pump during each interval. The bit's value would be one if the pump is active during the time interval, or zero if the pump is idle.

$$s^p = \begin{array}{|c|c|c|c|c|c|} \multicolumn{1}{c}{t_0} & \multicolumn{1}{c}{t_1} & \multicolumn{1}{c}{t_2} & \multicolumn{1}{c}{t_3} & \multicolumn{1}{c}{t_j} & \multicolumn{1}{c}{t_{N^{\mathrm{T}}}} \\ \hline 0/1 & 0/1 & 0/1 & 0/1 & \cdots & 0/1 \\ \hline \end{array}$$

Figure 3.1: Binary representation for a single pump p and N^{T} time intervals.

Given a particular solution, the number of pump switches is the number of $\boxed{0\ 1}$ sequences plus one if the schedule starts with $\boxed{1}$ and ends with $\boxed{0}$. Thus, the maximum number of switches per pump is $N_p^{\mathrm{sw}} \leq \lfloor N^{\mathrm{T}}/2 \rfloor$. The size of the search space depends

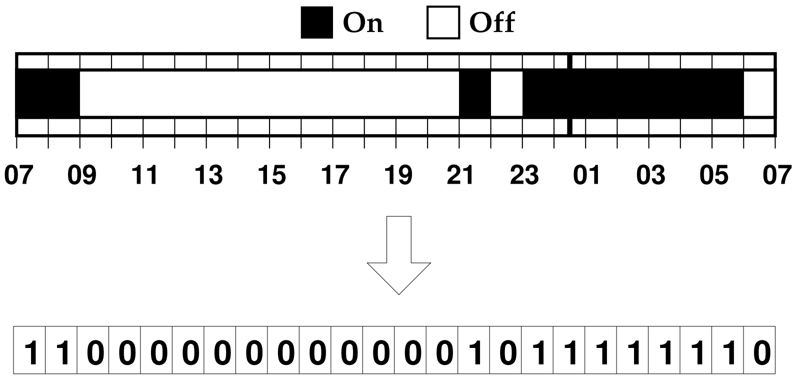


Figure 3.2: A pump schedule and the corresponding binary representation.

on both the number of time intervals and the number of pumps, being the total number of possible solutions $2^{(N^{\mathrm{T}} \cdot N^{\mathrm{P}})}$. That is, in the binary representation of the schedule of a single pump containing 24 one-hour intervals, there are at most 12 pump switches and the search space contains $2^{24} = 16\,777\,216$ candidate solutions.

Schedules represented by the binary representation are limited to switching pumps on/off at times that are multiple of ratio T/N^{T}. This limits the flexibility of the schedules. In the above example of 24 one-hour intervals, the status of a pump cannot change in the middle of a one-hour period, thus an operating interval may last two or three hours but not 2.5 hours. Of course, the flexibility can be increased just by using a larger N^{T}. On the other hand, this limitation prevents sudden consecutive changes on the status of pumps, since the minimum idle period between two operating intervals is T/N^{T}.

3.2 Level-controlled Triggers

The operation of a pump can be triggered at certain water levels of a storage tank. Typically, a pair of trigger levels, lower and upper, are set up in a tank such that when water falls below or goes above the respective level, the pump activates or stops (see Fig. 3.4). Level-controlled triggers are used to keep the water level in a tank within an operating range that is a reduced interval of a contractual range. If the level falls below or goes over the levels of this contractual range, then a penalty cost may exist. We must point out that this penalty cost does not exist in our approach, since the EPANET simulator, even when level-controlled triggers are not used, prevents the water level to fall outside the contractual range by closing the tank riser. When a tank cannot supply water, pressure at some of the demand nodes decreases, which may result in violations of pressure constraints.

Since electricity tariff is typically divided into peak and off-peak periods, different pairs of level-controlled triggers are used for each period (Dandy & Gibbs, 2003; van Zyl, Savic & Walters, 2004). Thus, a solution in level-controlled triggers representation has the following formulation:

	peak		off-peak	
pump p	l_{lo}	l_{up}	l'_{lo}	l'_{up}
	lower	upper	lower	upper

Figure 3.3: Level-controlled triggers.

Each trigger level is constrained to values within the contractual range of the tank. Additionally, for each pair of lower/upper trigger levels, the lower level is constrained to be always lower than the corresponding upper level ($l_{\text{lo}} \leq l_{\text{up}}$ and $l'_{\text{lo}} \leq l'_{\text{up}}$).

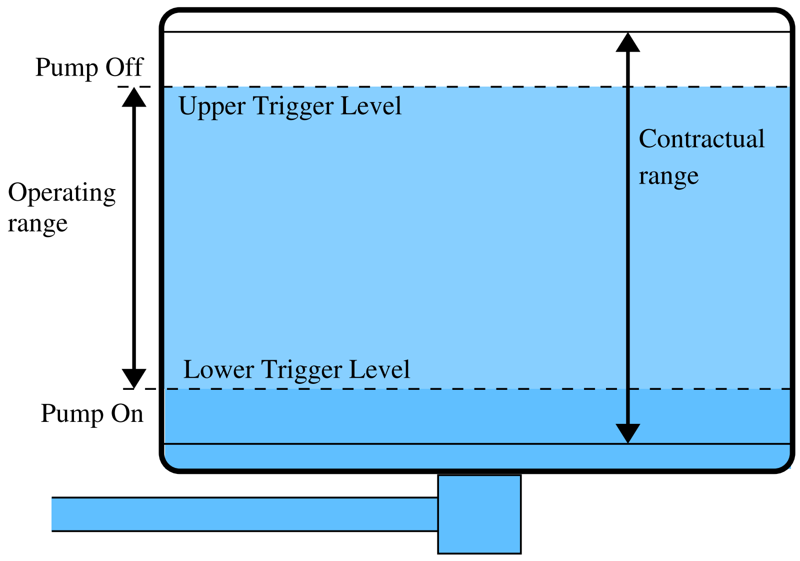


Figure 3.4: Trigger levels on a tank.

As for pump switches, there is no limit in the number of pump switches that the level-controlled triggers representation may generate. Apart from the fact that narrow operating ranges will, in general, result in more pump switches than wider ranges, there is no rule of thumb that can be applied to estimate the number of pump switches generated by a particular setting of level-controlled triggers. In fact, excessive number of pump switches is often a problem in applications using level-controlled triggers. This is sometimes acknowledged as a potential problem by the authors, but no strategies are considered to prevent it (Kazantzis *et al.*, 2002). In other cases, the schedules obtained by the optimisation algorithm are manually fine-tuned a posteriori to reduce the number of pump switches (Atkinson *et al.*, 2000). A better approach incorporates an explicit constraint on the number of pump switches into the optimisation algorithm and constraint handling techniques are applied, e.g., penalising the objective function (electrical cost) proportionally to the number of pump switches (van Zyl, Savic & Walters, 2004).

3.3 A New Representation for Pump Scheduling: Time-controlled Triggers

We propose a new explicit representation based on the concept of time-controlled triggers. This new representation has the advantage of satisfying the constraint on the number of pump switches implicitly.

In contrast to the binary representation, which encodes the status of a pump during each time interval, the *time-controlled triggers* representation encodes the time when a pump changes its status. The concept of encoding time has already been proposed in the literature. Sakarya & Mays (2000) and McCormick & Powell (2004) used continuous variables to encode the proportion of time that a pump (or combination of pumps) is active during a time interval. However, our proposal allows us to directly encode pump switches in the representation. A pump switch is defined as turning on a pump that was previously off (Lansey & Awumah, 1994), however in a *periodic* schedule each pump switch actually implies two changes in the status of a pump: *off* to *on* for the pump switch and *on* to *off* to achieve the situation previous to the pump switch. For example, a schedule where a pump is initially on and is never turned off during the whole scheduling period does not contain any pump switch or status change. On the other hand, a schedule where a pump is initially on and it is eventually turned off contains one pump switch and two status changes.

Therefore, a pair of decision variables is required to define a pump switch in time-controlled triggers representation. We can thus limit the maximum number of switches per pump simply by limiting the number of decision variables of each solution. For example, a schedule of a single pump in a *time-controlled triggers* representation of length six will allow a maximum of three pump switches. In general, for a maximum of SW switches per pump, there will be $2 \cdot SW$ decision variables for each pump.

$$s^p = \begin{array}{|c|c||c|c||c|c||c|c|}\hline t_1 & t'_1 & t_2 & t'_2 & \dots & \dots & t_{SW} & t'_{SW} \\ \hline\end{array}$$

Figure 3.5: Time-controlled triggers for pump p with SW pump switches.

This new representation enables the optimisation algorithm to conduct the search in a reduced search space. For example, let us consider the schedule of a single pump for 24 intervals of one hour. In the binary representation, each interval can have either of the two states (on/off) and, thus, there are 12 possible pump switches. The search space contains $2^{24} = 16\,777\,216$ candidate solutions. However, if the number of pump switches is restricted to three ($N_p^{\text{sw}} \leq 3$), the feasible search space with respect to this constraint is reduced to $290\,998$ solutions, which is less than 1.74% of the total search space. Table 3.1 gives the number of potential solutions with respect to the number of pump switches. These values were obtained by explicitly enumerating all 2^{24} possible schedules of a single pump.

Table 3.1: Search space size for the scheduling of a single pump in 24 hours with respect to various limits on the number of pump switches (SW).

SW	$N_p^{\text{sw}} = SW$		$N_p^{\text{sw}} \leq SW$	
	Feasible space	$\%$ of total	Feasible space	$\%$ of total
1	552	0.0033	554	0.0033
2	21252	0.1267	21806	0.13
3	269192	1.6045	290998	1.7345
4	1470942	8.7675	1761940	10.502
5	3922512	23.38	5684452	33.882
6	5408312	32.2361	11092764	66.118
12	2	0.00001	16777216	100

The following equation can be used to compute the number of solutions with exactly SW switches:

$$\text{No. of solutions} = 2 \cdot \binom{N^{\text{T}}}{2SW} = \frac{2 \cdot N^{\text{T}}!}{(N^{\text{T}} - 2SW)! \cdot (2SW)!} \tag{3.2}$$

where N^{T} is the number of time intervals.

The size of the search space grows with N^{T}. For example, when considering $N^{\text{T}} = 48$, that is, 30 minutes intervals in a daily scheduling period, there are $24\,934\,442$ possible schedules of a single pump with three or less pump switches. This is more than 85 times the number of solutions for $N^{\text{T}} = 24$. On the other hand, it is only 8.86×10^{-6} percent of the total number of possible schedules for a single pump with $N^{\text{T}} = 48$. That is, on the one hand, the size of the reduced search space grows with N^{T}, but on the other hand, it grows far slower than the total search space.

The above discussion focuses on a single pump. With additional pumps, the search space grows exponentially. For just two pumps there are $2^{24} \cdot 2^{24} = 2^{48}$ potential schedules in total. The number of solutions with exactly SW switches is similarly squared: $4 \cdot \binom{N^{\text{T}}}{2SW}^2$. That is, for $N_p^{\text{sw}} \leq 3$ there are $84\,679\,836\,004$ potential solutions, which is actually 0.03% of the total number. Again we see that, although the number of potential solutions increases dramatically for higher number of pumps, the reduced search space becomes a smaller fraction of the total search space.

The range of decision variables (t_i and t'_i in Fig. 3.5) depends on the precision with which we want to measure time and on the time horizon. If continuous values are used, then the precision is arbitrary. On the other hand, discrete values may be preferred in order to limit search space and because arbitrary time precision is impractical. Therefore, assuming a time horizon of 24 hours, the range of decision variables may be an integer within $[0, 86400]$ for a precision of seconds, $[0, 1440]$ for a precision of minutes, $[0, 144]$ for a precision of ten minutes, and $[0, 24]$ for a precision of hours. For simplicity and following the typical binary representation, we will assume intervals of one hour during our exposition.

The time-controlled triggers representation may be implemented in several ways depending on whether the time encoded by the representation is absolute time since the start of the scheduling period, or relative to a previous change of pump status:

1. **Absolute:** the decision variables are absolute time since the start of the scheduling period. For each pair $\langle t_i, t'_i \rangle$ of decision variables, one value gives the time when the pump is turned on and the other value is the time when the pump is turned off.

2. **Relative:** the decision variables are time intervals relative to the previous change of pump status. Each pair $\langle t_i, t'_i \rangle$ of decision variables represent the time for which a pump is inactive and active, respectively, during a pump switch.

Each of these implementations may lead to different results. Moreover, we do not want to impose an exact number of pump switches but a maximum limit. Therefore, there should be some mechanism to represent schedules with fewer switches than the number of pairs of decision variables. Finally, our main goal is to be able to represent any possible schedule within a maximum limit of pump switches. In the following sections, we will examine how to achieve this for each possible implementation of time-controlled triggers.

3.3.1 Absolute Time-controlled Triggers

When decision variables are absolute time, each decision variable represents the time from the start of scheduling period at which the status of a pump changes. Let us assume for now that pumps are off at the start of the scheduling period. We consider that t_i corresponds to a transition from *off* to *on*, and t'_i corresponds to a transition from *on* to *off*. Therefore, a pair of decision variables $\langle t_i, t'_i \rangle$ represents an operating interval during which the pump is active. An example of absolute time-controlled triggers representation is shown in Fig 3.6, where each value is a number of hours since the start of the scheduling period (7 *am*).

A possible formulation of the above description would be:

$$s^p = \{\langle t_1, t'_1 \rangle, \langle t_2, t'_2 \rangle, \ldots, \langle t_{SW}, t'_{SW} \rangle\} \tag{3.3}$$

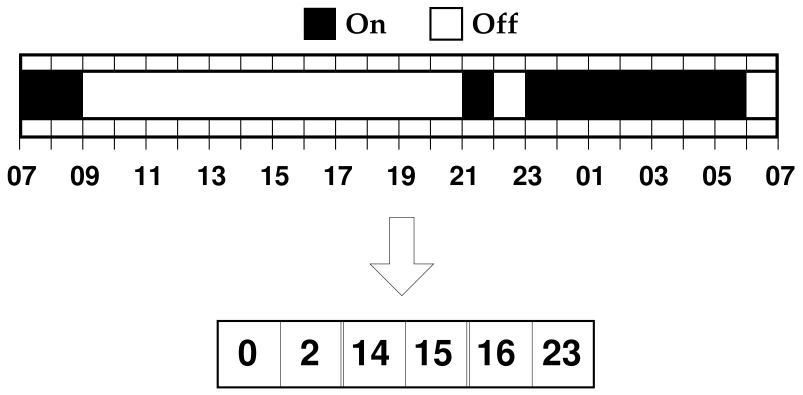


Figure 3.6: Example of absolute time-controlled triggers representation.

$$\forall i \in \{1, \ldots, SW\} \quad t_i, t'_i \in [0, T]$$
$$\forall i, j, k \in \{1, \ldots, SW\} \quad i < j < k \Rightarrow t_i < t'_i < t_j < t'_j < t_k < t'_k$$

The inclusion of the value 0 in the range of values makes unnecessary the previous assumption that pumps are off at the start of the scheduling period, since a pair $\langle 0, t'_1 \rangle$ represents an operating interval where the pump is active at the start of the scheduling period. However, this formulation is too strict in the sense that it cannot represent schedules with less than SW pump switches ($N_p^{\text{sw}} \leq SW$). In order to represent such schedules we need to introduce *empty operating intervals*, that is, an operating interval that has no effect whatsoever on the schedule apart from reducing the number of switches in a solution. Since all empty operating intervals have the same meaning independent of the particular values of t_i and t'_i, it is better to denote all empty operating intervals with a special symbol such as $\langle -, - \rangle$. Thus, we extend the formulation above in the following way:

$$s^p = \{\langle t_1, t'_1 \rangle, \langle t_2, t'_2 \rangle, \ldots, \langle t_{SW}, t'_{SW} \rangle\} \tag{3.4}$$

$$\forall i \in \{1, \ldots, SW\} \quad \langle t_i, t'_i \rangle \in \langle [0, T], [0, T] \rangle \cup \langle -, - \rangle$$
$$\forall i, j, k \in \{1, \ldots, SW\} \quad i < j < k \Rightarrow t_i < t'_i < t_j < t'_j < t_k < t'_k$$

3.3.2 Relative Time-controlled Triggers

If decision variables are relative time intervals, each pair of decision variables represents the time during which a pump is inactive and active, respectively. According to this, a pair of decision variables $\langle t_i, t'_i \rangle$ represents a single pump switch, since it implies a transition from an inactive status (during t_i) to an active one (during t'_i).

It immediately follows from this definition that the sum of all time intervals for each pump must be less than or equal to the scheduling period T. By allowing the sum to be less than T, we can represent schedules where the pump is not active at the end of the

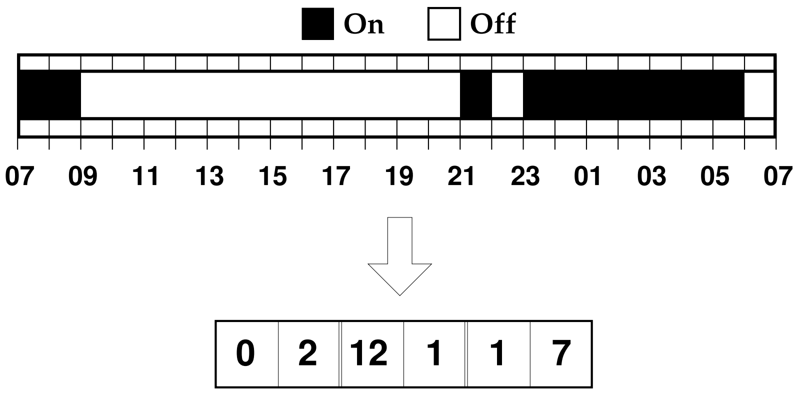


Figure 3.7: Example of relative time-controlled triggers representation.

scheduling period:

$$\sum_{i=1}^{SW}(t_i + t'_i) \;\leq\; T \tag{3.5}$$

When relative times are used in the time-controlled triggers representation, the most important implementation issue is the range of decision variables. First, let us assume that zero-length intervals are allowed for any decision variable, and thus the range of valid values is $[0, T]$. This formulation enables the representation of schedules with less than or equal to SW pump switches ($N_p^{\text{sw}} \leq SW$):

$$s^p = \{\langle t_1, t'_1\rangle, \langle t_2, t'_2\rangle, \ldots, \langle t_{SW}, t'_{SW}\rangle\} \tag{3.6}$$

$$\forall i \in \{1, \ldots, SW\} \quad t_i, t'_i \in [0, T]$$
$$\sum_{i=1}^{SW}(t_i + t'_i) \;\leq\; T$$

On the other hand, we could restrict the minimum value to be 1. For a solution where $(2 \cdot SW - 1)$ decision variables (all decision variables except one) have a value of 1, then, following constraint (3.5), the remaining decision variables must be lower than or equal to $(T - 2 \cdot SW + 1)$. Therefore, the range of values is within $[1, (T - 2 \cdot SW + 1)]$. Such formulation ensures the resulting schedule generates exactly SW switches ($N_p^{\text{sw}} = SW$):

$$s^p = \{\langle t_1, t'_1\rangle, \langle t_2, t'_2\rangle, \ldots, \langle t_{SW}, t'_{SW}\rangle\} \tag{3.7}$$

$$\forall i \in \{1, \ldots, SW\} \quad t_i, t'_i \in [1, (T - 2 \cdot SW + 1)]$$
$$\sum_{i=1}^{SW}(t_i + t'_i) \;\leq\; T$$

An example of relative time-controlled triggers representation is shown in Fig. 3.7.

Each value is the number of hours that the pump is either inactive or active.

3.4 Summary

This chapter reviewed the two most important representations used for the pump scheduling problem: binary representation and level-controlled triggers. In addition, a new representation based on explicit time-controlled triggers was proposed. We described two variants of the new representation, using either absolute or relative time. In the case of absolute time-controlled triggers, decision variables are the time at which a change in pump status occurs. On the other hand, in relative time-controlled triggers, decision variables are time intervals during which a pump is on or off. The main advantage of this new representation is implicitly enforcing the constraint on pump switches. This, in turn, leads to a notable reduction in the search space that needs to be explored by the optimisation algorithm. We expect that this reduced search space will further help the optimisation algorithm to quickly achieve a satisfactory pump schedule.

In the next chapter, experiments will be performed in order to compare these four representations. As far as we know, such a comparative analysis of different representations on the same network instances has not been performed in the literature.[1] In addition, different settings of the time-controlled triggers representation will be empirically investigated in order to fine-tune this new representation.

[1] Not even studies proposing specialised representations have performed such analysis. Instead, a new algorithm using the new representation is often compared with unoptimised settings or a previous algorithm with a different representation (Kazantzis *et al.*, 2002). In some cases, no comparison is done at all (Wegley, Eusuff & Lansey, 2000).

Chapter 4

Evolutionary Algorithms

In this chapter a very simple evolutionary algorithm, hence the name Simple Evolutionary Algorithm (SEA), is used to study the representations described in the previous chapter (binary representation, level-controlled triggers and two variants of the new time-controlled triggers representation). It is expected that parameters of an evolutionary algorithm will interact in different ways with each representation. This is particularly true in the case of recombination and mutation operators, since they inherently depend on the representation of solutions. Therefore, experiments will be carried out to fine-tune the parameters for all four representations. Moreover, new recombination and mutation operators will be developed for the proposed time-controlled triggers representations. A statistical analysis of the experiments will be conducted, taking into account possible interactions between different parameters, in order to fine tune SEA for each representation. The best results of each representation will be compared to determine their relative performance. Finally, the effect of different settings of the time-controlled triggers representation will be investigated by means of SEA. In particular, we will empirically study the effect of the limits on the number of pump switches and the number of time intervals on the performance of SEA. Larger values of either limit will provide more flexibility while constructing potential schedules, allowing, in principle, to generate finely tuned schedules with lower electrical costs. On the other hand, such larger values increase the search space, and it is reasonable to expect that a larger search space would increase the difficulty of finding lower cost schedules.

4.1 A Simple Evolutionary Algorithm

Evolutionary algorithms are inspired by the concept of natural evolution of genes. The algorithm introduced in this chapter for empirically studying various representations is a very basic evolutionary algorithm. It is henceforth called the Simple Evolutionary Algorithm (SEA). A description of the algorithm is shown in Fig. 4.1. The algorithm starts with the initialisation of the main population $\mathrm{P}_{\mathrm{all}}$ with α random solutions. Then, μ solutions from the main population are selected as parents using a binary tournament. A recombination operator is applied to pairs of parents in order to generate μ offspring solutions. Each of these offspring solutions may be further modified by a mutation operator. Mutation is applied to each decision variable of each offspring with a certain probability. The μ new solutions generated are evaluated to calculate the objective function value and constraints. These new solutions replace the μ worst solutions in the main population. As long as $\alpha > \mu$, the best solution found will always be present in the population, implementing *elitism*. Larger the difference between α and μ, stronger the elitism.

Apart from α and μ, the other parameters of SEA are the recombination and mutation operators, which are strongly influenced by the representation of solutions. In Section 4.1.2, evolutionary operators suitable for each representation will be discussed. For the binary and level-controlled triggers representations we use operators well-known in the evolutionary optimisation literature. For the time-controlled triggers we propose operators strongly inspired by those.

In addition to the evolutionary operators, a method to handle feasibility constraints (balance between supply and demand, pressure deficits and warnings from the simulator) must be defined.

4.1.1 Constraint Handling Methodology

In the pump scheduling problem there are a number of constraints that determine the feasibility of a solution (Section 2.2 on page 9). In particular, constraints on total volume deficit (Eq. 2.4) and minimum pressure requirements at demand nodes (Eq. 2.6) must be explicitly handled by the optimisation algorithm. In the proposed approach, warnings from the simulator (Section 2.4.1), which occur for example when a pump cannot deliver sufficient head, are also considered constraint violations.

Previous studies have dealt with constraints by penalising the objective function (Boulos *et al.*, 2001; Goldman & Mays, 2000; Mäckle, Savic & Walters, 1995; van Zyl, Savic & Walters, 2004). This requires the definition of a penalty function and appropriate penalty values. The penalty function method imposes a fixed trade-off between the amount of constraint violation and the value of the objective function. Low penalty

```
Require: $\alpha$ (population size), $\mu$ (number of parents and offspring),
    recombination_operator, and mutation_operator.
 1: $P_{\text{all}} \leftarrow$ generate $\alpha$ random solutions
 2: evaluate solutions in $P_{\text{all}}$
 3: $S^{\text{bf}} \leftarrow$ best solution from $P_{\text{all}}$
 4: while termination criteria not met do
 5:     $P_{\text{parents}} \leftarrow$ select by binary tournament $\mu$ solutions from $P_{\text{all}}$
 6:     $P_{\text{offspring}} \leftarrow$ apply recombination_operator to $P_{\text{parents}}$ to create $\mu$ solutions
 7:     apply mutation_operator to each solution in $P_{\text{offspring}}$
 8:     evaluate solutions in $P_{\text{offspring}}$
 9:     replace $\mu$ worst solutions from $P_{\text{all}}$ with solutions from $P_{\text{offspring}}$
10:     $S^{\text{bf}} \leftarrow$ best solution from $P_{\text{all}} \cup \{S^{\text{bf}}\}$
11: end while
12: return $S^{\text{bf}}$
```

Figure 4.1: Algorithmic schema of the Simple Evolutionary Algorithm (SEA).

values would allow constraint violations in return for small reductions in the objective value, while higher penalty values would require a larger decrease of the objective value to compensate the same amount of constraint violation. Moreover, different penalty values are required for different types of constraints and the degree of violation of some of these constraints cannot be easily quantified. Penalty values, in general, are obtained either using rudimentary techniques or by trial-and-error, requiring additional fine-tuning and experimental runs of the particular algorithm. Furthermore, penalty values that are optimal for one network instance are unlikely to be appropriate for a different network. Interactions between the optimal penalty values and other parameters are not clear, and it may be possible that different penalty values are optimal for different configurations of an optimisation algorithm.

For these reasons, the use of penalty function is avoided and a simpler and more general method based on ranking solutions with respect to their constraint violations and objective function values is adopted (Deb, 2000). In this ranking method, given two candidate solutions, the criteria to choose the best solution are:

1. select the solution with the lowest pressure violation (ΔH in Eq. 2.6);

2. if pressure violations are equal, select the solution with the lower number of warnings from the simulator;

3. for equal number of warnings, select the solution with the lower total volume deficit (ΔV in Eq. 2.4);

4. if total volume deficits are equal, select the solution with the lowest objective function value, that is, lowest electricity cost (C_E in Eq. 2.1).

These criteria effectively rank a feasible solution (zero total volume deficit, no warnings and no pressure violations) better than any infeasible one. Feasible solutions are compared with respect to their objective function values only, and infeasible solutions are compared according to their degree of infeasibility. The order chosen for the comparison of constraint violations establishes some preferences. A solution with a tank volume deficit, where enough water is supplied to meet the demand but a balance is not achieved at the end of the simulation, is preferred over a solution having pressure violations, where the adequate demand cannot be supplied. Warnings from the simulator are considered to be worse than a tank volume deficit, since warnings indicate some problem preventing the correct evaluation of the solution by the simulator (e.g., a pump was forcefully shut down because it could not deliver enough head). However, a solution that generates warnings and no pressure violations is preferred over a solution that has pressure violations. It was observed that a small modification to such a solution removes the warnings. On the other hand, a solution with pressure violations requires more fine-tuning, that is, pumps are required to be active for more hours to supply the demand at required pressures.

4.1.2 Evolutionary Operators

Evolutionary algorithms implement transformations of solutions that are equivalent to the mechanism of recombination and mutation of genetic material. These transformations are called recombination (or crossover) and mutation operators. Recombination is the operation by which at least two solutions are combined in order to generate a new offspring solution that inherits some of the characteristics of its parents. Mutation is the transformation of a single solution. It tends to be less disruptive than recombination, since the goal is to generate a fairly similar solution with small variation. The degree of variation is controlled by a probability of mutating each decision variable. A common setting in evolutionary computation is to perform one mutation per individual. In terms of probability, this corresponds to a mutation probability of one divided by the number of decision variables. There are many evolutionary operators described in the Evolutionary Computation literature. Michalewicz (1996) and Herrera, Lozano & Sánchez (2003) provide an extensive survey that together covers all the operators discussed henceforth. Since operators work directly on the representation of solutions, there are some differences between the operators used for each particular representation.

Binary Representation

For the binary representation, we focus on three well-known types of recombination: *one-point*, *two-point* and *uniform* crossover. In the case of one- and two-point recombination, the schedules are recombined per pump by using the same crossover point for each pump. That is, given a crossover point $k \in [1, N^{\mathrm{T}} - 1]$, the offspring schedule is formed by combining, for each pump p, the schedule of pump p from one parent up to time interval k and the schedule of the same pump from the other parent from time interval $k + 1$ up to N^{T}. This approach does not seem to have been explicitly used in previous algorithms using the binary representation, which simply divided the whole binary string containing the schedule of all pumps (Boulos *et al.*, 2001; Mäckle, Savic & Walters, 1995; Savic, Walters & Schwab, 1997). This latter approach disregards the interactions between the pumps and the fact that all pumps have an effect at the same time over the network. By comparison, our approach defines building blocks in terms of the status of all pumps during a time interval because the result of a simulation step is influenced by the combined status of all pumps at once. In a sense, the schedules of the pumps are applied "in parallel" to the network. Our intuition is that the proposed crossover captures better the "parallel" nature of the problem. Strictly speaking, our approach should be called one-point-per-pump and two-point-per-pump. However, for brevity we will refer to them simply as one-point and two-point crossover, since we only use this variant of point crossover in this thesis.

Mutation is performed using *flip* mutation operator, which reverses the status of a pump at a particular time interval. For each offspring solution, the mutation operator is applied to each time interval of each pump with a certain mutation probability. In the binary representation, there are $N^{\mathrm{p}} \cdot N^{\mathrm{T}}$ decision variables per solution, and hence, a probability of mutation of $1/(N^{\mathrm{p}} \cdot N^{\mathrm{T}})$ would mutate one time interval per solution. In earlier experiments, we found out that this mutation rate was too low, and better results were obtained with a mutation rate of two time intervals modified per solution, that is, with a mutation probability equal to $2/(N^{\mathrm{p}} \cdot N^{\mathrm{T}})$.

Level-controlled Triggers

In the case of level-controlled triggers representation, the variables are real numbers representing water level of a particular tank. Therefore, we use three well-known recombination operators for real-valued variables:

Rand-arithmetical, where each decision variable i in the offspring schedule c is calculated from the parent schedules a and b as $c_i = \lambda a_i + (1 - \lambda) b_i$, where λ is a random value between 0 and 1. It is also known as *line recombination* (Mühlenbein

& Schlierkamp-Voosen, 1993) or *arithmetical recombination* (Michalewicz, 1996, p. 128).

Average recombination is equivalent to arithmetical recombination with $\lambda = 0.5$, that is, $c_i = (a_i + b_i)/2$.

Extended intermediate recombination is also known as BLX-α with $\alpha = 0.25$ (Eshelman & Schaffer, 1992). Each offspring level c_i is a value randomly chosen from the interval $[c_{\text{min}} - I \cdot 0.25, c_{\text{max}} + I \cdot 0.25]$, where $c_{\text{max}} = \max\{a_i, b_i\}$, $c_{\text{min}} = \min\{a_i, b_i\}$ and $I = c_{\text{max}} - c_{\text{min}}$.

The following three mutation operators are tested in this study:

Uniform mutation, called *random mutation* by Michalewicz (1996), replaces a value by a new randomly uniform value from the allowed domain. In our case, if the value is an upper trigger, it is replaced by a random value between the lower trigger and the maximum level of the corresponding tank. For a lower trigger, the interval is from the minimum level to the upper trigger.

Replace mutation is a less restricted version of *uniform mutation*, where a trigger can take any value within the limits of the corresponding tank.

Gaussian mutation (Fogel, 1995) modifies a value by adding some amount of gaussian noise. In our case, we replace a trigger level c_i with a new value obtained from the normal (Gaussian) distribution $N(c_i, H_k^{\text{range}}/12)$. That is, the random distribution has mean equal to the original value and standard deviation equal to the difference (divided by 12) between the maximum and minimum water levels of the tank k ($H_k^{\text{range}} = H_k^{\text{max}} - H_k^{\text{min}}$) associated with the level-controlled trigger i. For the normal distribution about 99.7% of values are within three standard deviations. Hence, by choosing a standard deviation of $H_k^{\text{range}}/12$, we ensure that the mutation of a trigger value such $H_k^{\text{min}} + 0.25 \cdot H_k^{\text{range}} \leq c_i \leq H_k^{\text{max}} - 0.25 \cdot H_k^{\text{range}}$ will produce a result within the limits of the tank.

The above mutation operators are applied with a probability equal to one divided by the length of the solution, that is, $1/(4 \cdot N^{\text{p}})$, since for each pump there are four trigger levels. Hence, the expected number of trigger levels modified per solution is one.

After recombination and mutation, we ensure that the values do not exceed a tank's operational levels by setting values over the maximum to the maximum level and values below the minimum to the minimum level. Moreover, if the value of the lower trigger is higher than the value of the corresponding upper trigger, the values are exchanged.

Absolute Time-controlled Triggers

Custom recombination and mutation operators are required for the time-controlled triggers representation in order to maintain the implicit constraint on the number of pump switches and other representation constraints (see Section 3.3.1). We developed variants of *one-point*, *two-point*, *uniform* and *rand-arithmetical* recombination, and of *uniform* and *replace* mutation. These operators are applied for each pump's schedule in a solution. For *absolute time-controlled triggers*, the recombination operators are adapted as follows:

One-point recombination. Given a crossover point $k \in [1, 2 \cdot SW - 1]$, the schedule of each pump p in the offspring solution is obtained by joining the values from 1 to k of the first parent and the values from $k + 1$ to $2 \cdot SW$ of the second parent. For example, schedule of pump p in offspring c is obtained from two parents a and b with ($SW = 4$, $k = 3$) as shown in Fig. 4.2.

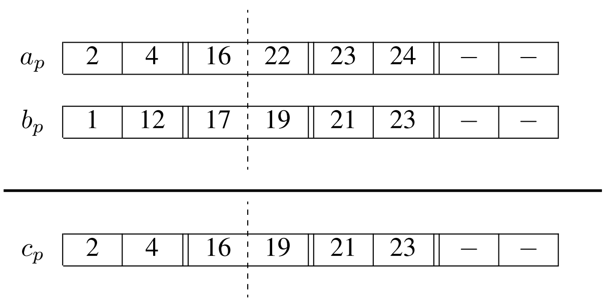


Figure 4.2: Example of one-point recombination for absolute time-controlled triggers representation.

This recombination may eventually generate solutions with invalid representations, since it may break the increasing order of the values. To keep the order, the values are sorted after recombination. This may result in two equal successive values. If the constraint on number of pump switches is of the form $N_p^{\text{sw}} \leq SW$, then those successive values are replaced by empty switches, such as $\langle -, - \rangle$, and moved to the end of the schedule. In case $N_p^{\text{sw}} = SW$ is used, then repeated values are iteratively modified by increasing the values by one until there are no more repeated values in the schedule. This one-point recombination can be straightforwardly extended to n-point recombination. Later, we experimentally study both *one-point* and *two-point* variants.

Uniform recombination. The idea of uniform crossover is modified to suit the absolute time-controlled triggers representation in the following way. First, combine the triggers for both parents and keep track of the status (on/off) of each parent at

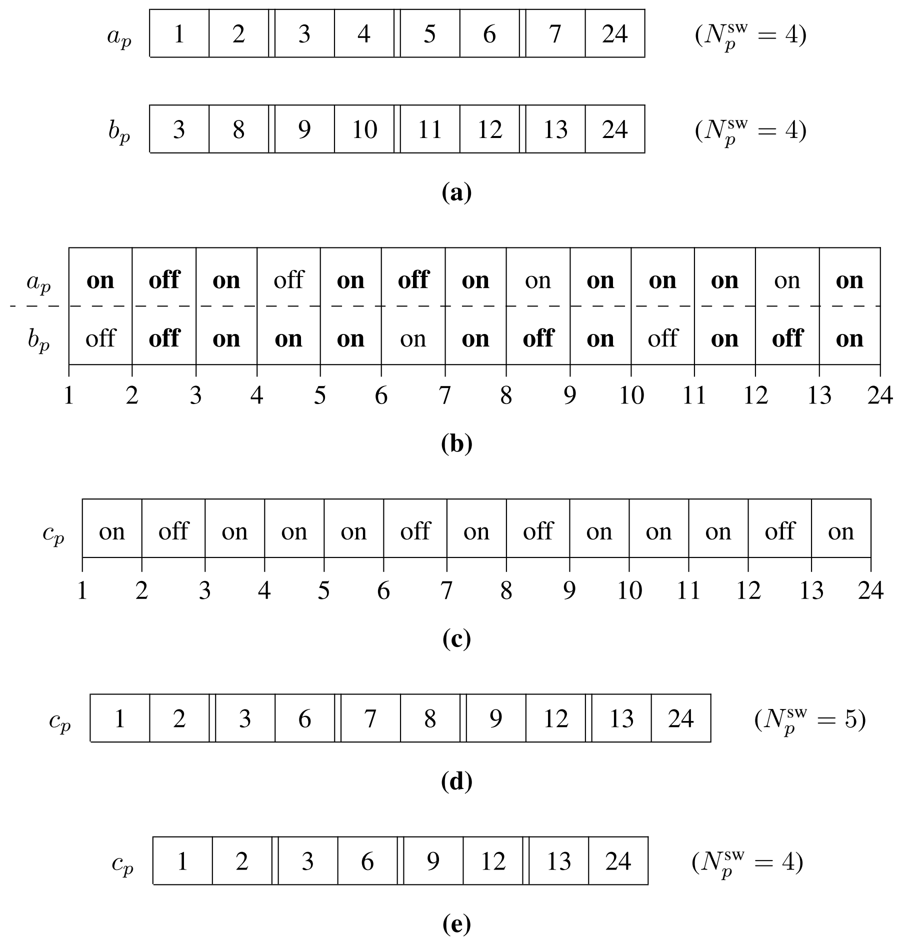

a_p | 1 | 2 || 3 | 4 || 5 | 6 || 7 | 24 | ($N_p^{\mathrm{sw}} = 4$)
b_p | 3 | 8 || 9 | 10 || 11 | 12 || 13 | 24 | ($N_p^{\mathrm{sw}} = 4$)
(a)
a_p on off on off on off on on on on on on on
b_p off off on on on on on off on off on off on
1 2 3 4 5 6 7 8 9 10 11 12 13 24
(b)
c_p on off on on on off on off on on on off on
1 2 3 4 5 6 7 8 9 10 11 12 13 24
(c)
c_p | 1 | 2 || 3 | 6 || 7 | 8 || 9 | 12 || 13 | 24 | ($N_p^{\mathrm{sw}} = 5$)
(d)
c_p | 1 | 2 || 3 | 6 || 9 | 12 || 13 | 24 | ($N_p^{\mathrm{sw}} = 4$)
(e)

Figure 4.3: Example of uniform recombination for absolute time-controlled triggers representation.

each time interval. Next, randomly select one status, following the traditional uniform recombination. Finally, merge contiguous time intervals with the same status into larger time intervals by removing the intermediate trigger values. The resulting trigger values are used to construct the offspring solution. The following example illustrates the procedure.

Let us assume the schedules of pump p in two parents a and b shown in Fig 4.3a. In absolute time-controlled triggers, each trigger value represents the time of the day at which a pump is turned on/off. Pumps initial status is assumed to be inactive (a pump initially active has 0 as its first trigger value). First, the trigger values of both parents are combined into one single time line (Fig 4.3b). The horizontal axis of this time line contains as many time intervals as different trigger values there are in the parents. Trigger values that do not appear in either parent are not taken into account. The time line describes the status (on/off) during each time interval of each parent given in the vertical axis. In the next step, for each time interval, one single status is randomly selected with equal probability among the status of the parents. Let us assume that the values marked in bold were the ones randomly selected, producing the combined time line shown in Fig 4.3c. Finally, contiguous intervals with the same on/off state are merged into larger time intervals, and the boundaries of the intervals are used to construct the time-controlled triggers representation of the offspring solution (Fig 4.3d).

The resulting schedule may have any number of pump switches, hence breaking the implicit constraint on pump switches. A lower number of pump switches is problematic only if constraint $N_p^{\text{sw}} = SW$. In such case, a pair of additional time-controlled triggers differing by just one time unit is repeatedly generated until the solution has the required number of switches. On the other hand, a number of switches higher than SW is always problematic. This is what occurs in Fig 4.3d. In this case, the trigger values corresponding to the shortest time interval are successively eliminated from the solution until it contains SW pump switches (Fig 4.3e).

Rand-arithmetical recombination. This is similar to the rand-arithmetical recombination described for level-controlled triggers. For each pump p, the offspring schedule c^p is calculated from the parent schedules a^p and b^p as $c_i^p = \lambda a_i^p + (1 - \lambda) b_i^p$, where λ is a random value between 0 and 1 and $i \in \{1, \ldots, 2 \cdot SW\}$. As a special case, if either a_i^p or b_i^p is part of an empty switch $\langle -, - \rangle$, then the other value is directly chosen (if both a_i^p and b_i^p are empty switches, the result is also an empty switch). As in the n-point recombination described above, offspring schedules may be invalid with respect to representation constraints. We use the same procedure as described above after recombination to satisfy these representation constraints.

The mutation operators used for this representation are *uniform* mutation and *replace* mutation:

Uniform mutation. The uniform mutation applied to level-controlled triggers is adapted to the absolute time-triggers representation as follows. The domain of a time-controlled trigger c_i^p is restricted to $[c_{i-1}^p + 1, c_{i+1}^p - 1]$, with special cases of $[0, c_2^p - 1]$ for $i = 1$, and $[c_{2 \cdot SW-1}^p + 1, 24]$ for $i = 2 \cdot SW$. If c_i^p is part of an empty switch, then we replace it with a new, randomly generated, pair of trigger values.

Replace mutation. This is a more aggressive mutation than *uniform* mutation. It replaces one trigger value with a uniform random integer in the range $[0, T]$. In the special case that the replaced value was part of an empty switch, the whole empty switch (two trigger values) is replaced by generating an additional random integer.

After mutation, representation constraints are enforced by repairing solutions using the same method as for recombination operators. Mutation is applied with a probability such that the expected number of time-triggers modified per solution is two, that is two divided by the number of decision variables. Since the number of decision variables is $2 \cdot SW$ trigger values per pump, we use a mutation probability of $2/(2 \cdot SW \cdot N^{\mathrm{p}})$. This value is larger than the typical one mutation per solution because, after repairing a mutated solution to satisfy representation constraints, the mutated solution may be equal to the original.

Relative Time-controlled Triggers

The same recombination and mutation operators used for absolute time-controlled triggers are also used for the variant based on *relative time*. Since the representation constraints are slightly different (see Section 3.3.2), the particular implementation of the operators is different as well. The recombination operators used for relative time-controlled triggers are:

> *N-point recombination*. This operator follows basically the same procedure as in the absolute time-controlled triggers representation, except the repair mechanism used if representation constraints are broken. In the experiments, we focus on *one-point* and *two-point* recombination.
>
> *Uniform recombination*. In this case we have implemented a simpler alternative than for the *absolute* time-controlled triggers. Here, the schedule of pump p in the offspring solution is obtained by randomly selecting, for each trigger, the value of either parent with equal probability, as shown in Fig. 4.4.

a_p	0	2	**12**	1	**1**	7
b_p	**1**	**5**	6	**0**	10	**1**
c_p	1	5	12	0	1	1

Figure 4.4: Example of uniform recombination for relative time-controlled triggers representation.

> *Rand-arithmetical recombination*. It is similar to the variant for absolute time-controlled triggers, with the difference that, in this case, we do not handle empty switches in any special way, since they are represented by the value 0.

We propose the following implementation of the mutation operators of relative time-controlled triggers:

Uniform mutation. For each mutated trigger, another trigger of the same pump is randomly chosen and their total time is redistributed randomly among them. For example, let us assume that mutation is applied to trigger k of a schedule of pump p denoted by c^p_k. First, a different random position j is generated. Next, the value of c^p_k after mutation (c'^p_k) is a random integer generated in the interval $[0, c^p_k + c^p_j]$ (or $[1, c^p_k + c^p_j]$ in case $N^{\text{sw}}_p = SW$ and $k \neq 0$). The new value for trigger j is $c'^j_j = (c^p_k + c^p_j) - c'^p_k$. Figure 4.5 gives an example with $k = 4$ and $j = 5$.

c_p	1	5	6	**0**	**10**	1
c'_p	1	5	6	6	4	1

Figure 4.5: Example of uniform mutation for relative time-controlled triggers representation.

Replace mutation. Similar to the previous implementations, *replace* mutation is meant to be a more aggressive variant of *uniform* mutation. A trigger value is replaced by an integer randomly generated from the interval $[0, T - (2 \cdot SW)]$ (or $[1, T - (2 \cdot SW)]$ if $N^{\text{sw}}_p = SW$ and $k \neq 0$).

The solution resulting after recombination or mutation may have a total amount of time greater than the scheduling period T, hence violating representation constraints. To repair such solutions, we iteratively reduce the value of the time intervals by one time unit until the total sum is equal to T.

Similar to the case of *absolute time-controlled triggers*, mutation is applied with a probability of $2/(2 \cdot SW \cdot N^{\text{p}})$, that is, two divided by the number of decision variables, the such that the expected number of time-triggers modified per solution is around two.

4.2 Experiments on Different Representations

The simplicity of SEA allows us to focus on the differences between the representations rather than in other algorithmic details. However, it is reasonable to expect performance differences depending on the values of the population size (α), parent/offspring population size (μ), and the particular recombination and mutation operators used. Therefore, we will first study the effect of different parameters for each representation However, it is not our goal to "*over-tune*" the algorithm for each representation and network instance. Hence, only a few reasonable values of α and μ will be tested in order to identify general trends rather than particular optimal settings.

In order to obtain a running time comparable to that considered by van Zyl, Savic & Walters (2004), we use the same termination criteria, that is, 6000 function evaluations per run for the Vanzyl network, and 8000 function evaluations per run for the Richmond network. Experiments with longer runs ($50\,000$ evaluations) are reported when testing the effect of the limit of pump switches (Section 4.4) and the number of time intervals (Section 4.5). Since the algorithms are stochastic, in order to assess the typical behaviour, each combination of parameters is repeated a number of times with different random seeds. For the Vanzyl network, after some initial testing, 25 repetitions for each parameter combination were considered to provide a sufficiently accurate median value. For the Richmond network the results showed less variability and each run is computationally more expensive (the network is much larger and complex). Therefore, 15 runs for each parameter combination were found to be sufficient for assessing the average behaviour.

The experimental results are analysed by techniques from the field of Experimental Design. Dean & Voss (1999), and Sheskin (2000) cover in detail the techniques discussed in the following paragraphs, but these are standard techniques that may be found in various textbooks on the subject. In particular, experiments are analysed by means of Analysis of Variance (ANOVA). ANOVA identifies which factors (or combination thereof) produce a statistically significant effect on the response variable. In our case, the factors are the parameters of SEA and the response variable is the electrical cost. A factor is significant if the probability of the factor not having an effect on the response variable is less than a given significance level. The p-value of ANOVA (or any other statistical test) is the smallest significance level that would identify the factor as significant. Therefore, low p-values are preferred. It is standard practice to consider a factor (or combination thereof) significant if the corresponding p-value is lower than 0.05 (5%).

ANOVA also indicates whether the effect of one parameter is conditioned by the settings of another parameter. For example, a large population size might compensate for lack of mutation. This is called an interaction and the model analysed includes all possible pairwise interactions among parameters. ANOVA does not measure how much the different settings of a significant parameter (or an combination thereof in the case of interactions) influence the response variable. Differences in mean electrical cost between parameter settings are measured by means of Tukey's Honest Significant Difference (HSD) 95% confidence intervals (Dean & Voss, 1999). These confidence intervals are incorporated to the interaction plots as error bars around the mean electrical value. An illustration example would be Fig. 4.6a. The error bars of $\alpha = 50$ and $\alpha = 100$ (both using *flip* mutation) overlap, and hence, we conclude that they do not have a (statistically significant) different effect on the electrical cost. On the other hand, there is no overlap among the other error bars, and hence, we conclude that the corresponding settings have a distinct effect on the resulting electrical cost.

The correctness of ANOVA depends on three assumptions about the data and the model being analysed. These requirements of ANOVA are denoted as independence, normality and homoscedasticity (equal-variance) assumptions. Precise details about their nature and the procedures for checking them are beyond the scope of this thesis. Suffice to say that we always check these assumptions before applying ANOVA. If any of them is not met, we apply standard procedures for correcting the problem. The correction procedures employed in this thesis involve removing one or more parameter settings from the experimental analysis (e.g., parameters that result in a strong effect on the electrical cost or large variability). Alternatively, standard transformations of the data may also be employed.

In some cases no transformation or correction would allow to meet ANOVA's assumptions, and ANOVA cannot be performed. Instead, we use boxplots to compare the parameter settings, and the (possible) interaction between two parameters. Typical boxplots are used to summarise a sample of data values. The line in the middle of the box corresponds to the median value. The "box" is delimited by the first and third quartiles, where the first quartile delimits the lowest 25 percent of the data and the third quartile delimits the lowest 75 percent of the data. Hence, the box contains at least 50 percent of the data. The height of the box corresponds to the inter-quartile range (IQR), which measures the variability of the sample. The extra lines above and below the box are called "whiskers" and they extend to the smallest value (respectively largest value) that is no more than $1.5 \cdot \text{IQR}$ times lower than the first quartile (respectively, higher than the third quartile). Any value beyond the two whiskers is called an outlier and is displayed as a point. In boxplots of interactions (see Fig. 4.15 for an example), points correspond to the median electrical value and only the "box", which contains 50% of the values, of each boxplot is shown. Whiskers and outliers are omitted for clarity. The "boxes" have different widths to appreciate overlapping boxes. Boxplots do not provide confidence levels (it cannot be said that one parameter setting is better than another with a confidence of 95%), however, they are useful to examine the distribution of the data and identify trends. In general, wide boxes (and whiskers) indicate high variability of the results, and the larger overlap between two boxes, the smaller is the difference between two parameter settings.

4.2.1 Binary Representation

The experimental setup of SEA using the binary representation considers all possible combinations of $\alpha = \{50, 100, 200\}$, $\mu = \{5, 20\}$, recombination and mutation operators. Recombination can be either *one-point*, *two-point*, or *uniform*, while both *flip* mutation and no mutation are tested.

The constraint on the number of pump switches ($N_p^{\text{sw}} \leq 3$, Eq. 2.8 with $SW = 3$) is explicitly incorporated to our constraint handling method, in addition to the other constraints discussed in Section 4.1.1. Out of two solutions that violate this constraint, the one containing the pump with the highest number of switches is considered worse. Moreover, this constraint is given a lower priority than the constraint on volume deficit, and thus it is considered after all the other constraints. Solutions satisfying this constraint are compared according to electrical cost.

The Vanzyl Network

The results of SEA with the binary representation for this network are summarised in Table A.1 on page 168. The Analysis of Variance (ANOVA) indicates that strong interactions exist between mutation and recombination, and between mutation and population size (α). The number of offspring solutions μ does not have a significant influence on the electrical cost. The relevant interactions are shown in Fig. 4.6.

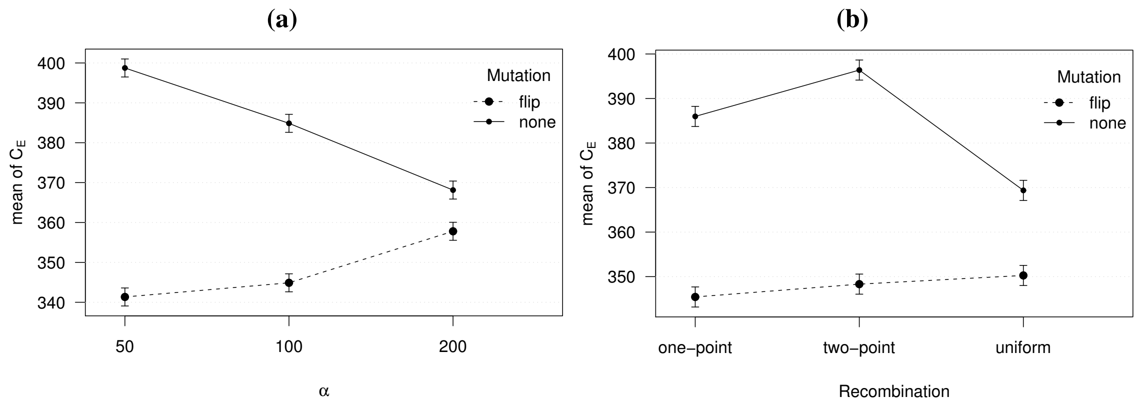


Figure 4.6: Interaction plots for SEA using the binary representation and a constraint of $N_p^{\text{sw}} \leq 3$ (the Vanzyl network).

Mutation and population size (α). Figure 4.6a indicates that, without mutation, a higher value of α is preferable and the best results were obtained with *flip* mutation and lower values of α. The higher variability provided by a larger population slightly compensates for lack of mutation. However, using mutation and a smaller population is a better approach.

Recombination and mutation operators. In Fig. 4.6b we have almost opposite behaviours with and without mutation. Without mutation, *uniform* recombination is much better than other recombination operators, whereas when using mutation the differences between various recombination operators are rather small.

In summary, the best combination of parameters of SEA using the binary representation is $\alpha = 50$, *one-point* recombination, and *flip* mutation, which is essential in order to achieve satisfactory results. In contrast, the offspring population size ($\mu = \{5, 20\}$) does not have a significant effect on the results.

The Richmond Network

We apply the same parameter setup of SEA to the Richmond network and the results are summarised in Table A.2 on page 169. The use of mutation is even more important for this network. In fact, some runs without mutation are not able to find a pump schedule that satisfies the volume deficit constraint. Therefore, we focus the analysis on those runs that use *flip* mutation.

The results do not satisfy ANOVA assumptions initially. By transforming the data using the reciprocal function ($x' = -1/x$), ANOVA requirements are satisfied. We perform ANOVA on the transformed results and it identifies the interaction between the population size (α) and the recombination operator as having a significant effect on the results. The interaction plot in Fig. 4.7 shows that the combination of $\alpha = 200$ and *uniform* recombination performs particularly worse than other combinations of parameters. Similar behaviour was also observed in the case of the Vanzyl network. The interaction plot shows that the best value for α is 50. Moreover, with $\alpha = 50$, performance of various recombination operators is not significantly different. For the purpose of choosing a better configuration of parameters, we plot all combinations with $\alpha = 50$ and *flip* mutation in Fig. 4.8. The combination of $\mu = 5$ and *one-point* recombination (plus $\alpha = 50$ and *flip* mutation) is selected as the best combination of parameters, since it obtains the lowest median value and a small variability.

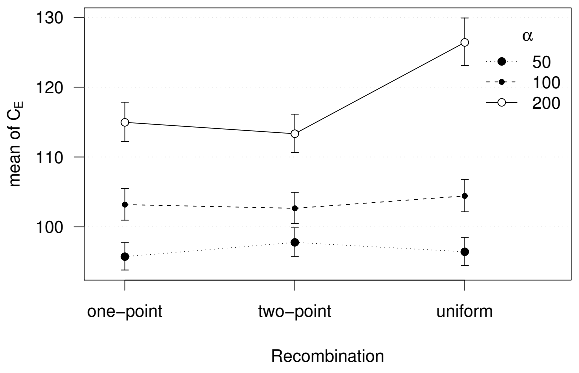


Figure 4.7: Interaction plot of SEA using the binary representation and a constraint of $N_p^{\text{sw}} \leq 3$ (the Richmond network).

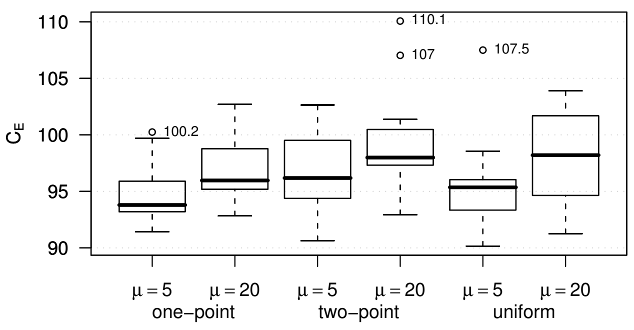


Figure 4.8: Results of SEA with $\alpha = 50$ and *flip* mutation using the binary representation and a constraint of $N_p^{\text{sw}} \leq 3$ (the Richmond network).

4.2.2 Level-controlled Triggers

For the representation based on level-controlled triggers we perform experiments using *average*, *rand-arithmetical* and *extended-intermediate* recombination operators, and *uniform*, *gaussian*, *replace* and *none* mutation operators. Population sizes of $\alpha = \{50, 100, 200\}$, and offspring population sizes of $\mu = \{5, 20\}$ were used in these experiments. An explicit constraint on the number of pump switches is added so that solutions with more than three switches per pump are penalised ($N_p^{\text{sw}} \leq 3$). This constraint is implemented in the same way as for the binary representation, that is, by adding a new condition to our constraint handling method that applies to solutions with equal volume deficit.

Experimental results are shown in Tables A.3 and A.4 on page 170, for the Vanzyl and Richmond networks, respectively. Only the results corresponding to *extended-intermediate* recombination are presented because all experiments using either *average* or *rand-arithmetical* recombination generated solutions with volume deficit for at least one run. On the other hand, runs using *extended-intermediate* recombination were always able to obtain solutions with zero volume deficit. Therefore, from now on, we restrict our analysis to the results obtained by using *extended-intermediate* recombination. We analyse separately the results for the Vanzyl and Richmond networks to identify which parameters produce a significant effect on the output of SEA.

The Vanzyl Network

We apply ANOVA to the results summarised in Table A.3. The only significant factor identified by ANOVA is the interaction between mutation operator and α. The interaction plot (Fig. 4.9) shows that for $\alpha = 50$, *replace* mutation produces a statistically distinct effect on the electrical cost. On the other hand, without mutation, $\alpha = 50$ performs significantly worse than $\alpha = 200$, which is somehow expected since the latter provides more diversity.

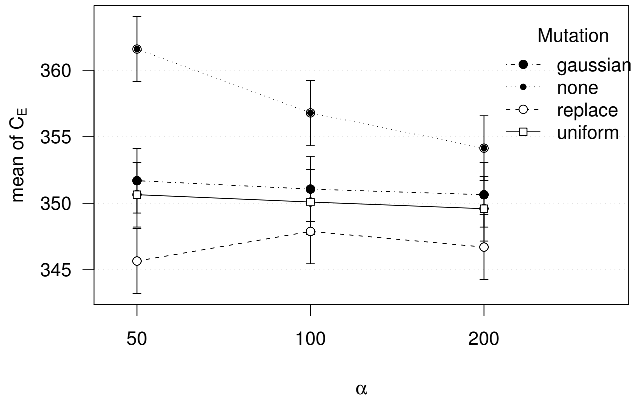


Figure 4.9: Interaction plot of SEA using level-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ (the Vanzyl network).

The Richmond Network

In the case of the Richmond network, the results suggest that the effect of mutation is even stronger (Table A.4 on page 170). Both *gaussian* mutation and no mutation (*none*) obtain very high electrical costs in the worst case. The strong differences between *gaussian* and *none* on one side, and *replace* and *uniform* mutation on the other, would actually hide the effect of other parameters. This strong effect can clearly be observed in the boxplot in Fig. 4.10. Therefore, the following analysis will focus on the results obtained using *uniform* and *replace* mutation.

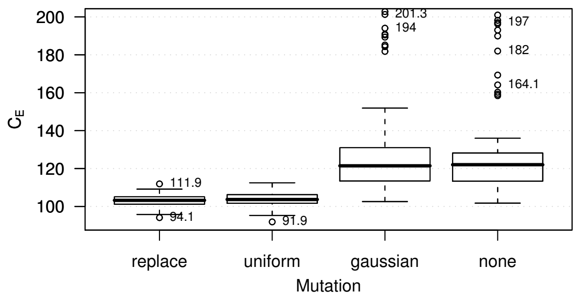


Figure 4.10: Boxplot of SEA using level-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ (the Richmond network).

We do not apply ANOVA here because the data does not strictly conform to the normality assumption. Even if we accepted the deviation from normality as small and performed ANOVA, it would not indicate a strong influence of any parameter or interaction thereof.

Instead, we study the results for each combination of parameters (Fig. 4.11). We observe that some combinations are better than the others, e.g., $\mu = 20$, $\alpha = 50$, and *replace* mutation, obtains better results than using $\alpha = 200$ in most runs. However, there is no discernible pattern among the parameters. Hence, the configuration with the lowest median ($\mu = 20$, $\alpha = 50$, *replace* mutation and *extended-intermediate* recombination) was chosen for comparison among representations.

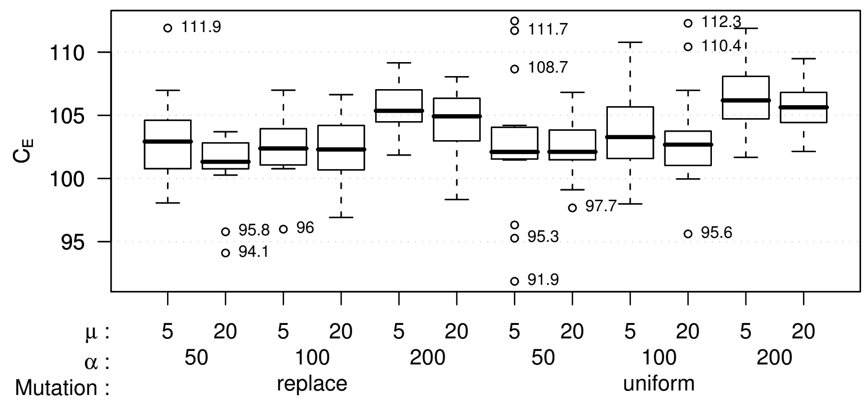


Figure 4.11: Boxplot of SEA results using level-controlled triggers and a constraint of $N_p^{\mathrm{sw}} \leq 3$ (the Richmond network).

4.2.3 Absolute Time-controlled Triggers

The experimental setup in the case of absolute time-controlled triggers representation is $\alpha = \{50, 100, 200\}$, $\mu = \{5, 20\}$, *uniform*, *one-point*, *two-point* and *rand-arithmetical* recombinations, and *uniform*, *replace* or *none* mutations. As it was done for the previous representations, the number of switches per pump (N_p^{sw}) is constrained to be less than or equal to three. However, this constraint does not need to be handled explicitly because the time-controlled triggers representation implicitly enforces this constraint.

Results of the experiments for the Vanzyl and Richmond networks are presented, respectively, in Tables A.5 on page 171 and A.6 on page 173. Before performing any statistical analysis, we notice that mutation seems to have a strong influence on the results of both networks. By plotting the distribution of the results for each mutation operator, as in Fig 4.12, we can conclude that not using mutation is clearly detrimental, irrespective of the other parameters. Therefore, we perform ANOVA only on the results that use mutation, to appreciate the interactions between the other parameters.

The Vanzyl Network

ANOVA of the SEA results obtained for the Vanzyl network identifies three interactions that have a significant effect on the electricity cost of the resulting schedule. In particular, the interactions between: recombination operator and population size (α); mutation operator and α; and recombination operator and mutation operator. These interactions are examined in detail in the following paragraphs.

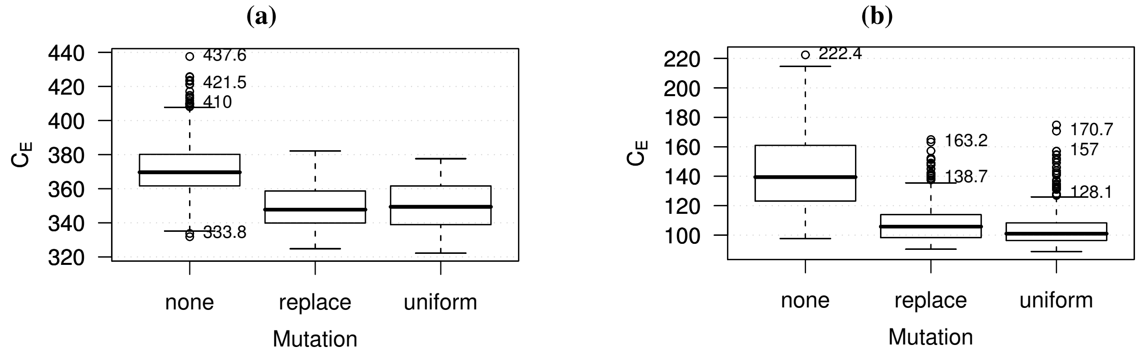


Figure 4.12: Boxplot showing the effect of mutation on SEA when using absolute time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ for (a) the Vanzyl network and (b) the Richmond network.

Recombination operator and population size (α). As shown in Fig. 4.13a, the most notable result is that *uniform* recombination generates worse electrical cost than other recombination operators, independently of the value of α. In the case of *rand-arithmetical* and *two-point* recombinations, $\alpha = 200$ is statistically worse than the other possible settings of α. On the other hand, the results of $\alpha = \{50, 100\}$ and *one-point*, *two-point*, and *rand-arithmetical* are not statistically different.

Mutation operator and population size (α). Figure 4.13b also shows that $\alpha = 200$ gives poor quality results. As for mutation, it does not seem to have any effect except for $\alpha = 50$, where *replace* mutation is better than *uniform* mutation.

Mutation operator and recombination operator. It can be observed from Figure 4.13c that *uniform* recombination is clearly worse than other recombination operators. Furthermore, the combinations of *one-point* recombination plus *replace* mutation, and *two-point* recombination plus *uniform* mutation are both significantly better than any combination of parameters using *rand-arithmetical*, because their corresponding confidence intervals do not overlap.

From the above analysis, the best parameters would be $\alpha = 50$, *replace* mutation, and *one-point* or *two-point* recombination. The offspring population size (μ) does not play a significant role in the performance of SEA when using absolute time-controlled triggers. For further comparison, we choose the configuration with $\alpha = 50$, $\mu = 20$, *two-point* recombination and *replace* mutation. This combination obtains both a low median electrical cost (although not the lowest) and a low variability, as shown in Fig. 4.14.

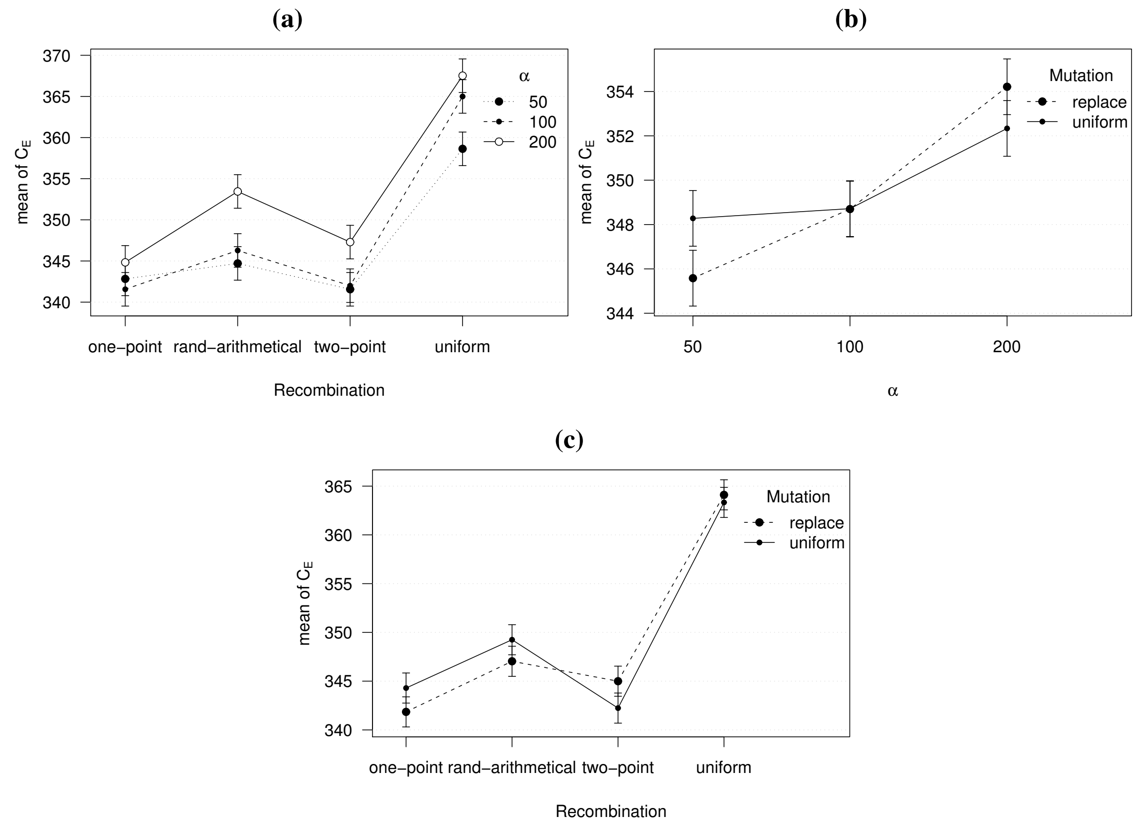


Figure 4.13: Interaction plots for SEA using absolute time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ (the Vanzyl network).

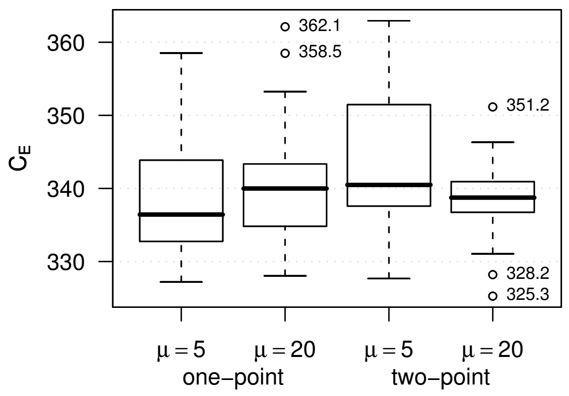


Figure 4.14: Boxplot of results of SEA using absolute time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ on the Vanzyl network with $\alpha = 50$ and *replace* mutation.

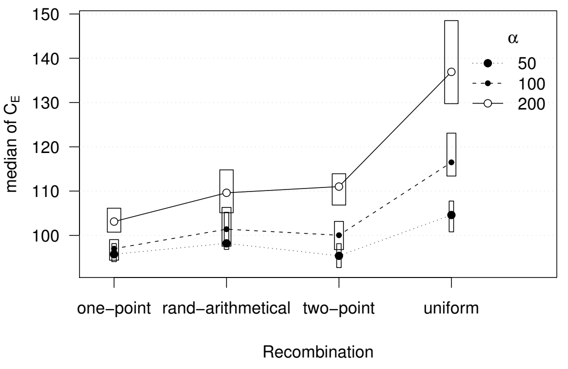


Figure 4.15: Boxplot of SEA using absolute time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ for the Richmond network (excluding *none* mutation).

The Richmond Network

In the case of the Richmond network, the results of SEA do not satisfy the requirements of ANOVA, even after removing the results corresponding to not using mutation and applying various transformations.

We explore the data by plotting the distribution of electricity cost for various combinations of parameters. Fig. 4.15 indicates that, given a value of α, *uniform* recombination always produces worse results than the other recombination operators. Excluding those runs using *uniform* recombination from our analysis and checking the ANOVA assumptions again, we conclude that the data sufficiently meets the requirements and proceed with ANOVA.

ANOVA identifies the interactions between α and recombination, mutation and α, and mutation and recombination, as having a significant effect on the results. We examine these interactions in more detail in the following paragraphs.

Population size (α) and recombination operator. As illustrated by Fig. 4.16a, a small population size ($\alpha = 50$) seems preferable compared to larger values ($\alpha = 200$). Moreover, *one-point* and *two-point* recombination are superior to *rand-arithmetical* recombination when $\alpha = 50$.

Population size (α) and mutation operator. Figure 4.16b indicates that a small population size generates better results than a setting of $\alpha = 200$. Although there are significant differences between the two mutation operators for large values of α (in favour of *uniform* mutation) this is not true any more for $\alpha = 50$.

Mutation and recombination operators. Figure 4.16c shows that when using *replace* mutation, the best results are obtained with *one-point* recombination, which is not affected by the mutation operator. On the other hand, when using *uniform* mutation, there is no clear winner among the recombination operators.

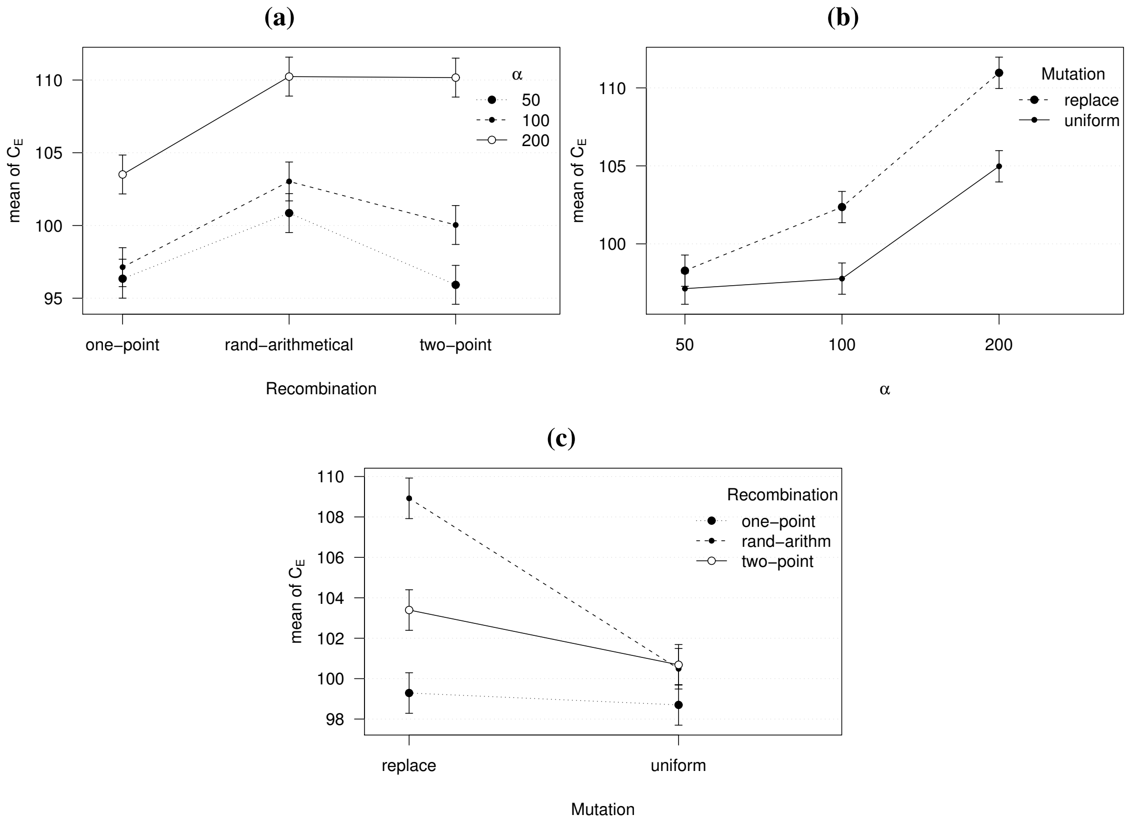


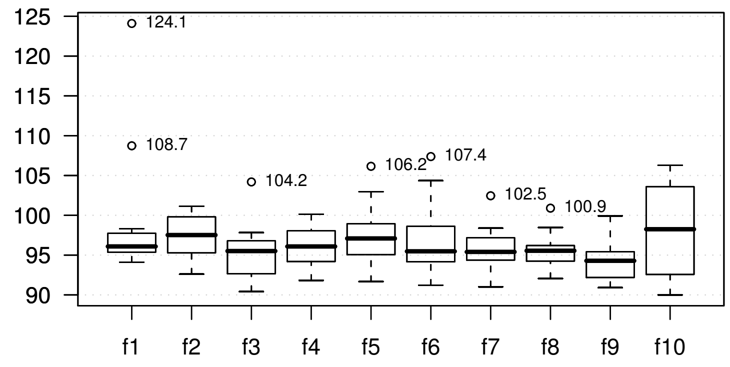
Figure 4.16: Interaction plots for SEA using absolute time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ (the Richmond network).

According to the above analysis, the best combination of parameters for SEA when using absolute time-controlled triggers is $\alpha = 50$, *one-point* or *two-point* recombination, and *uniform* or *replace* mutation. Among these possible configurations of SEA, there is no clear winner. In order to make a decision, we examine some combinations of parameters with $\alpha = 50$ (Fig. 4.17). As expected, the results are very similar. We choose the combination (f3 in the plot) $\mu = 5$, *one-point* recombination and *replace* mutation, given its low median value, small worst case (ignoring the single outlier) and excellent best case.

4.2.4 Relative Time-controlled Triggers

The same experiments performed using absolute time-controlled triggers are also performed for relative time-controlled triggers. That is, $\alpha = \{50, 100, 200\}$; $\mu = \{5, 20\}$; *uniform*, *one-point*, *two-point* and *rand-arithmetical* recombination; and *uniform*, *replace* and no mutation (*none*). The number of switches per pump is limited to three ($N_p^{\text{sw}} \leq 3$). This constraint is implicitly handled by the time-controlled triggers representation.

Experimental results are summarised in Tables A.7 on page 175 and A.8 on page 177



	μ	Recomb	Mutation
f1	5	rand-arith.	uniform
f2	20	rand-arith.	uniform
f3	5	one-point	replace
f4	20	one-point	replace
f5	5	one-point	uniform
f6	20	one-point	uniform
f7	5	two-point	replace
f8	20	two-point	replace
f9	5	two-point	uniform
f10	20	two-point	uniform

Figure 4.17: Boxplot of the best combinations of SEA using absolute time-controlled triggers and a constraint of $N_p^{\mathrm{sw}} \leq 3$ (the Richmond network).

for the Vanzyl and Richmond networks, respectively. The first observation is that mutation is essential to achieve satisfactory results for both networks. As shown in Fig. 4.18, the runs using mutation are clearly better than those without mutation, independent of the other parameters. Therefore, the following analysis includes only those results obtained using either *uniform* or *replace* mutation.

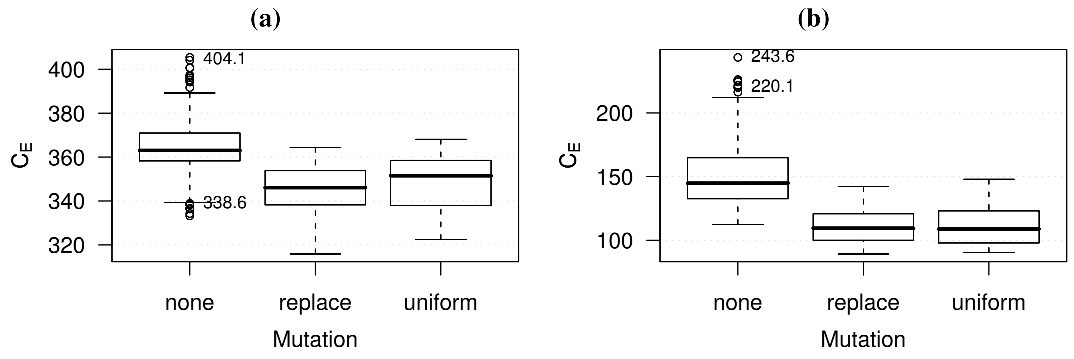


Figure 4.18: Boxplot showing the effect of mutation on SEA when using relative time-controlled triggers and a constraint of $N_p^{\mathrm{sw}} \leq 3$ for (a) the Vanzyl network and (b) the Richmond network.

The Vanzyl Network

For the Vanzyl network, before applying ANOVA, we notice that the results of SEA with *rand-arithmetical* recombination are better than those obtained using other recombination operators, independent of the other parameters of SEA (see Fig. 4.19). In order to confirm this observation, we perform a non-parametric statistical multiple-samples test, Kruskal-Wallis one-way analysis of variance by ranks (Sheskin, 2000), on the hypothesis that the median electrical cost is the same for the four recombination operators. This hypothesis is rejected by the test with a p-value close to zero. Then, we perform pairwise nonparametric two-sample tests (Wilcoxon rank-sum test) on the hypothesis that the median electrical cost is the same for two recombination operators. These tests give a p-value less than 0.05 only when comparing with *rand-arithmetical* recombination. We conclude that the median electrical cost obtained by SEA using *rand-arithmetical* recombination is lower than the median obtained with any other recombination operator. Therefore, we focus on the results obtained using *rand-arithmetical* recombination.

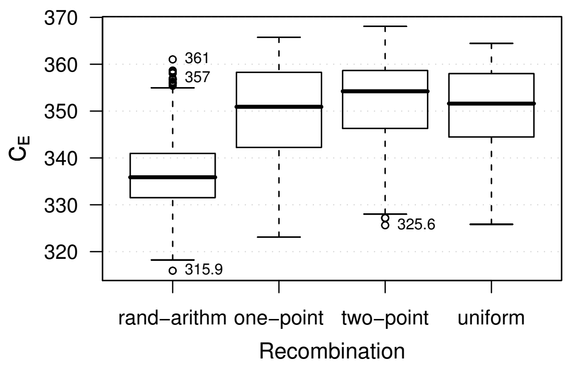


Figure 4.19: Boxplot of the results of SEA using relative time-controlled triggers and a constraint of $N_p^{\mathrm{sw}} \leq 3$ (the Vanzyl network).

We cannot apply ANOVA because the requirements are not satisfied, even after trying several transformations of the data. In order to assess the best parameters, we examine in more detail the results in Fig. 4.20. The only discernible pattern is that $\alpha = 200$ typically produces worse results than using 50 or 100. Apart from those, the apparent differences are very small or are restricted to a particular configuration. We identify a *best* configuration of parameters for further comparison based on a preference for a low median electrical cost and a small variability, in order to obtain a good value most of the time and a reasonable worst case. Therefore, following the boxplot in Fig. 4.20, the best choices would be either $\alpha = 50$, $\mu = 20$ and *replace* mutation or $\alpha = 100$, $\mu = 20$ and *uniform* mutation. Although the latter configuration obtained a slightly smaller median, we choose the former because of its small variability, which denotes more consistent results.

The Richmond Network

For the results of SEA for the Richmond network, before applying ANOVA, we also observe that some parameters have a strong influence. In particular, we observe that α has a strong influence on the performance of SEA. The *rand-arithmetical* recombination seems to produce better results than other recombination operators. We examine the combined effect of these two parameters in Fig. 4.21. There is a substantial degradation of the

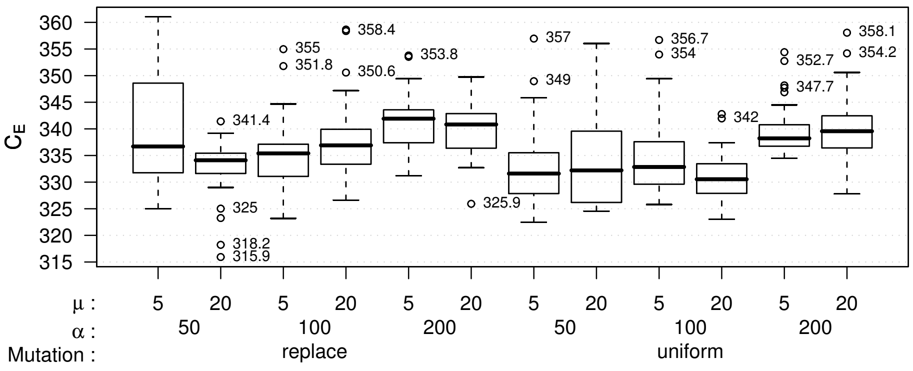


Figure 4.20: Boxplot of SEA with *rand-arithmetical* recombination using relative time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ (the Vanzyl network).

results as α increases for any recombination operator, although it is less marked for *rand-arithmetical* recombination. Furthermore, given a particular value of α, *rand-arithmetical* is always the best recombination operator.

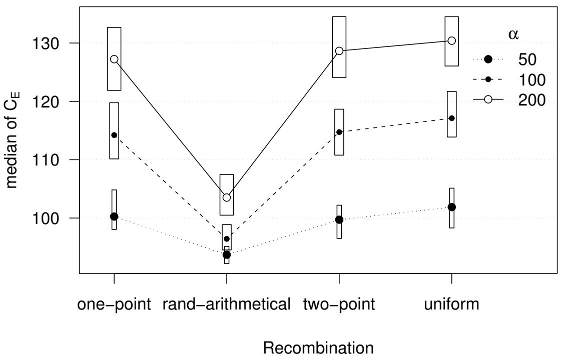


Figure 4.21: Boxplot of SEA using relative time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ (the Richmond network).

We therefore focus our analysis on the results obtained using *rand-arithmetical* recombination and $\alpha = \{50, 100\}$. This subset of data does not satisfy ANOVA assumptions. Instead, we analyse the results for each configuration of the parameters in Fig. 4.22. We observe that the results using *replace* mutation and $\alpha = 100$ are worse than other configurations of parameters. On the other hand, the results using *uniform* mutation are only slightly affected by the settings of α and μ. Among these combinations, we choose the combination $\alpha = 50$, $\mu = 5$ and *uniform* mutation as the best configuration of SEA for this representation.

4.3 Comparison among Representations

In the previous sections we have examined different combinations of the parameters of SEA for each representation and each network instance. The best combinations of pa-

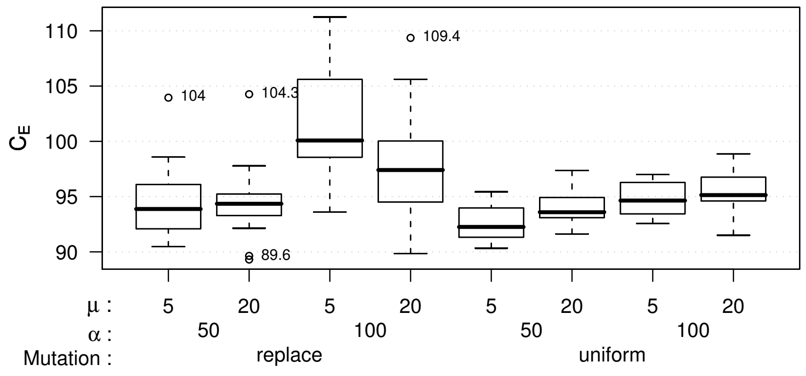


Figure 4.22: Boxplot of SEA using relative time-controlled triggers and a constraint of $N_p^{\text{sw}} \leq 3$ with *rand-arithmetical* recombination (the Richmond network).

rameters are shown in Table 4.1 for the Vanzyl network and Table 4.2 for the Richmond network. Figures 4.23 and 4.24 compare graphically the different representations with the results obtained by the Hybrid GA proposed by van Zyl (2001). The results of Hybrid GA shown here are those published by van Zyl, Savic & Walters (2004). The Hybrid GA was designed for optimising level-controlled triggers, and hence, we are not comparing algorithms directly but rather alternative representations. The Hybrid GA uses *one-point* crossover and a mutation operator similar to *replace* mutation. For comparison purposes, the results shown here were obtained after the same number of evaluations, that is, 6000 evaluations for the Vanzyl network and 8000 for the Richmond network.

From these results, it can be concluded that the binary and time-controlled representations outperform both SEA and HybridGA with level-controlled triggers. In addition, SEA with time-controlled triggers using relative time values obtains a lower median electrical cost and a lower variability than using absolute time values. Although the binary representation obtains results similar to relative time-controlled triggers, the latter ensures that the constraint on pump switches is always satisfied. The binary representation has to search for feasible schedules satisfying this constraint. Table 4.2 indicates that SEA with the binary representation did not find a feasible schedule satisfying $N_p^{\text{sw}} \leq 3$ in at least half of the runs in the Richmond network because the median N^{sw} is larger than three times the number of pumps ($N^{\text{p}} = 7$). Both Tables 4.1 and 4.2 show that the number of pump switches obtained when using the binary representation is higher than when using relative time-controlled triggers.

Our conclusion is that the proposed relative time-controlled trigger representation is better than level-controlled triggers and the binary representation, with the added benefit that time-controlled triggers always enforce a maximum number of pump switches. Moreover, the comparison with HybridGA shows that a carefully chosen representation, with the appropriate recombination and mutation operators, may make a simplistic algorithm,

Table 4.1: Comparison of different representations for SEA in the Vanzyl network.

		Representation			
		binary	**level-triggers**	**time-triggers**	
				absolute	**relative**
SEA	α	50	50	50	50
	μ	5	5	20	20
	Recomb.	one-point	ext.-interm.	two-point	rand-arithm.
	Mutation	flip	replace	replace	replace
C_{E}	median	333.0	346.9	338.7	334.1
	sd	11.2	5.3	5.6	6.1
	min	324.7	337.2	325.3	315.9
	max	359.6	357.1	351.2	341.4
N^{sw}	median	7.0	3.0	6.0	5.0
	sd	1.2	0.9	1.3	1.2
	min	5.0	2.0	4.0	3.0
	max	9.0	6.0	8.0	7.0

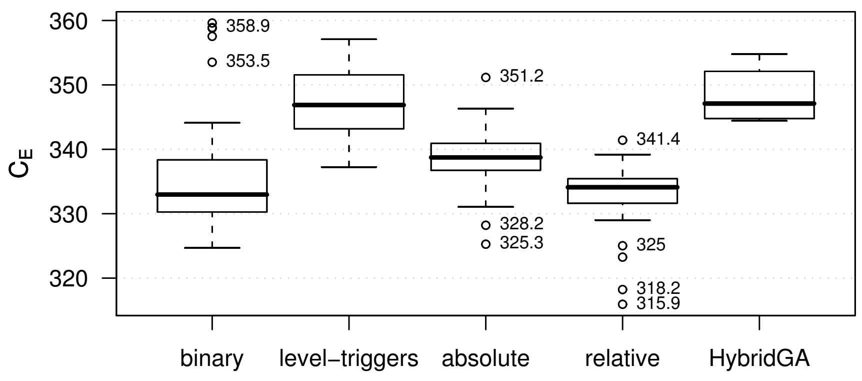


Figure 4.23: Boxplot of the results of SEA in the Vanzyl network for different representations.

Table 4.2: Comparison of different representations for SEA for the Richmond network.

		Representation			
		binary	**level-triggers**	**time-triggers**	
				absolute	**relative**
SEA	α	50	50	50	50
	μ	5	20	5	5
	Recomb.	one-point	ext.-interm.	one-point	rand-arithm.
	Mutation	flip	replace	replace	uniform
C_{E}	median	93.8	100.0	95.5	92.3
	sd	2.6	2.5	3.4	1.6
	min	91.4	99.1	90.4	90.3
	max	100.2	107.0	104.2	95.4
N^{sw}	median	26.0	10.0	13.0	16.0
	sd	3.1	1.5	1.1	1.7
	min	19.0	8.0	12.0	14.0
	max	32.0	13.0	15.0	20.0

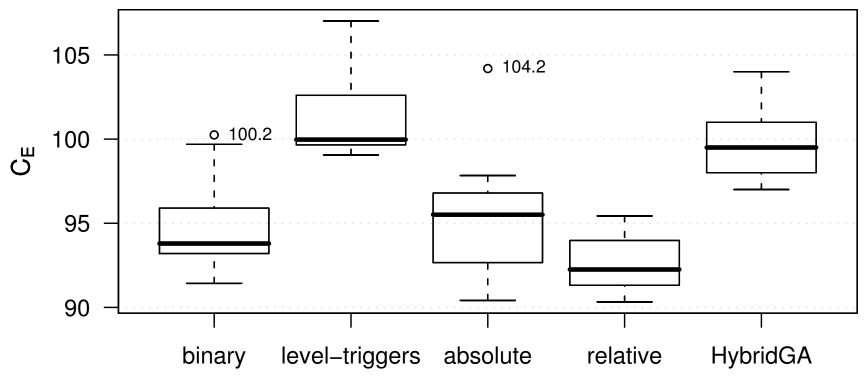


Figure 4.24: Boxplot of the results of SEA for the Richmond network for different representations.

such as SEA, outperform a more advanced and complex algorithm, such as HybridGA.

4.4 Effect of the Constraint on the Number of Pump Switches

In the following sections, we investigate in more detail the effect that the constraint on the number of pump switches has on the performance of the evolutionary algorithm SEA. First, we compare the results obtained by SEA using the binary representation with and without constraint on the pump switches to identify: (*i*) whether the optimal parameters are different depending on the use of the constraint; (*ii*) whether the electrical cost is further reduced; and (*iii*) how much the number of pump switches increases when no limit is enforced. Later, using the time-controlled triggers representation, we investigate the effect of maximum number of pump switches.

4.4.1 Binary Representation

Experiments using the binary representation without restrictions on the number of switches will be conducted to understand the benefits of allowing large number of pump switches. First, we perform experiments with the same parameter setup used in Section 4.2.1. We use the same three types of recombination (*one-point*, *two-point* and *uniform*) and *flip* mutation.

The Vanzyl Network

For the Vanzyl network, we tested population sizes of $\alpha = \{50, 100, 200\}$ and offspring size of $\mu = \{20\}$. The complete results are provided in Table A.9 on page 178. ANOVA cannot be performed on the results because the homoscedasticity assumption is not met. Figure. 4.25 shows boxplots of the results of SEA for the different values of mutation against the other parameter settings. In all cases, the variability of the results is much higher when no mutation is used.

Population size (α) and mutation operator. The use of mutation completely changes the effect of α, as shown in Fig. 4.25a. When using mutation, smaller population sizes work best. On the other hand, without mutation, $\alpha = 200$ obtains the best results. However, the results obtained with $\alpha = 200$ and no mutation have a large variability. The "box", which contains 50% of the values, ranges from 330 to 360. In contrast, when using $\alpha = 50$ and *flip* mutation, 50% of the results are contained within 342 and 348. The latter is preferable, since a low variability ensures more predictable results.

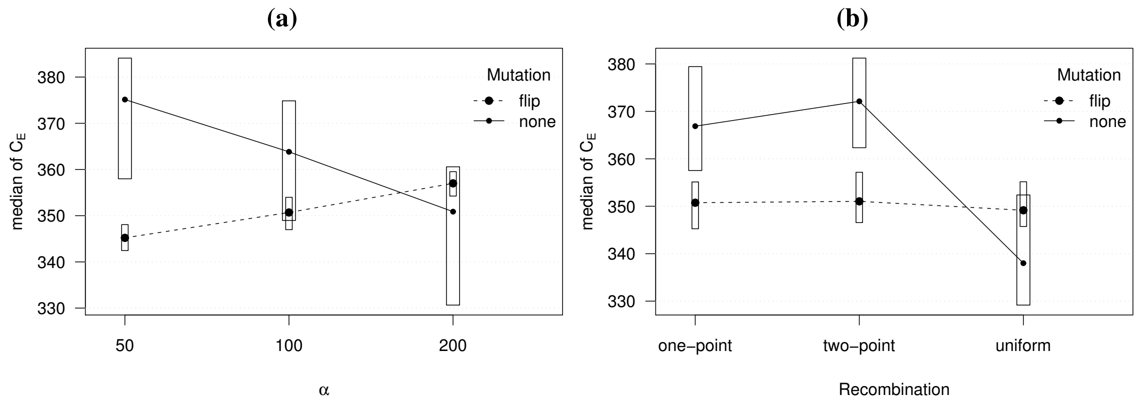


Figure 4.25: Boxplots of SEA using the binary representation and no constraint on the number of pump switches (the Vanzyl network).

Recombination and mutation operators. Figure 4.25b shows that when using mutation, the particular recombination operator is of little importance. However, without mutation, *uniform* recombination seems to be better. The median value obtained by SEA when using *uniform* recombination and no mutation is much lower than when using any other recombination, with or without mutation.

The above analysis indicates that the configuration with $\alpha = 200$, *uniform* recombination and no mutation is particularly good. However, it is not clear whether it is better than combinations with $\alpha = 50$ and *flip* mutation, due to the high variability of results when not using mutation. Figure 4.26 shows the electrical cost (C_{E}) and total number of pump switches (N^{sw}) obtained by configurations of SEA using *uniform* recombination. The median electrical cost and variability of $\alpha = 200$ and no mutation are much lower than for any other combination of parameters. This is in contrast with the optimal parameter settings when using a constraint on the pump switches (Section 4.2.1), which were $\alpha = 50$, *one-point* recombination and *flip* mutation. Another interesting observation from Fig. 4.26 is that the number of pump switches tends to decrease along with the electrical cost. This strongly suggests that the schedules with the lowest electrical cost do not have a high number of pump switches.

The Richmond Network

In the case of the Richmond network, we arrived at similar conclusions as those reached for the Vanzyl network. The summary of the experimental results of SEA are given in Table A.10 on page 179. Important differences in median value and variance depending on the combination of parameters can be observed in the table.

ANOVA cannot be performed because its statistical assumptions are not satisfied. How-

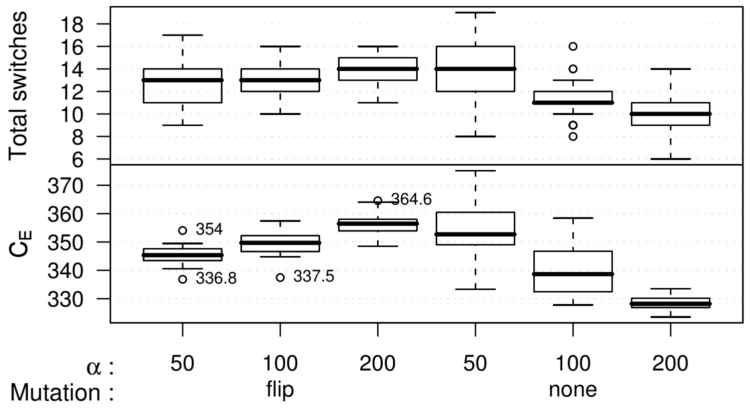


Figure 4.26: Boxplot of SEA using the binary representation and no constraint on the number of pump switches with *uniform* recombination (the Vanzyl network).

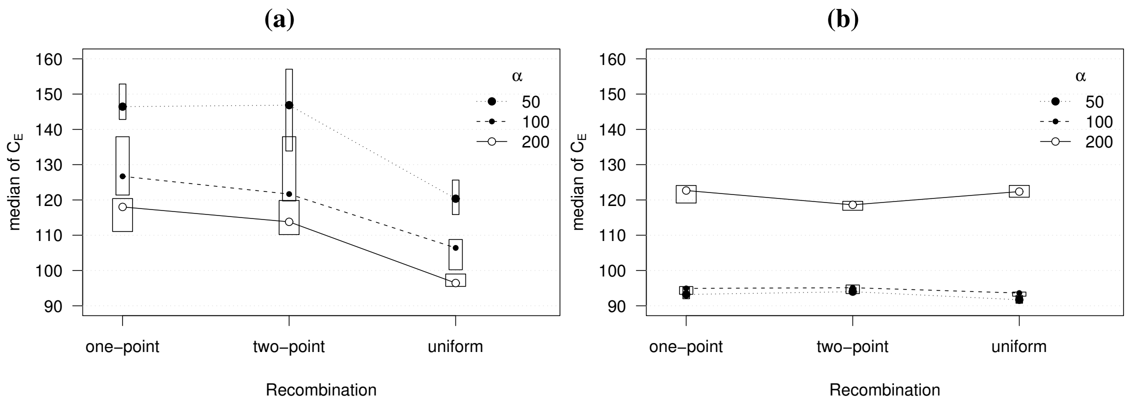


Figure 4.27: Boxplots of SEA using the binary representation and no constraint on the number of pump switches (the Richmond network) (a) without mutation and (b) with flip mutation.

ever, there are interactions between the parameters, as illustrated by Fig. 4.27, which shows simplified boxplots for different combinations of recombination, α, and with and without mutation.

When no mutation was used. Figure 4.27a shows that higher values of α produced lower electrical cost. The best combination is $\alpha = 200$ and *uniform* recombination.

When *flip* mutation was used. Figure 4.27b, indicates that $\alpha = 200$ generated much worse results than $\alpha = \{50, 100\}$. The particular recombination operator does not have a clear influence.

The above analysis suggests that $\alpha = 200$ has opposite effects depending on whether mutation is used. This interaction between α and mutation is similar to the one observed in the case of the Vanzyl network. At the first glance, one may conclude that for both networks the use of *flip* mutation would seem counter-productive. However, as shown in

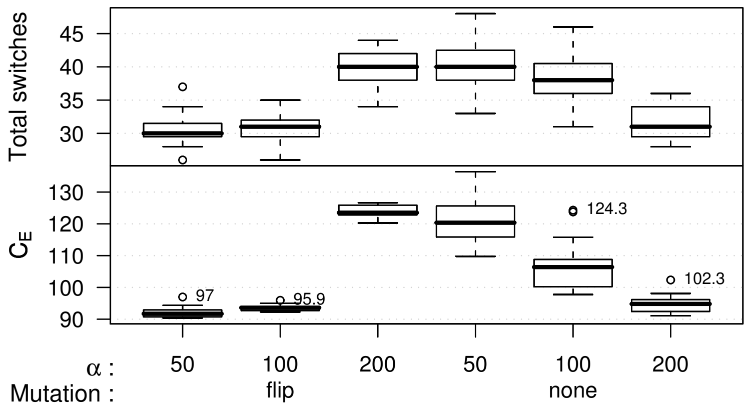


Figure 4.28: Boxplot of SEA using the binary representation and no constraint on the number of pump switches with *uniform* recombination (the Richmond network).

Fig. 4.28, the use of *flip* mutation generates the best results for the Richmond network, but it must be combined with a lower value of α.

Finally, similar to the Vanzyl network, schedules with less pump switches are also those with lower electrical cost, which contradicts the notion that a higher flexibility on the switching of pumps would allow lower cost schedules. In fact, this suggests that there is an optimal number of pump switches and schedules with more pump switches break additional constraints or increase electrical costs.

Number of Pump Switches Without a Constraint

Despite not using an explicit constraint on the number of pump switches, the median total number of switches (N^{sw}) of the results shown in Tables A.9 and A.10 are surprisingly low. Taking into account that the Vanzyl network has three pumps and the Richmond network has seven pumps, the average number of switches per pump (N_p^{sw}) is between four and five. These are very low numbers if we take into account that the binary representation allows up to a maximum of 12 switches per pump. This indicates that higher number of switches do not necessarily mean lower electrical cost. On the contrary, the worst results typically have a higher number of pump switches than the best results.

The average number of switches per pump above was calculated simply by dividing the total number of switches by the number of pumps. In the actual schedules generated by the SEA algorithm, the lack of a constraint on the number of switches may produce higher number of switches for some pumps while other pumps are switched fewer times. This is observed in the schedule shown in Fig. 4.29 corresponding to the median solution obtained by SEA ($\alpha = 50$, $\mu = 5$, *uniform* recombination and *flip* mutation) in the Richmond network. Although in this example the average number of switches per pump is below five, most pumps contain five switches. Moreover, pump 2A contains six

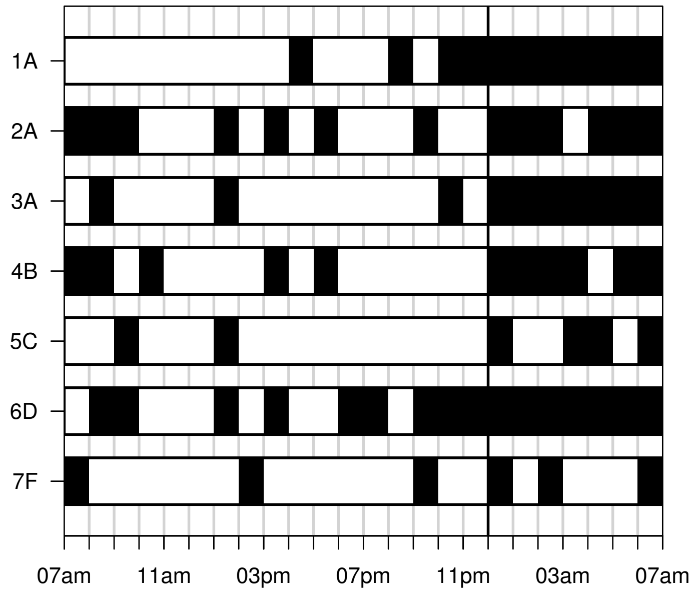


Figure 4.29: Schedule obtained by SEA for the Richmond network using the binary representation and no constraint on the number of pump switches.

switches, while pump 1A only contains three. Therefore, it is important to restrict the number of switches of each pump by including a constraint such as Eq. 2.8, as discussed in Section 2.2.

Effect of a Explicit Constraint on the Number of Pump Switches

By comparing the results of the best configurations of SEA using the binary representation without and with the constraint on pump switches, we observe an interesting difference between the two network instances. In Fig. 4.30a, corresponding to the Vanzyl network, the best results are obtained when using the constraint. The opposite occurs in the Richmond network, as shown in Fig. 4.30b, where the results obtained without constraint are slightly better than those obtained when using the constraint.

The result for the Vanzyl network suggests that the increased flexibility obtained by allowing a higher number of pump switches does not lead to better schedules. On the other hand, in the case of the Richmond network, there are schedules with lower electrical cost that require more switches per pump than those allowed by the constraint.

So far, we have examined the explicit constraint on the number of pump switches added to the binary representation. In the next section, we will analyse the effect of the implicit constraint enforced by the use of time-controlled triggers representation.

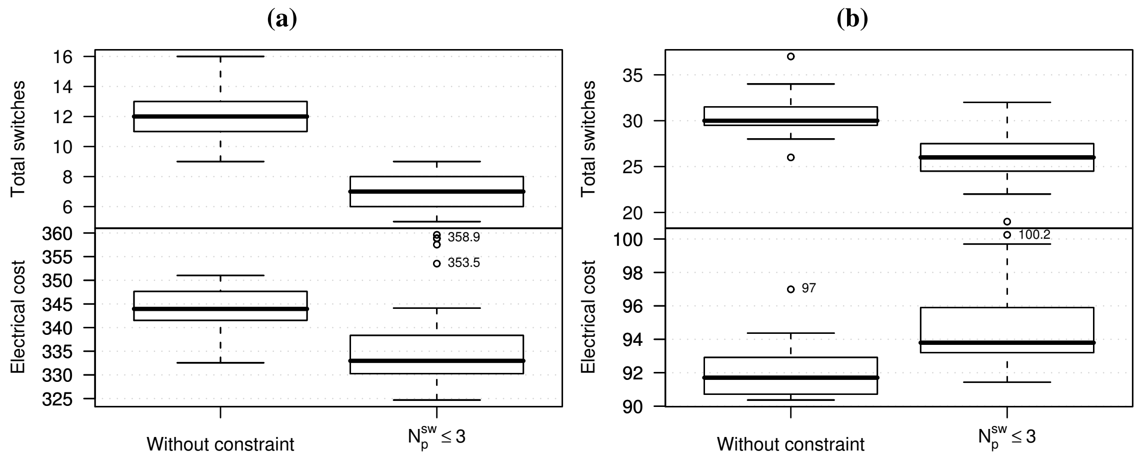


Figure 4.30: Results of best configurations of SEA using the binary representation with and without constraint on pump switches in (a) the Vanzyl network and (b) the Richmond network.

4.4.2 Time-controlled Triggers

One of the features of the time-controlled trigger representation, as discussed in Section 3.3, is that the constraint on the number of pump switches is implicitly enforced. The maximum number of pump switches becomes a parameter SW that defines the representation of schedules. In this section, we study the effect of this parameter.

First, we want to find out whether there are differences in the optimal value that can be obtained given a fixed number of switches per pump. Therefore, we perform very long runs of SEA with a strict constraint in the number of pump switches ($N_p^{\text{sw}} = SW$) varying from $SW = 1$ to $SW = 5$ switches per pump. The configuration used is $\alpha = 50$, $\mu = 5$, *one-point* recombination and *replace* mutation, with absolute time-controlled triggers representation. The results after $50\,000$ evaluations are shown in Figures 4.31 and 4.32 for the Vanzyl and Richmond networks respectively. With just one switch per pump the results are clearly worse than with higher number of switches, since the algorithm does not have much flexibility as to when operate the pumps. However, for two or three switches per pump, the results improve dramatically. As the number of switches per pump further increases, the results become slightly worse. This fact suggest that the lowest cost schedules with two or three switches are easier to find than the lowest cost schedules with higher number of switches. In summary, the growth of the search space caused by increasing the number of pump switches over a particular limit negatively affects the performance of the algorithm, and it is not compensated for by a greater number of low cost schedules.

We next looked at whether the particular setting of the maximum number of pump switches negatively affects the performance of the algorithm, even when using a constraint

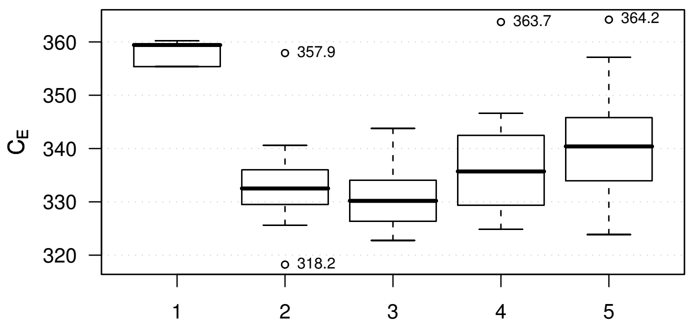


Figure 4.31: Effect of different values of SW in constraint $N_p^{\text{sw}} = SW$ in the Vanzyl network. (SEA with absolute time-triggers representation, $50\,000$ evaluations).

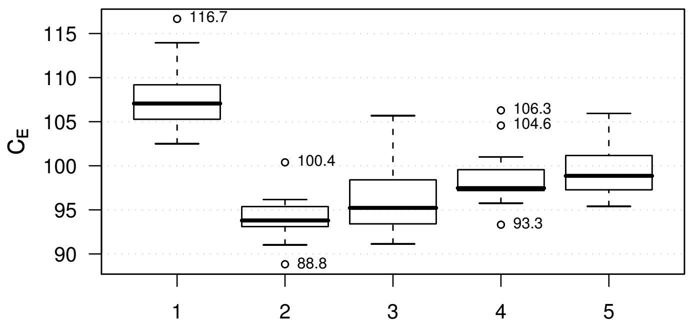


Figure 4.32: Effect of different values of SW on constraint $N_p^{\text{sw}} = SW$ for the Richmond network. (SEA with absolute time-triggers representation, $50\,000$ evaluations).

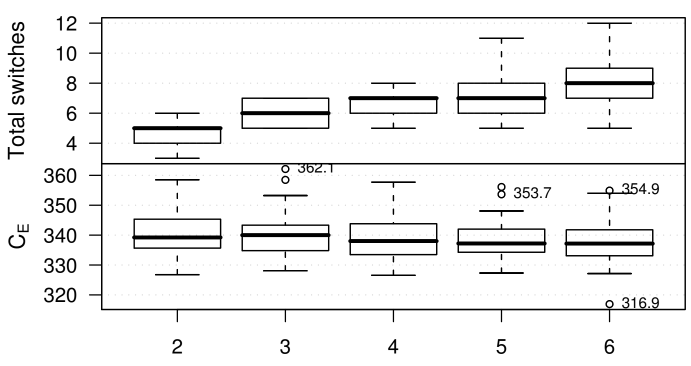


Figure 4.33: Effect of different values of SW in constraint $N_p^{\text{sw}} \leq SW$, in the Vanzyl network (SEA with absolute time-triggers representation, $6\,000$ evaluations).

that allows for less pump switches than the limit ($N_p^{\text{sw}} \leq SW$). On the one hand, as we just discussed above, high number of pump switches do not necessarily mean less electrical cost and, actually, the opposite may be true, since a high limit increases the search space, making harder for the algorithm to find the optimal schedules. On the other hand, if schedules with few pump switches tend to be better, then the algorithm may converge to those, even if the original maximum limit allowed higher number of switches. This is precisely what occurs for the Vanzyl network as shown in Fig. 4.33 (similar results were obtained for the Richmond network). The results are indistinguishable with respect to the electrical cost, even when comparing a limit of two switches per pump with a limit of six switches per pump. Moreover, the total number of pump switches grows slower than the limit. For a limit of two switches per pump, the median number of switches is five, that is, ≈ 1.7 switches per pump, while for a limit of six switches per pump, the median number of switches is eight, that is, ≈ 2.7 switches per pump. We must remember that the algorithm does not have any incentive to reduce the number of pump switches, as its only goal is to minimise the objective function while satisfying the explicit constraints. The constraint on the number of pump switches is always implicitly satisfied by the representation. It can be concluded from these results that the algorithm using a time-triggers representation with a constraint allowing for less switches than the limit is able to adapt the number of pump switches in order to generate lower cost schedules.

4.5 Effect of the Length of Time Intervals

In this section, the effect of using different time intervals is investigated. As described in Chapter 3, explicit representations, such as binary and time-controlled triggers, divide the scheduling period T into several time intervals N^{T}. Normally, the number of time intervals is constant, and, hence, a change in the status of pumps may only occur at fixed times with a minimum interval of T/N^{T}. Up to now, we have considered a scheduling period of 24 hours divided into 24 intervals of one hour. However, in both the time-controlled triggers and binary representation, any length of time intervals could be considered. Finer time intervals would allow more precise scheduling of the pumps and, ideally, may reduce electrical costs. In practice, however, very short time intervals are undesirable, since sudden changes in pump status may cause water hammer and damage the system.

Despite these potential maintenance costs, most studies did not take into account these issues at all. For example, optimisation algorithms using level-controlled triggers assume that the time intervals are only limited by the simulation time step (one second in EPANET) and did not consider any constraints on the minimum time interval between two consecutive pump switches (Atkinson *et al.*, 2000; van Zyl, Savic & Walters, 2004). This often produces small time intervals between consecutive switches and such a schedule may not be practicable.

Therefore, we study the behaviour of SEA with time-controlled triggers representation for different time intervals. In order to obtain a complete picture we consider intervals of 10 minutes, 30 minutes, one hour and two hours.

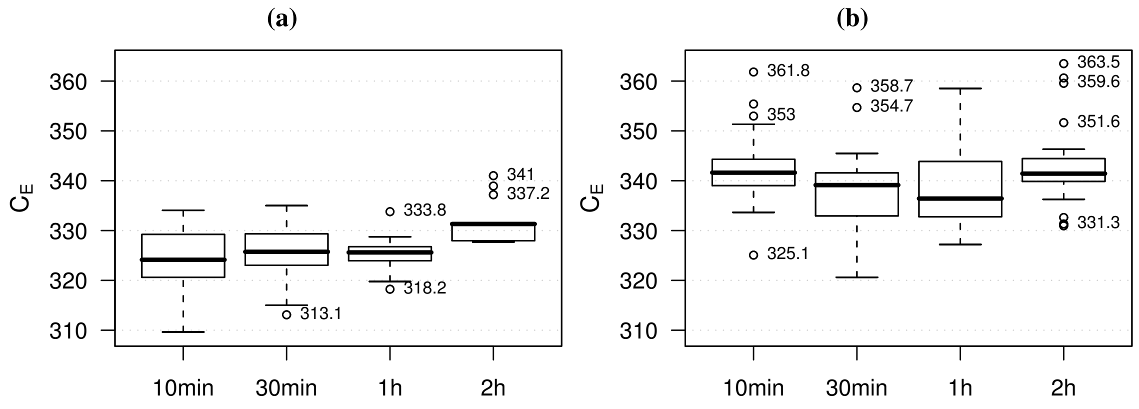


Figure 4.34: Comparison of different time length in the Vanzyl network. SEA with absolute time-triggers representation, 200 000 evaluations (*left*), 6 000 evaluations (*right*).

First, we study the schedules obtained from a long optimisation (200 000 evaluations per run). The results in Fig. 4.34a show a clear degradation of performance when two hour intervals were used, and increase in the variability of results when intervals smaller than one hour were used. In the case of two hour intervals, the near-optimal schedules are simply worse than for shorter time-intervals. For 10 and 30 minutes, the increase of flexibility sometimes does not compensate the increased search space. In other words, there are schedules with lower cost, since the best case is improved, but they are more difficult to find, and hence, the median value is almost identical.

When the number of evaluations is restricted to 6 000, the differences are much smaller, as shown in Fig. 4.34b. In such case, the increased search space introduced by using shorter time intervals does not lead to lower electrical cost and the median electrical cost is very similar, and even worse for 10 minutes time intervals. Our conclusion is that, although shorter time intervals may allow more scheduling flexibility in order to further reduce electrical cost, the growth of the search space is not counterbalanced by a proportional increment in the number of low cost schedules. Hence, the effort required to find near-optimal schedules is greater when using shorter time intervals.

The above results show that both 30 minutes and one hour intervals are best suited for this network. Depending on the real-time variation of demands, 30 minutes intervals may be more appropriate.

4.6 Summary

In this chapter we have empirically studied the new representation, time-controlled triggers, proposed in this study. For this purpose, a Simple Evolutionary Algorithm (SEA) was used, and custom recombination and mutation operators were developed for the proposed time-controlled triggers representations. We tested different settings of SEA, to find adequate combinations of parameters for each representation. A statistical analysis of the experimental results pointed out the parameters and particular settings thereof that had a significant effect on the resulting electrical cost. The analysis indicated that interactions between parameters exist. For example, in absolute time-triggers representation with $\alpha = 200$ and *uniform* mutation produced slightly better results than using *replace* mutation. At the same time, $\alpha = 50$ always improved the results over using $\alpha = 200$. However, for $\alpha = 50$, the best mutation operator was clearly *replace* mutation. Such interactions were not taken into account when fine-tuning similar algorithms in the literature, and it is frequently the case that one parameter is fine-tuned at a time, effectively disregarding the interactions between parameters, what may lead to incorrect conclusions about the effect of each parameter on a particular algorithm.

An improved fine-tuning of the parameters may explain, in part, how the proposed SEA obtains, despite its simplicity, results as good as the best available in the literature when level-controlled trigger representation was used, and clearly better when the binary or time-controlled triggers representation was used. In particular, the new representation based on time-controlled triggers using relative time stands out among the other representations, obtaining the schedules with lowest electrical cost and with a number of pump switches often much lower than the specified limit.

In addition, we studied the effect of the constraint on the number of pump switches. We repeated the experiments of SEA using the binary representation with and without constraint on the number of pump switches. Interestingly, the algorithm without this constraint generated schedules that have relatively low number of pump switches. Our conclusion is that allowing a high number of pump switches with the binary representation does not necessarily lead to lower cost schedules, because the best schedules have lower number of schedules.

Next we investigated the behaviour of the new time-controlled triggers representation with different limits on the number of pump switches and various lengths of the time intervals. The experiments indicated that a strict constraint on the number of pump switches may degrade the quality of the schedules if the limit on the pump switches is too low or too high. It is clear from our experiments that, unless there is a strong reason to prefer an exact number of pump switches per pump, it is always better to use a maximum limit and a relaxed constraint that allows less switches than the limit.

Lastly, we experimentally tested that, although lower cost schedules may be found by using time intervals smaller than one hour, finding these good solutions becomes increasingly difficult because of the growth of the search space. On the other hand, intervals larger than one hour provide poor results. This is consistent with the fact that in both network instances a time step of one hour in the extended period simulation is accurate enough to model the hydraulic behaviour of the network (Atkinson *et al.*, 2000; van Zyl, Savic & Walters, 2004). Therefore, one hour intervals for pump scheduling is a logical choice in this case. However, depending upon the real-time demand variation, a minimum time interval of 30 minutes may also be a possibility.

Chapter 5

Multi-Objective Evolutionary Optimisation for the Pump Scheduling Problem

Most formulations of the pump scheduling problem implicitly define multiple objectives to satisfy. In general, these objectives are minimisation of electrical and maintenance costs. Almost all optimisation models published for the pump scheduling problem aggregate these two objectives into a single objective through the use of penalty functions. Defining appropriate penalty costs is difficult and it requires multiple test runs to find values tailored to a particular network instance. More importantly, a single objective approach prevents system operators from trading maintenance costs for electrical costs according to their expertise and knowledge about the system. By comparison, a multi-objective approach in terms of Pareto-optimality allows system operators to examine a set of candidate pump schedules that models the trade-off between electrical and maintenance costs.

Few multi-objective approaches have been proposed in the literature for the pump scheduling problem. Savic, Walters & Schwab (1997) studied a multi-objective evolutionary algorithm that uses Pareto optimal ranking (Goldberg, 1989), for the minimisation of energy costs and the minimisation of number of pump switches. Constraint violations on tank water levels were incorporated into the electricity cost as penalties and infeasible solutions were considered always dominated by feasible ones. This work did not consider experimental analysis of parameters. Sotelo, von Lücken & Barán (2002) tested various multi-objective evolutionary algorithms (MOEA), namely, SPEA, NSGA, NSGA2, CNSGA, NPGA, and MOGA, to minimise four objectives: (*1*) electricity consumption cost, (*2*) maximum demand charge, (*3*) number of pump switches, and (*4*) difference between the initial and final levels of the tank. Experiments were performed on a network instance with five parallel pumps connected to a tank by a single pipe. The simplicity of the network allowed the use of a mass-balance model instead of hydraulic simulation.

They identified SPEA as the best algorithm overall. The metrics used in this work to assess performance of MOEAs have been later identified as unreliable by Zitzler *et al.* (2003).

Since the above works were published, there have been notable advances in multi-objective algorithms and performance assessment methodology. In the next sections we briefly describe SPEA2 (Zitzler, Laumanns & Thiele, 2002), the multi-objective optimiser used in this chapter, which is an improved version of SPEA. SPEA2 was a state-of-the-art multi-objective algorithm at the time in which this research was conducted. However, at the time of this writing, new state-of-the-art multi-objective algorithms have been proposed (Zhang & Li, 2007; Zhang & Suganthan, 2009). Nonetheless, since we have focused on the general formulation of pump scheduling as a multi-objective problem and its empirical evaluation, the extension of our results to newer multi-objective algorithms should be straightforward. We also give a brief introduction to the performance assessment methods followed in our experimental analysis. These methods are based on state-of-the-art techniques and suggested practices. In fact, some of the tools and techniques have been developed in parallel with the main work of this thesis (Fonseca, Paquete & López-Ibáñez, 2006; López-Ibáñez, Paquete & Stützle, 2009a,b). Next, we perform an experimental analysis of two different alternatives. Firstly, we study the minimisation of both electrical cost and number of pump switches (Section 5.4). In contrast to previous studies, the experimental analysis used in this work compares the results obtained with previous approaches on two different network instances. Secondly, we propose in Section 5.5 an alternative secondary objective, maximisation of shortest idle interval, to replace the minimisation of number of pump switches as a surrogate measure of maintenance costs.

5.1 Multi-objective Optimisation

In multi-objective optimisation (Deb, 2001; Ehrgott, 2000) there is no single objective but a vector of objectives, that is, an *objective vector*. Given two different objective vectors u and v, we say that u *dominates* v if u is not worse than v for each objective value and better for at least one objective. When neither u dominates v nor vice versa, we say that the two objective vectors are *incomparable*. Since each solution represents an objective vector, we use the same terminology among solutions for simplicity. Therefore, given a set of solutions, we can use the dominance relation among their objective vectors to define a subset of solutions which are not dominated by any other solution of that set. This is the *Pareto-optimality principle* and the subset of solutions that are Pareto-optimal are, by definition, mutually not dominated. In this thesis, we denote the subset of Pareto-optimal solutions as the *Pareto set*. The elements of such a Pareto set define implicitly a partition in the objective space between the region dominated by them and the region not dominated by them. Thus, it is often called *Pareto frontier* or *Pareto surface* in the literature. In the context of a multi-objective problem in terms of Pareto-optimality, the goal is to find, or at least approximate to, the *optimal Pareto set*, that is, the set of feasible solutions such that none of its elements is dominated by any other feasible solution.

The dominance relation among solutions of a multi-objective problem can be extended to Pareto sets. Given two different Pareto sets A and B, A is said to be *better* in terms of Pareto-optimality than B iff every solution in B is dominated by or equal to at least one solution in A. If neither A is better than B nor B is better than A, then A and B are *incomparable*. For a formal description of the relations between objective vectors and Pareto sets see Zitzler *et al.* (2003).

5.2 Performance Assessment of Multi-objective Optimisers

The comparison of two optimisation algorithms in terms of quality of their results implies the comparison of their outcomes, either directly or by means of measures that summarise multiple executions of the optimisers. In single-objective problems, the outcome of an algorithm is typically one solution that has single objective value. Single objective values can be totally ordered and summarised by statistical measures such as the mean, the median, the variance, etc. In contrast, in multi-objective problems, the outcome of the algorithm is a Pareto set. Pareto sets can be partially ordered according to Pareto-optimality by means of the relations described in the previous section. Consequently, some Pareto sets can be said to be *better* than others. On the other hand, Pareto sets are often *incomparable* in terms of Pareto-optimality. Hence, the analysis of multi-objective algorithms often requires more advanced techniques. In this thesis, we utilise the hypervolume indicator and the empirical attainment function for this purpose.

5.2.1 The Hypervolume Indicator

Given two incomparable Pareto sets, none of the two is preferable to the other according to Pareto-optimality. Nonetheless, there exist other, more subjective, characteristics of Pareto sets that are generally agreed to be desirable: closeness to the optimal Pareto set in the objective space, and a diverse distribution of objective vectors that exhaustively cover the range of optimal solutions in the objective space. Numerous quality indicators have been proposed in the literature to measure these characteristics. Unary quality indicators assign a real value to each Pareto set, whereas binary indicators assign a real value to pairs of Pareto sets.

Recent research has identified many of these indicators as *Pareto non-compliant*, in the sense that they contradict the Pareto-optimality principle (Knowles, Thiele & Zitzler, 2006). Even among those quality indicators that are *Pareto compliant*, the research has shown they are inherently limited on their descriptive power (Zitzler *et al.*, 2003). One of the least limited unary quality indicators is the hypervolume indicator (HV), introduced by Zitzler & Thiele (1999). The hypervolume indicator calculates the volume of the multi-dimensional region of the objective space enclosed by the polytope defined by the points in a Pareto set P and a reference point that is dominated by all points in P. The largest hypervolume is obtained by the optimal Pareto set and, intuitively, a larger hypervolume would mean a closer approximation to the optimal Pareto set and a wide coverage of the objective space. The hypervolume is the only known unary indicator that correctly identifies that one Pareto set A is not worse than another set B, i.e., $\text{HV}(A) \geq \text{HV}(B)$, for all cases where A is actually better than B in terms of Pareto-optimality (Zitzler *et al.*, 2003). On the other hand, the hypervolume indicator may be $\text{HV}(A) > \text{HV}(B)$ for two sets A and B which are actually incomparable in terms of Pareto-optimality. In this sense, the hypervolume indicator chooses among incomparable sets the one that has the largest volume. Since the volume is defined according to a reference point, this decision is very sensitive to the particular reference point, and, consequently, different reference points may yield different orderings among incomparable Pareto sets. Despite these drawbacks, the hypervolume indicator allows the application of classical statistical measures and techniques to compare and summarise the outcomes of several runs of a multi-objective optimisers. Moreover, recent algorithmic advances have notably reduced its computation time (Beume & Rudolph, 2006; Fonseca, Paquete & López-Ibáñez, 2006). In particular, the algorithm developed in collaboration with Carlos M. Fonseca and Luis Paquete (Fonseca, Paquete & López-Ibáñez, 2006) is used in this chapter to calculate the hypervolume.

5.2.2 Empirical Attainment Function

An alternative to quality indicators is the empirical attainment function (EAF) proposed by Grunert da Fonseca, Fonseca & Hall (2001). The attainment function represents the probability of *weakly-dominating* (dominate or equal) an arbitrary objective vector in the objective space during a single run of an algorithm. The outcome of a stochastic optimiser can be described by its corresponding attainment function. This attainment function is generally unknown but it can be estimated by its empirical attainment function (EAF). The EAF is calculated by means of data collected from several runs of the particular algorithm. Given its relation to other statistical concepts, the EAF can summarise diverse quality aspects of the outcome of an algorithm. The EAF does not address the dependence structure within each outcome, and thus, it does not show the frequency of two objective vectors being attained in the same run (Fonseca, Grunert da Fonseca & Paquete, 2005). Nonetheless, in this work we are not interested on the dependency structure of the outcomes. Hence, the location information is typically sufficient to assess the quality of the expected outcome of the algorithms in our case.

The attainment function is a generalisation of the multivariate cumulative distribution function (CDF). The inverse of the CDF is the quantile function of a probability distribution, which for a particular probability returns the minimum value that is higher with a given probability than random values of the distribution. For example, given a probability of 50% the quantile function returns the median of a distribution. Given probabilities of 25% and 75%, the quantile function would return the first and third quartiles respectively, which are both used in the calculation of boxplots and the inter-quartile range (IQR). Similarly, by calculating the inverse of the EAF for a given probability $k\%$, we can describe the likely location of the outcome of an algorithm by its $k\%$-attainment surface. For example, 50%-attainment surface contains objective vectors that are weakly-dominated by 50% of the runs of the algorithm. Hence, it is called the *median attainment surface*. The 25%- and 75% attainment surfaces are, in a similar way, equivalent to the first-quartile and third-quartile. Moreover, we can define the *best attainment surface* as those objective vectors that are weakly-dominated by only one of the runs carried out, whereas objective vectors in the *worst attainment surface* were weakly-dominated by all runs carried out.

Finally, two algorithms can be compared by calculating the difference in the value of the EAFs at each point in the objective space. The difference can only be calculated for those points where a value of the EAF is available, though. By plotting the differences in favour of each of the two algorithms, we can quickly identify the regions of the objective space where each algorithm outperforms the other. This technique was first proposed by López-Ibáñez, Paquete & Stützle (2006) to analyse the quality of several multi-objective algorithms for the bi-objective quadratic assignment problem. Recently, López-Ibáñez, Paquete & Stützle (2009a,b) have made publicly available tools for generating such plots.

5.3 SPEA2

The optimiser used is the second version of the Strength Pareto Evolutionary Algorithm (SPEA2) (Zitzler, Laumanns & Thiele, 2002). The main features of SPEA2 are: (*i*) the fitness of a solution depends on the strength of the solutions by which it is dominated, where the strength of a solution is defined as the number of other solutions in the current population that it dominates; (*ii*) the ties of solutions with the same fitness are broken by a nearest neighbour density estimation technique; (*iii*) the size of the archive of nondominated solutions is a fixed value α. When the actual number of nondominated solutions is lower than α, the archive is filled with dominated solutions. When the actual number of nondominated solutions exceeds α, some of them are discarded by a truncation operator which preserves boundary solutions. Because the values of different objectives are not comparable, the distances between solutions with respect to a given objective are normalised. More details on SPEA2 can be found in the original publication (Zitzler, Laumanns & Thiele, 2002).

Our implementation of SPEA2 is based on original `C` source code[1] from the PISA project (Bleuler *et al.*, 2003) but with significant modifications to serve our purposes. The algorithm schema of SPEA2 as implemented here can be summarised as follows. The archive is initialised with α randomly generated solutions. At each iteration, μ number of solutions are selected from the archive as parents using a binary tournament selection. Recombination is applied to parents in order to generate μ offspring solutions. Then, the mutation operator is applied to the offspring solutions with a certain probability. These new solutions are evaluated to calculate their fitness and all nondominated solutions are added to the archive. If the size of the archive becomes larger than α, the solution which has the minimum distance to another solution (according to the truncation operator) is discarded until archive size is exactly α. In case of the number of nondominated solutions is less than α, the dominated solution with the minimum fitness value is added to the archive until there are α solutions in the archive. A new iteration would start by selecting again μ solutions from the archive.

The above implementation of SPEA2 is deliberately similar to the algorithm schema of SEA described in Section 4.1. In fact, the only difference is the way new solutions are merged into the population (archive) and the fitness calculation. Furthermore, exactly the same recombination and mutation operators proposed for SEA (Section 4.1.2) are used in SPEA2. Therefore, differences in the quality of results obtained by SEA and SPEA2 can only be attributed to the multi-objective nature of the latter.

5.3.1 Constraint Handling

We use for SPEA2 a method for handling constraints similar to the one used in SEA (Section 4.1.1) by extending the dominance criteria with the following rules:

1. Any solution is dominated by another solution with a lower pressure deficit (ΔH in Eq. 2.6).

[1] Source code available at `http://www.tik.ee.ethz.ch/sop/pisa/`

2. If pressure violations are equal, the solution with the lower number of warnings from the simulator dominates the other.

3. For equal number of warnings and pressure deficit, the solution with the lower total volume deficit (ΔV in Eq. 2.4) dominates the other.

4. Given two solutions with equal pressure deficit, number of warnings and total volume deficit, the normal dominance criteria between their objective values is applied.

The rationale for these rules and their relative order is discussed in detail in Section 4.1.1 on page 41.

5.4 Minimisation of Number of Pump Switches

The number of pump switches is frequently used as a surrogate measure of maintenance costs. Therefore, minimising the number of pump switches will result in minimisation of maintenance costs. So far we have limited the number of switches per pump, either using an explicit constraint for the binary and level-controlled triggers representations, or implicitly through the proposed time-controlled triggers representation. In Section 4.4 we studied the effect of this constraint on SEA.

Limiting maintenance costs may not be enough in all scenarios. System operators may be interested in pump schedules with minimum maintenance cost. The usual technique to achieve this goal is to assign a penalty cost to each additional pump switch. However, finding appropriate penalty costs may be complex: after all, estimation of maintenance costs is difficult. Moreover, a single pump switch penalty cost cannot account for excessive switching of a particular pump (versus the same number of switches performed by several pumps), or the influence of other constraints. Once penalty values are fixed, the optimisation algorithm offers the system operator one pump schedule that is optimal (or an approximation of the optimal) with respect to those penalty costs. If the system operator wishes to obtain a slightly cheaper solution, perhaps allowing a few more pump switches, a new run of the optimisation algorithm must be performed using a different set of penalty values. Even for a medium-sized real-world network a single optimisation run may require several hours of computation time, and thus, such a trial-and-error procedure would be prohibitively long.

A multi-objective approach that minimises both the electrical cost (C_{E}) and the number of pump switches (N^{sw}) would generate a Pareto set of feasible pump schedules. This Pareto set models the trade-off between electrical and maintenance costs and it can be examined by system operators in order to choose one particular schedule from the set. In the following sections we empirically investigate this approach by means of SPEA2.

5.4.1 Experimental Setup

The same recombination and mutation operators tested on the single-objective SEA in Chapter 4 are used in SPEA2. The multi-objective approach is tested on the four representations described in Chapter 3: binary, level-controlled triggers, and absolute and relative time-controlled triggers. One may argue that one of the characteristics of the time-controlled triggers representations, namely, the implicit constraint on pump switches, is not useful in the multi-objective approach proposed in this section. On the contrary, the implicit constraint is still useful to define a maximum number of switches per pump (N_p^{sw}) and to limit the size of the search space. Whether limiting the search space helps SPEA2 to find better schedules would be investigated by comparing the results obtained using the best settings for each representation.

Experiments were performed on both Vanzyl and Richmond network. For each combination of parameters, runs of SPEA2 were repeated with different random seeds: 25 repetitions in the case of the Vanzyl network and 15 repetitions for the Richmond network. After some initial experiments, these number of repetitions were deemed sufficiently large to achieve a stable variability. In fact, the variability of the results was higher in the case of Vanzyl network. Each run was stopped after a number of invocations of the hydraulic simulator EPANET to evaluate the objectives and constraints of a candidate solution. The maximum number of evaluations was set to 6000 for the Vanzyl network and 8000 for the Richmond network following the setup proposed by van Zyl, Savic & Walters (2004) and already used for SEA (Section 4.3). This setup allows to compare the results of SPEA2 with SEA and Hybrid GA (van Zyl, Savic & Walters, 2004), with the caveat that Hybrid GA was designed specifically for level-controlled triggers, whereas the other algorithms are fined-tuned for each of the four representations studied in this thesis.

The results were analysed by means of the hypervolume indicator (HV) and the empirical attainment function (EAF). First, we bounded the results to $C_{\text{E}} \leq 490$ and $N^{\text{sw}} \leq 18$ for the Vanzyl network, and $C_{\text{E}} \leq 290$ and $N^{\text{sw}} \leq 49$ for the Richmond network. This is done to prevent the results being distorted by very extreme values. Second, for the calculation of the hypervolume, all objective values were normalised to $[1, 2]$ and the reference point was set to $(2, 4)$. We choose the interval $[1, 2]$ instead of the more common $[0, 1]$ to avoid potential problems with division by zero or logarithmic transformations. The choice of this reference point assigns a higher hypervolume to Pareto sets that further minimise the electrical cost. We perform a statistical analysis of the hypervolume values similar to the one used to analyse the results of SEA. In addition, plots of the attainment surfaces and the differences between EAFs are used to obtain further insights into the results.

5.4.2 Binary Representation

The experimental setup of SPEA2 using the binary representation is similar to the one used for SEA in Section 4.2.1. We test all possible combinations of $\alpha = \{50, 100, 200\}$, $\mu = \{5, 20\}$, recombination and mutation operators, where recombination can be either *one-point*, *two-point*, or *uniform*, and mutation is either *flip* or no mutation.

The Vanzyl Network

ANOVA identifies all parameters as having a significant effect on the resulting hypervolume (HV) with varying degrees of significance. We focus on the two interactions with the highest significance level, namely the interactions between mutation and population size (α), and between mutation and recombination.

Mutation and population size (α). The most significant interaction (Fig. 5.1a) shows that, in general, *flip* mutation obtains better results than no mutation at all. With mutation, $\alpha = 200$ produces worse hypervolume values than other values of α. In contrast, without mutation, increasing the value of α improves considerably the quality of the results.

Mutation and recombination operators. The particular recombination operator is very important when not using mutation, being *uniform* recombination superior to the others, as shown in Fig. 5.1b. On the other hand, with *flip* mutation, the different recombination operators do not produce a significantly distinct effect on the resulting hypervolume.

Taking into account the above analysis, the best combinations would be *flip* mutation and $\alpha = \{50, 100\}$. Given these settings, the particular recombination operator and offspring population size (μ) do not make a clear difference. Among this group of similar

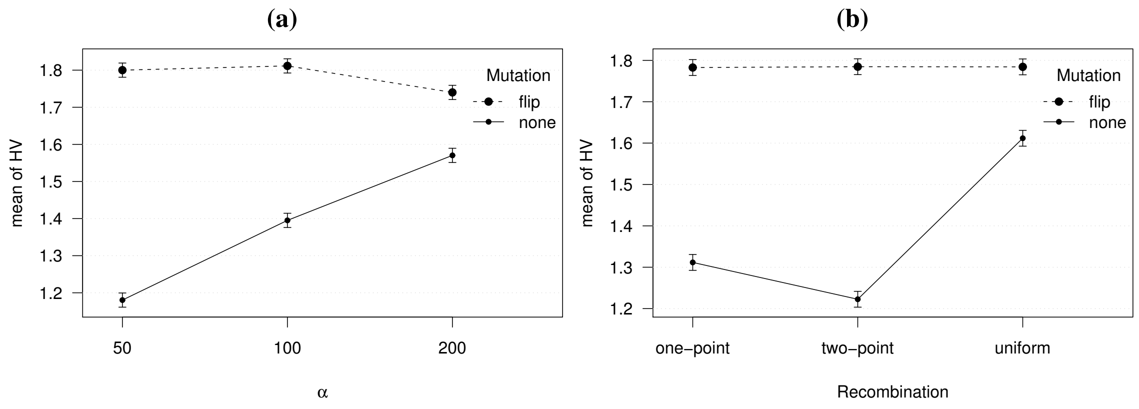


Figure 5.1: Interaction plots of hypervolume value obtained by SPEA2 using binary representation (the Vanzyl network).

configurations of parameters, we choose $\alpha = 50$, *two-point* recombination and $\mu = 5$, which obtained the highest median hypervolume.

We further compare the recombination operators by plotting the best, median, and worst attainment surfaces for the configurations with $\alpha = 50$, $\mu = 5$, *flip* mutation and recombination: *one-point* (Fig. 5.2a), *two-point* (Fig. 5.2b) and *uniform* (Fig. 5.2c). Although the plots are very similar, *two-point* recombination obtained slightly better best and median attainment surfaces than the rest. In the median case in particular, *two-point* recombination obtains schedules with electrical cost close to 330.

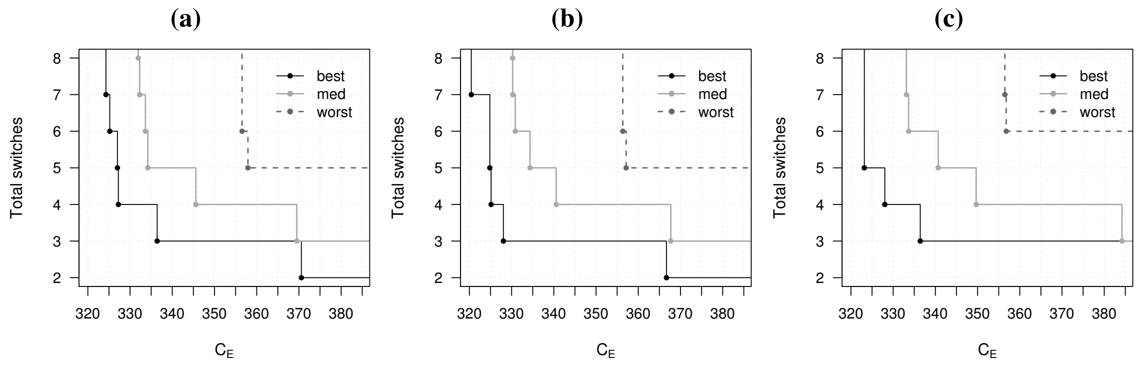


Figure 5.2: Attainment surfaces of SPEA2 using the binary representation and (a) *one-point*, (b) *two-point* and (c) *uniform* recombination (the Vanzyl network).

It is interesting to point out the similarities between the interaction plots shown in Fig. 5.1 and those corresponding to the single-objective SEA (Fig. 4.6 on page 53). Bearing in mind that the y-axis is inverted because hypervolume is maximised, whereas electrical cost is minimised, the plots mirror each other. This fact suggests that best settings for parameters are strongly related to the particular representation and may be similar for both SEA and SPEA2. In particular, we concluded in Chapter 4 that mutation is essential to obtain good schedules from SEA independent of the representation used. This is also true in the case of SPEA2. In order to simplify the experimental analysis of SPEA2, in the remainder of this chapter we will focus on the results obtained using mutation.

The Richmond Network

In the case of the Richmond network, we simplify the experimental setup and we only consider configurations with *flip* mutation and $\mu = 20$, $\alpha = \{50, 100, 200\}$ and *one-point*, *two-point*, and *uniform* recombination.

ANOVA identifies only α as a significant factor. We calculate 95% confidence intervals around the difference in means between the values of α in Table 5.1. The intervals show that there are significant differences among all values. The best setting is $\alpha = 50$, while the highest differences are against $\alpha = 200$.

Table 5.1: 95% confidence intervals around the difference in mean hypervolume (HV) obtained by SPEA2 with various values of α (binary representation, Richmond network).

α	95% CI (HV)
$50 - 100$	$[0.018, 0.085]$
$50 - 200$	$[0.126, 0.194]$
$100 - 200$	$[0.075, 0.142]$

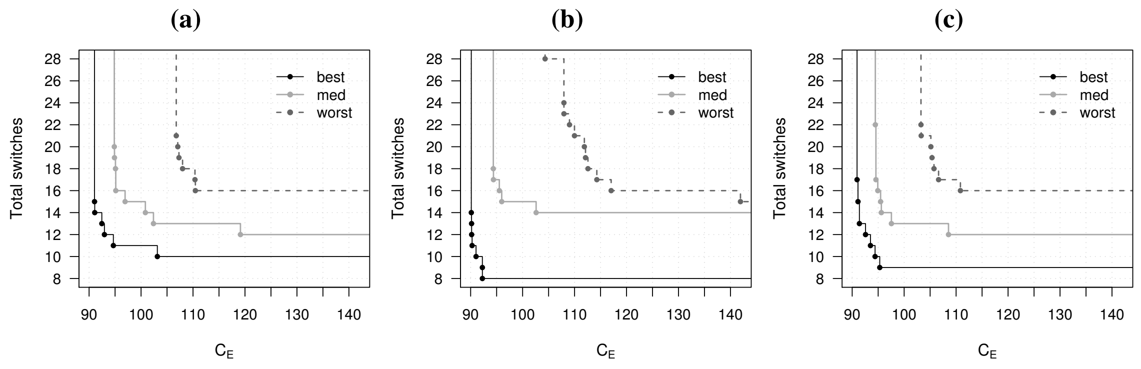


Figure 5.3: Attainment surfaces of SPEA2 using the binary representation and (a) *one-point*, (b) *two-point* and (c) *uniform* recombination (the Richmond network).

We plot the best, median and worst attainment surfaces corresponding to $\alpha = 50$, $\mu = 20$, *flip* mutation and recombination *one-point* (Fig. 5.3a), *two-point* (Fig. 5.3b) and *uniform* (Fig. 5.3c). In this case, despite *two-point* recombination obtains the best best-case, we choose *uniform* recombination since it obtains a similar, if not better, median attainment surface and much better worst-case.

Interestingly, most of the schedules shown in Fig. 5.3 have a number of pump switches lower than 21, i.e., on average there are less than three switches per pump. Since we are not using any constraint to enforce such number of pump switches, this is a confirmation of our observations in Section 4.4, where we noticed that allowing a higher number of pump switches over a certain limit may not lead to lower cost schedules.

5.4.3 Level-controlled triggers

For the representation based on level-controlled triggers, we test the following setup: $\alpha = \{50, 100, 200\}$, $\mu = 20$, *extended-intermediate* recombination and either *gaussian* or *replace* mutation.

The Vanzyl Network

ANOVA indicates that no particular factor, or interaction thereof, has a significant effect on the hypervolume values obtained by SPEA2. This is confirmed by the boxplot of hypervolume values obtained by each combination of parameters (Fig. 5.4).

We plot the attainment surfaces of each combination of parameters in Fig. 5.5. The plots are in principle very similar with the exception that *replace* mutation obtains a slightly better median and worst-case than *gaussian* mutation. The plot of differences between

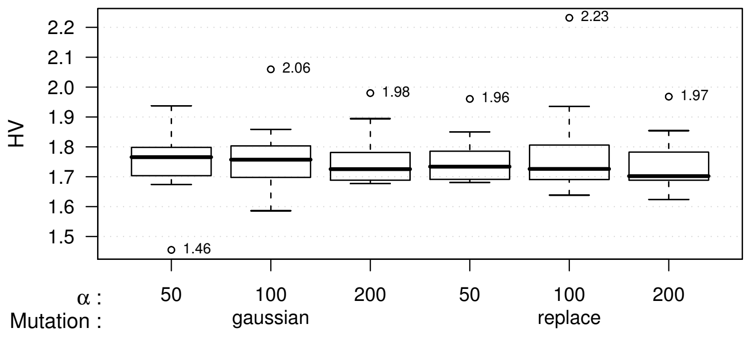


Figure 5.4: Boxplot of Hypervolume values obtained by SPEA2 using level-controlled triggers representation (the Vanzyl network).

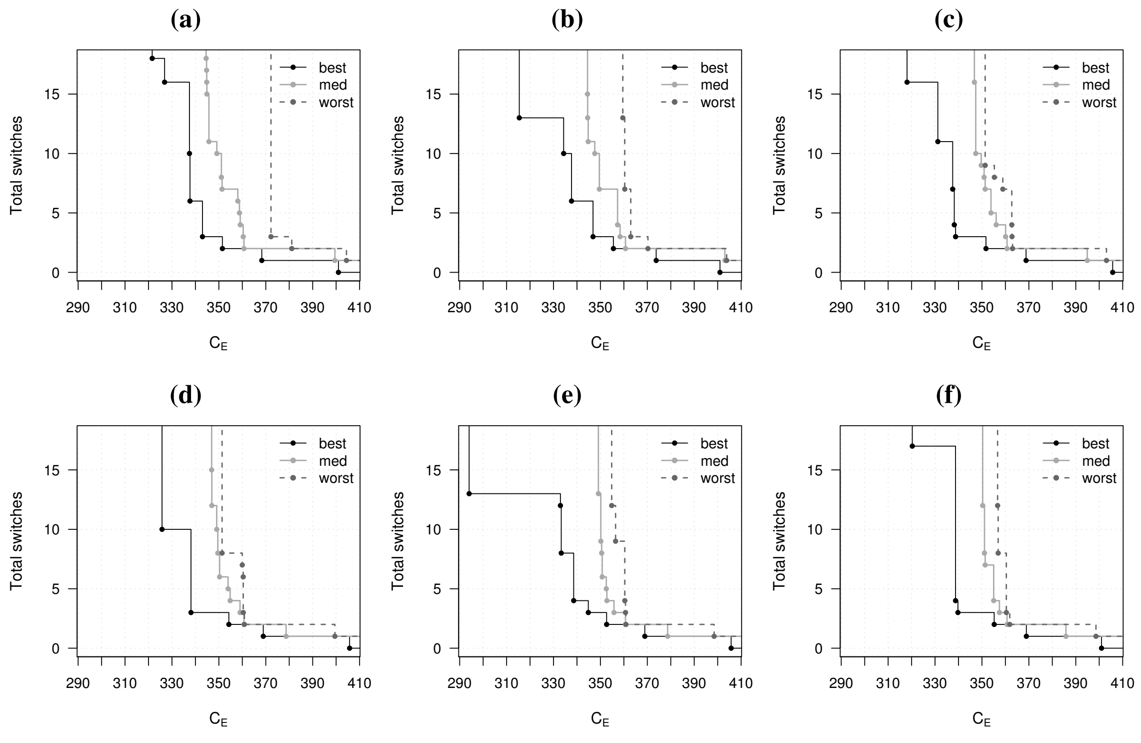


Figure 5.5: Attainment surfaces of SPEA2 using level-controlled triggers representation. *Gaussian* mutation with α equal to (a) 50, (b) 100, and (c) 200; and *replace* mutation with α equal to (d) 50, (e) 100, and (f) 200 (the Vanzyl network).

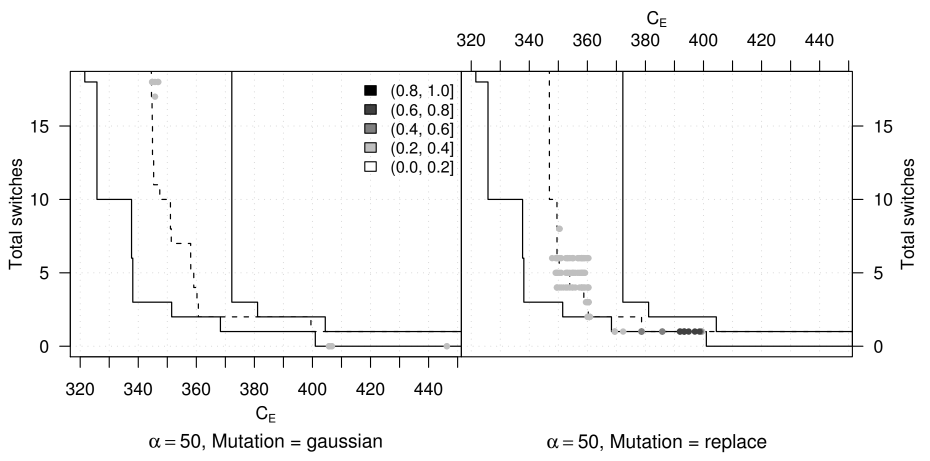


Figure 5.6: Differences of the EAF of SPEA2 using level-controlled triggers representation and $\alpha = 50$ between *gaussian* and *replace* mutation (the Vanzyl network).

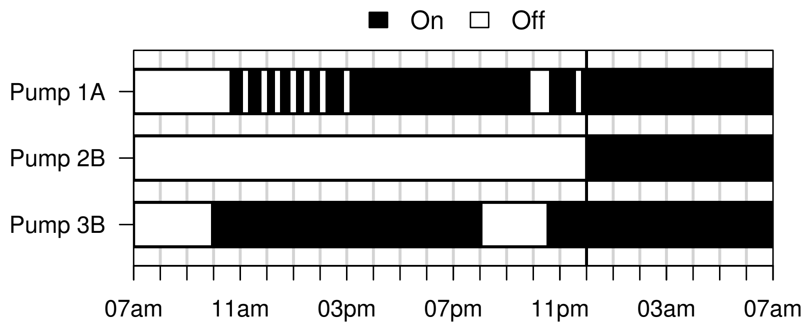


Figure 5.7: Best schedule found by SPEA2 using level-controlled triggers representation (the Vanzyl network).

the EAFs of *gaussian* and *replace* mutation for $\alpha = 50$ shows a small difference in favour of *replace* mutation (Fig. 5.6). Although there is no best combination of parameters, for the sake of comparison, we choose $\alpha = 50$ and *replace* mutation.

Figure 5.5e shows that SPEA2 has found a pump schedule with $C_{\text{E}} = 294.03$ and $N^{\text{sw}} = 13$, which is the best feasible schedule we found for the Vanzyl network in the course of this study. However, the practicality of this schedule, which is shown graphically in Fig. 5.7, is questionable. The rapid switching on/off of pump 1A is probably not practicable, though it is technically feasible. None of the constraints or secondary objectives we have discussed so far are able to prevent this issue. The same problem does not appear with the binary or time-controlled triggers representation, since pump switches are separated by a minimum time interval. In Section 5.5 we will discuss the maximisation of idle intervals as a secondary objective in order to prevent such fast switching of pumps.

The Richmond Network

In the case of the Richmond network, the requirements of ANOVA are not met. Instead of ANOVA, we examine directly the distribution of hypervolume values obtained by the different combinations of parameters in Fig. 5.8. The plot clearly indicates that *replace* mutation outperforms *gaussian* mutation in terms of hypervolume indicator. To confirm this hypothesis, we perform pairwise comparisons by means of Wilcoxon rank-sum tests. The stronger differences (p-values lower than 0.001) are between the configurations of SPEA2 using *replace* mutation and the other configurations, which confirms the initial hypothesis. On the other hand, no significant differences are found for the different values of α when using *replace* mutation. Nevertheless, we prefer $\alpha = 100$, which according to Fig. 5.8, produces a lower variability and a higher median hypervolume.

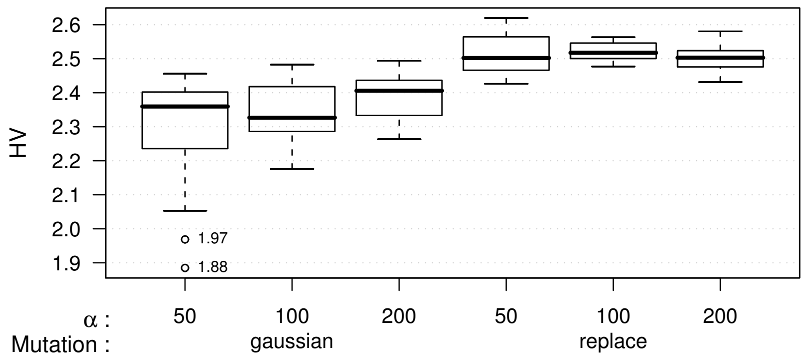


Figure 5.8: Boxplot of Hypervolume values of SPEA2 using level-controlled triggers representation (the Richmond network).

We confirm these conclusions by examining the corresponding EAFs. In Fig. 5.9, the comparison of the EAFs corresponding to *gaussian* and *replace* mutation (both with $\alpha = 200$) shows differences in favour of the latter but not the former. Differences are particularly strong in the region with a number of pump switches between 5 and 20, which is probably the most interesting region for network operators. Focusing on *replace* mutation, the plots of the attainment surfaces for each value of $\alpha = \{50, 100, 200\}$ are shown in Fig. 5.10, where it can be observed in the median and worst attainment surfaces that using $\alpha = 100$ has a small advantage over other options. Therefore, we choose $\alpha = 100$ and *replace* mutation as the best settings of SPEA2 with level-controlled triggers for the Richmond network.

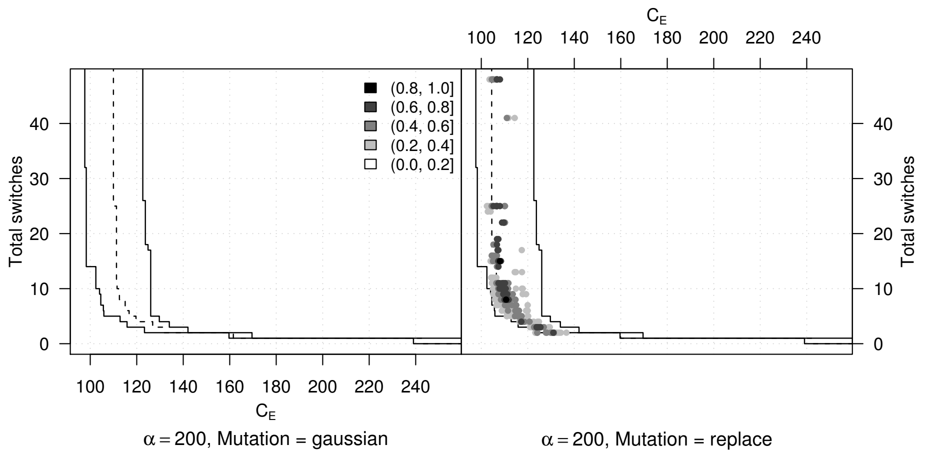


Figure 5.9: Differences of the EAF of SPEA2 using level-controlled triggers representation with $\alpha = 200$ between *gaussian* and *replace* mutation (the Richmond network).

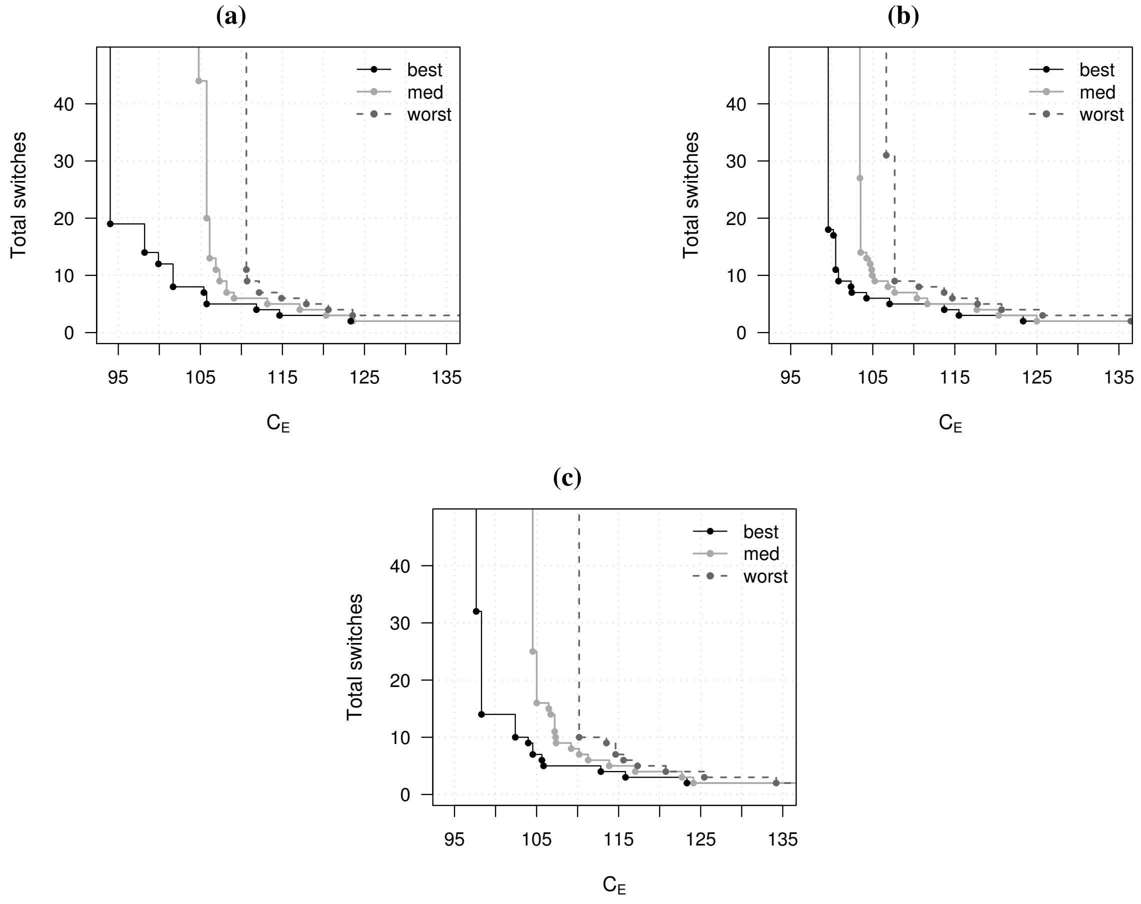


Figure 5.10: Attainment surfaces of SPEA2 using level-controlled triggers representation and *replace* mutation with (a) $\alpha = 50$, (b) $\alpha = 100$, and (c) $\alpha = 200$ (the Richmond network).

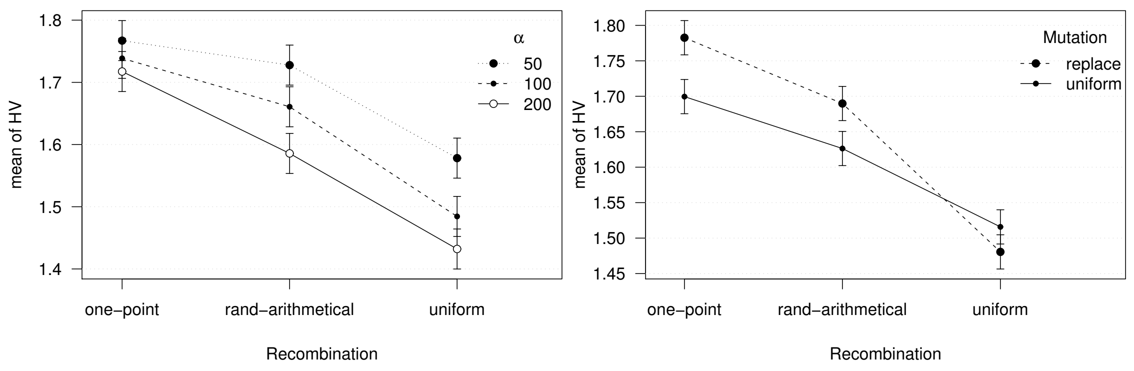


Figure 5.11: Interaction plots of SPEA2 using absolute time-controlled triggers (the Vanzyl network).

5.4.4 Absolute Time-controlled Triggers

In the case of the representation based on absolute time-controlled triggers, the experimental setup of SPEA2 is $\alpha = \{50, 100, 200\}$, *one-point*, *rand-arithmetical* or *uniform* recombination, and either *replace* or *uniform* mutation. The value of μ was set to 20 for the Vanzyl network and 5 for the Richmond network.

The Vanzyl Network

Interactions between α and recombination and between mutation and recombination are identified as significant by ANOVA. The interaction plots in Fig. 5.11 show that *one-point* recombination obtains the best hypervolume value overall, in particular when combined with *replace* mutation. Moreover, the different values of α only have a significant effect for the other recombination operators.

We study next the best, median and worst attainment surfaces obtained by SPEA2 using *one-point* recombination and *replace* mutation for various values of α (Fig. 5.12). We observe that the median attainment surface of $\alpha = 50$ practically dominates the median case obtained by the other values of α. Moreover, SPEA2 with $\alpha = 50$ obtains schedules with an electricity cost close to 340 and a number of pump switches of $N^{\text{sw}} = 5$. Therefore, we choose this configuration of parameters (*one-point* recombination, *replace* mutation and $\alpha = 50$) as the best.

The Richmond Network

The same experimental setup is tested for the Richmond network, except that $\mu = 5$. In this case the requirements of ANOVA are not clearly satisfied. Instead, we investigate the distribution of hypervolume for each parameter. In the case of recombination, Fig. 5.13 shows that *one-point* recombination is clearly superior to both *rand-arithmetical* and *uni-*

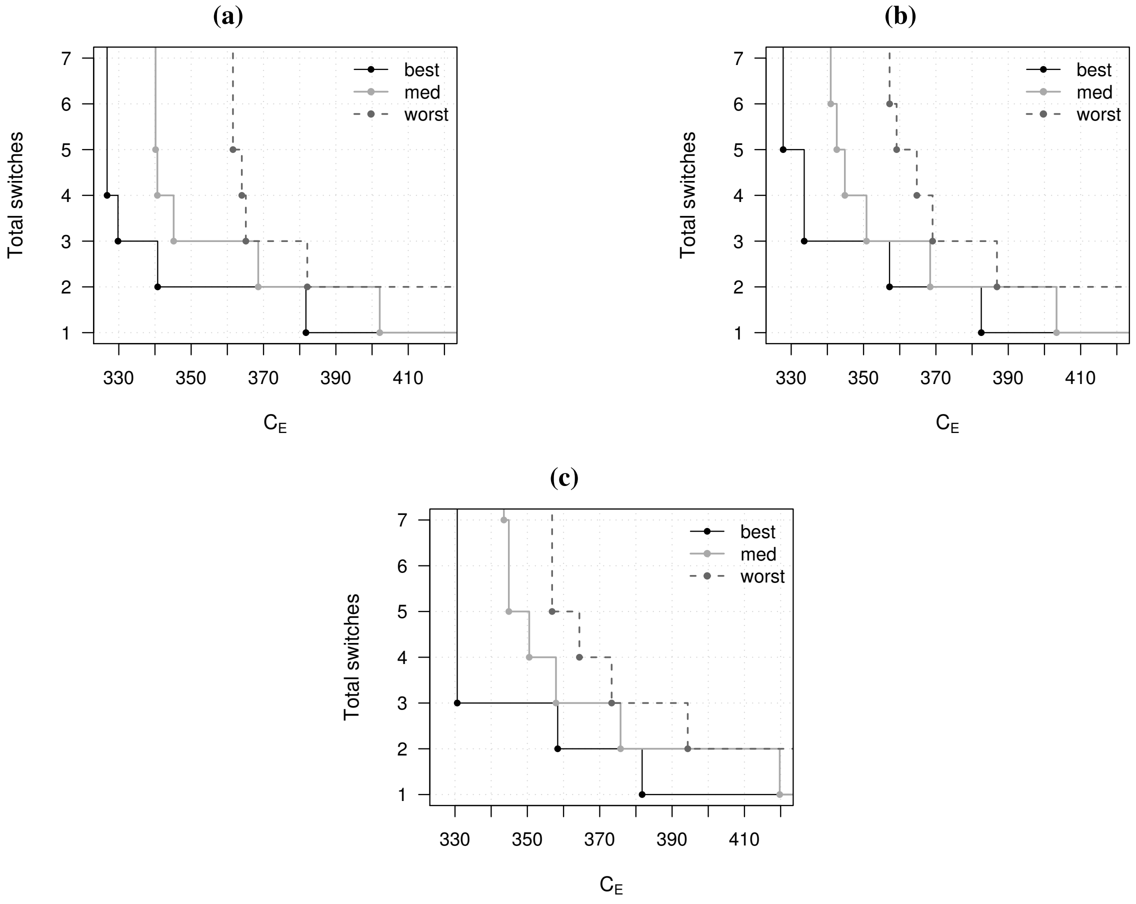


Figure 5.12: Attainment surfaces of SPEA2 using absolute time-controlled triggers, *one-point* recombination and (a) $\alpha = 50$, (b) $\alpha = 100$ and (c) $\alpha = 200$ (the Vanzyl network).

form recombination. Focusing on the results obtained by *one-point* recombination, we plot in Fig 5.14 the distribution of hypervolume values for the other parameter settings. The first observation is that low values of α produce better results when combined with *replace* mutation, whereas the effect of α is not evident when combined with *uniform* mutation. In terms of hypervolume indicator, *one-point* recombination, *replace* mutation and $\alpha = 50$ are the best settings.

We confirm these conclusions by examining in more detail the attainment surfaces of four combinations of parameters, namely, $\alpha = 50$ and *replace* mutation (Fig. 5.15a), $\alpha = 100$ and *replace* mutation (Fig. 5.15b), $\alpha = 50$ and *uniform* mutation, (Fig. 5.15c), and $\alpha = 100$ and *uniform* mutation, (Fig. 5.15d). The plots show that *replace* mutation obtains the lower cost schedules for the median and worst-case scenarios. Moreover, the combination of *replace* mutation and $\alpha = 50$ is able to obtain schedules with an electricity cost lower than the combination of *replace* mutation and $\alpha = 100$ for the median case, while keeping a low number of pump switches. This confirms the results obtained by analysing the hypervolume value, and thus, we conclude that $\alpha = 50$, *one-point* recom-

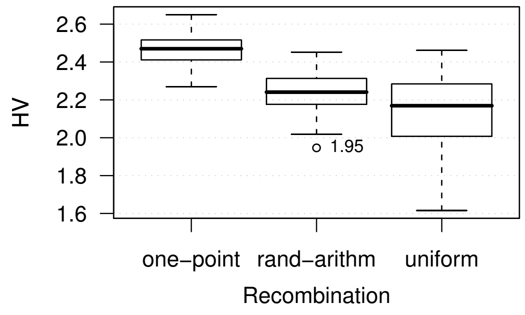


Figure 5.13: Boxplot of Hypervolume values of SPEA2 using absolute time-controlled triggers (the Richmond network).

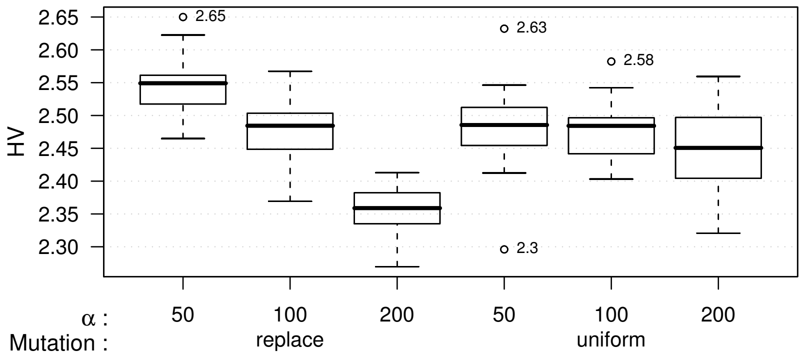


Figure 5.14: Boxplot of Hypervolume values of SPEA2 using absolute time-controlled triggers and *one-point* recombination (the Richmond network).

bination and *replace* mutation are the best settings of SPEA2 for the Richmond network when using absolute time-controlled triggers.

5.4.5 Relative Time-controlled Triggers

In this case, the experimental setup for both Vanzyl and Richmond networks, includes all combinations of $\alpha = \{50, 100, 200\}$, $\mu = 5$, *one-point*, *rand-arithmetical* and *uniform* recombination, and *replace* and *uniform* mutation.

The Vanzyl Network

ANOVA identifies all parameters as having a significant effect on the output of SPEA2. In particular, the interactions between α and mutation, and between mutation and recombination are statistically significant.

Mutation and population size (α)**.** Figure 5.16a reveals that for *replace* mutation, the value of α does not have a significant effect, whereas the opposite is true for *uniform* mutation. Only in the case of $\alpha = 200$, there are significant differences in the results obtained using different mutation operators.

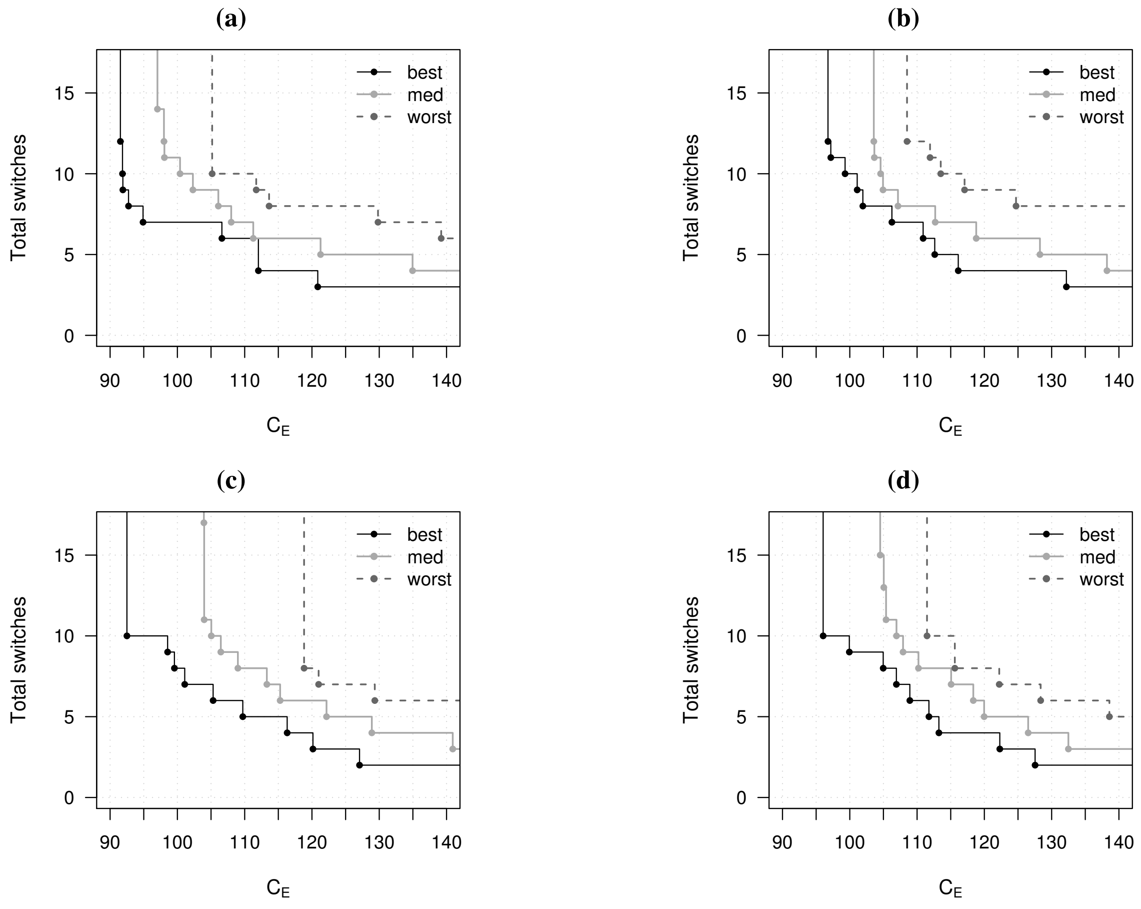


Figure 5.15: Attainment surfaces of SPEA2 using absolute time-controlled triggers, *one-point* recombination and: (a) $\alpha = 50$ and (b) $\alpha = 100$ and *replace* mutation, and (c) $\alpha = 50$ and (d) $\alpha = 100$ and *uniform* mutation (the Richmond network).

Recombination and mutation operators. As illustrated by Fig. 5.16b, *rand-arithmetical* recombination obtains the highest hypervolume values independent of the mutation operator. Moreover, the mean values suggest that *uniform* mutation works better in combination with *rand-arithmetical* recombination. The other recombination operators obtain the best mean hypervolume when combined with *replace* mutation. When the Tukey HSD intervals overlap the differences between the means cannot be considered significant, which is the case for *rand-arithmetical* recombination. Thus, no mutation operator can be said to maximise the mean hypervolume.

According to the above analysis, the best settings would be *rand-arithmetical* recombination, $\alpha = \{50, 100\}$ and either *replace* or *uniform* mutation. We examine in more detail these configurations by means of the attainment surfaces in Fig. 5.17. Each mutation operator in combination with $\alpha = 50$ obtains a slightly better median attainment surface than $\alpha = 100$. Comparing *replace* and *uniform* mutation, the latter produces better results for high number of pump switches, whereas *replace* obtains better results for low number

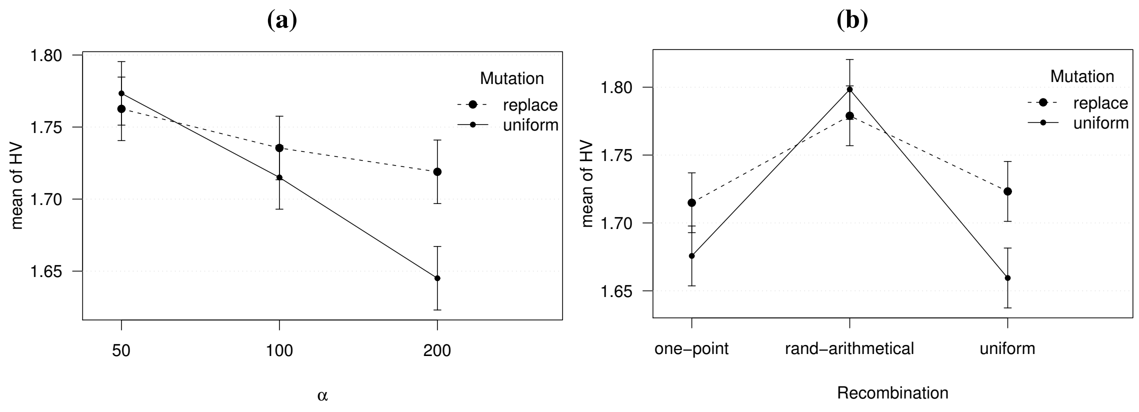


Figure 5.16: Interaction plots of SPEA2 using relative time-controlled triggers (the Vanzyl network).

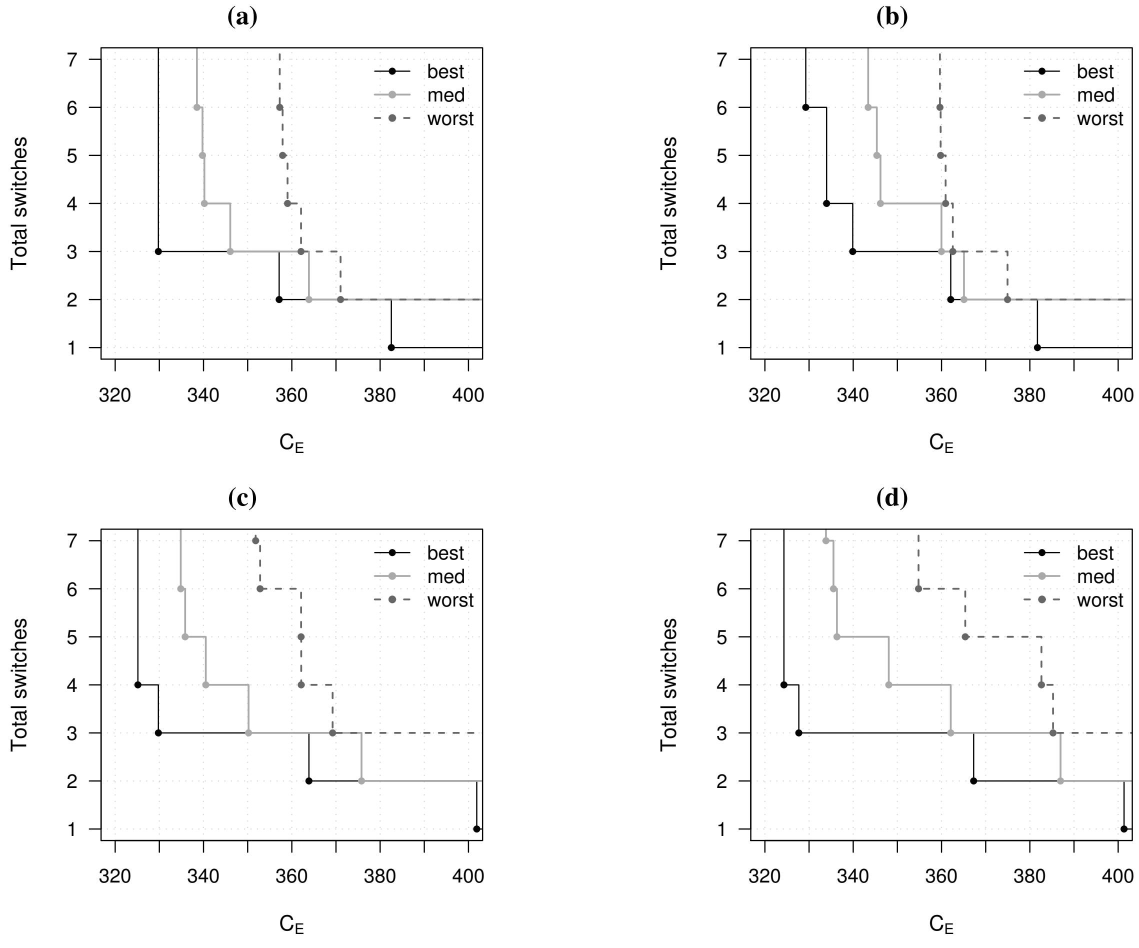


Figure 5.17: Attainment surfaces of SPEA2 using relative time-controlled triggers, *rand-arithmetical* recombination and (a) $\alpha = 50$ and *replace* mutation, (b) $\alpha = 100$ and *replace* mutation, (c) $\alpha = 50$ and *uniform* mutation, and (d) $\alpha = 100$ and *uniform* mutation (the Vanzyl network).

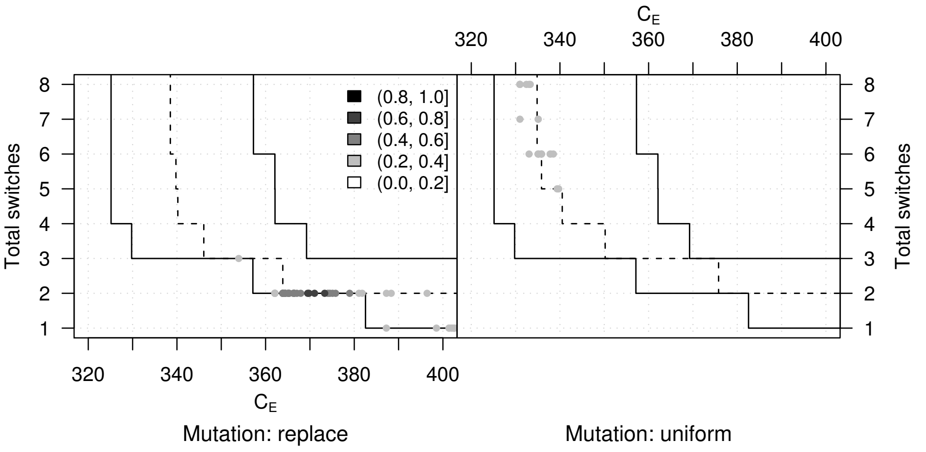


Figure 5.18: Differences of the EAFs of SPEA2 using relative time-controlled triggers representation, $\alpha = 50$ and *rand-arithmetical* recombination (the Vanzyl network).

of pump switches. This is confirmed by examining the differences between the EAFs obtained using *replace* mutation versus *uniform* mutation (both with $\alpha = 50$) in Fig. 5.18. The plot shows that *replace* mutation obtains a lower cost than *uniform* mutation only for schedules with less than three switches. For schedules with a number of switches between six and eight, *uniform* mutation obtains slightly lower electrical cost than *replace* mutation. We prefer in this case the lower electricity cost provided by *uniform* mutation. In summary, for the Vanzyl network, the best settings of SPEA2 with relative time-controlled triggers are $\alpha = 50$, *rand-arithmetical* recombination and *uniform* mutation.

The Richmond Network

In the case of Richmond network, ANOVA indicates that all parameters have a significant effect on the hypervolume value obtained by SPEA2. The difference in hypervolume value obtained by *replace* versus *uniform* mutation is small, within a 95% confidence interval of $[0.019, 0.056]$ in favour of the former. The interaction between recombination and α is also deemed significant by ANOVA. The corresponding interaction plot (Fig. 5.19) reveals that the best overall value for α is 50 and the best recombination operator is *rand-arithmetical*. The Tukey's HSD confidence intervals of $\alpha = 50$ and $\alpha = 100$ overlap slightly when *rand-arithmetical* recombination was used. Similarly, the interval corresponding to $\alpha = 100$ and *rand-arithmetical* overlaps with the intervals for the other recombination operators and $\alpha = 50$. Therefore, it cannot be concluded that there is a statistically significant difference between those configurations of parameters.

The examination of the attainment surfaces sheds some light over the best choice of settings. We plot in Fig. 5.20 the best, median and worst attainment surfaces corresponding to *rand-arithmetical* with either $\alpha = 50$ or $\alpha = 100$, and either *replace* or *uniform*

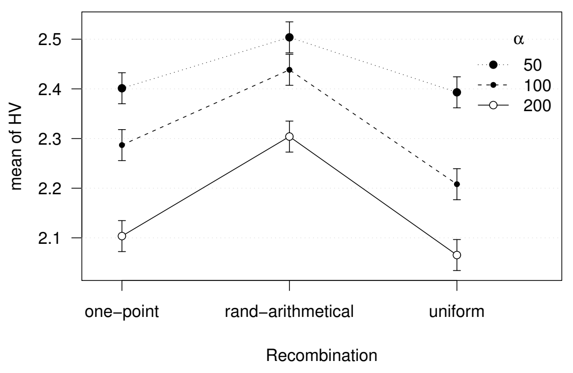


Figure 5.19: Interaction plot of hypervolume value obtained by SPEA2 using relative time-controlled triggers (the Richmond network).

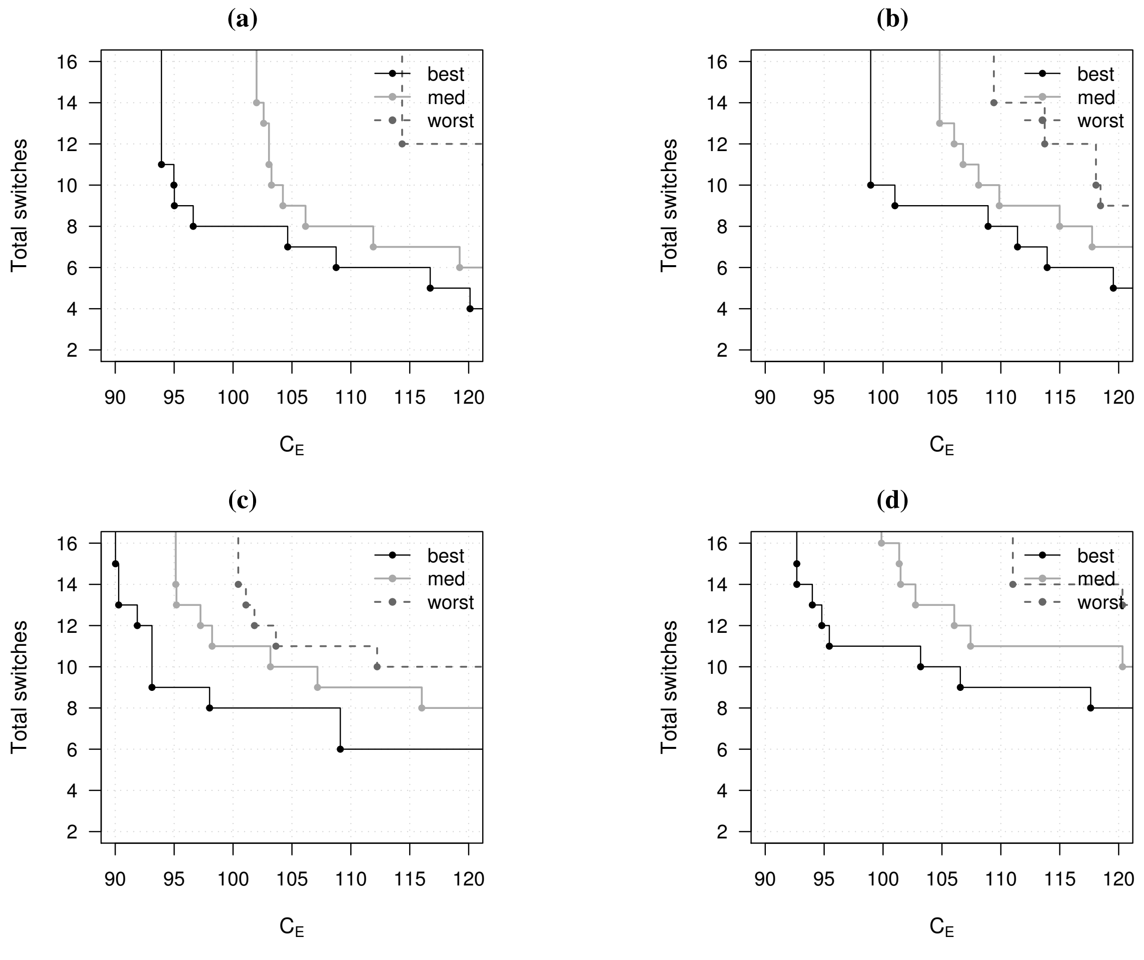


Figure 5.20: Attainment surfaces of SPEA2 using relative time-controlled triggers, *rand-arithmetical* recombination and: (a) $\alpha = 50$ and (b) $\alpha = 100$ with *replace* mutation; (c) $\alpha = 50$ and (d) $\alpha = 100$ with *uniform* mutation (the Richmond network).

mutation. We observe that $\alpha = 50$ almost dominates $\alpha = 100$, particularly in the median and best case. On the other hand, each mutation operator obtains good results with respect to a different objective. This can be observed clearly in Fig. 5.21, which graphically illustrates the differences between the EAFs corresponding to *replace* versus *uniform* mutation (both with $\alpha = 50$ and *rand-arithmetical* recombination). The plot shows that *replace* mutation obtains better schedules than *uniform* mutation for very low number of pump switches. However, for higher number of pump switches, *uniform* mutation obtains the schedules with lowest electricity cost. Since the time-controlled triggers representation enforces a constraint on pump switches, the number of switches is never excessive. We would rather trade higher number of pump switches for reduced electricity cost, and hence, we prefer the results obtained by *uniform* mutation. In conclusion, the settings $\alpha = 50$, *rand-arithmetical* recombination and *uniform* mutation give the best results for the Richmond network when using relative time-controlled triggers

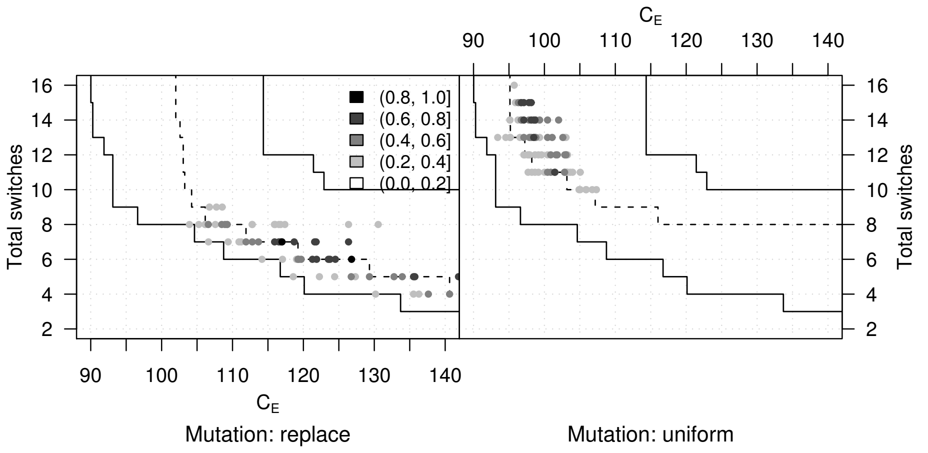


Figure 5.21: Differences in the EAFs of SPEA2 using relative time-controlled triggers representation, $\alpha = 50$ and *rand-arithmetical* recombination (the Richmond network).

5.4.6 Comparison among Representations

In the above sections we have examined the performance of SPEA2 for various parameters settings. We have identified the best settings for each alternative representation of solutions when applied to each of the two network instances under study. These configurations of SPEA2 are summarised in Tables 5.2 and 5.3, for the Vanzyl and Richmond networks respectively.

We now use these configurations of SPEA2 to compare the different representations by means of attainment surfaces in Fig. 5.22 for the Vanzyl network and Fig. 5.23 for the Richmond network. Instead of the best and worst attainment surfaces we plot the 25% and 75% attainment surfaces, which conceptually correspond to the first and third quartiles that are used to define the “box” in a boxplot. The first and third quartiles are less sensitive to outliers than the best and worst attainment surfaces, and hence, are more suited to assess the typical variability of the results. In addition, the results of Hybrid GA (van Zyl, Savic & Walters, 2004) in the Vanzyl network are plotted as points in Fig. 5.22. The original publication (van Zyl, Savic & Walters, 2004) do not provide information about the number of pump switches obtained by Hybrid GA in the case of Richmond network, and hence, we plot the results of Hybrid GA as lines according to their electrical cost in Fig. 5.23.

The Vanzyl Network

The plots in Fig. 5.22 compare the best settings of SPEA2 for each representation in the Vanzyl network. Binary representation is the one that obtains the lowest electrical cost with median values close to 330 (Fig. 5.2a). By comparison, absolute and relative time-controlled triggers obtain median values close to 340 (Fig. 5.2c) and 335 (Fig. 5.2d), respectively. The median attainment functions of these three representations dominate completely the results of Hybrid GA. On the other hand, the representation based on level-controlled triggers obtains much higher electrical cost for the same number of pump switches (Fig. 5.2b).

The Richmond Network

Figure 5.23 compares the best settings of SPEA2 for each representation for the Richmond network. The binary representation obtained the lowest electrical cost, with values lower than 95 (Fig. 5.3a). Relative time-controlled triggers generates as well schedules with electrical cost close to 95 (Fig. 5.3d). By comparison, absolute time-controlled triggers representation obtains slightly larger electrical values (Fig. 5.3c), except for very low number of pump switches ($N^{\text{sw}} < 7$). Lastly, the lowest electrical cost achieved by level-controlled triggers in the median case is close to 110 (Fig. 5.3b), and hence, far from the results obtained by the other representations.

Although level-controlled triggers obtained better results than the other representations for very low number of pump switches ($N^{\text{sw}} < 7$), such number of pump switches is very conservative (less than one switch per pump) and system operators may be more interested in schedules with $N^{\text{sw}} > 7$ that further reduce electrical cost. Both binary and relative time-controlled triggers representations were able to obtain a lower median electrical cost than all runs of Hybrid GA.

Table 5.2: Best parameter settings of SPEA2 using different representations (the Vanzyl network).

		Representation			
		binary	**level-triggers**	**time-triggers**	
				absolute	**relative**
SPEA2	α	50	50	50	50
	μ	5	20	20	5
	Recomb.	two-point	ext.-interm.	one-point	rand-arithm.
	Mutation	flip	replace	replace	uniform

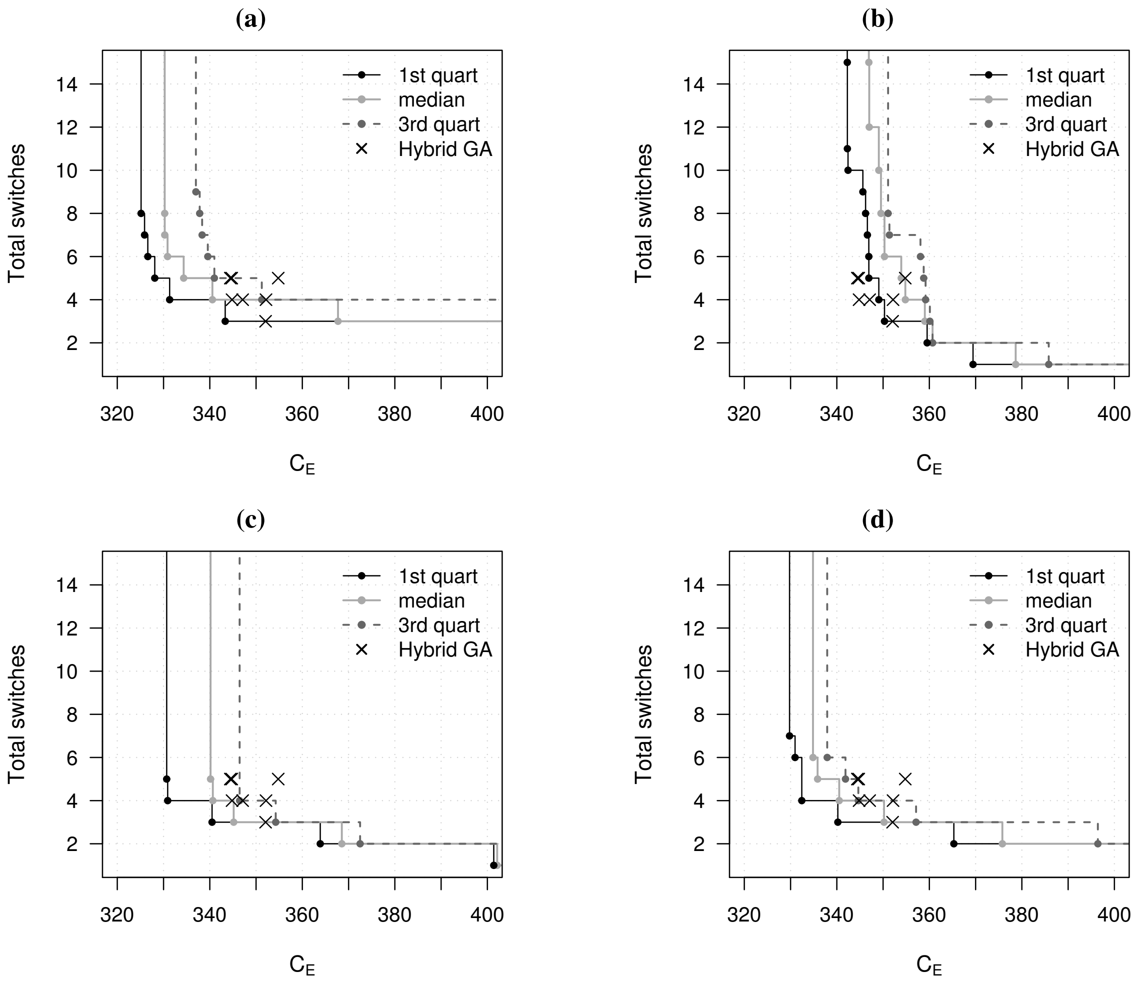


Figure 5.22: Attainment surfaces of SPEA2 using (a) the binary representation, (b) level-controlled triggers, (c) absolute time-controlled triggers and (d) relative time-controlled triggers (the Vanzyl network).

Table 5.3: Best parameter settings of SPEA2 using different representations (the Richmond network).

		Representation			
		binary	**level-triggers**	**time-triggers**	
				absolute	**relative**
SPEA2	α	50	100	50	50
	μ	20	20	5	5
	Recomb.	uniform	ext.-interm.	one-point	rand-arithm.
	Mutation	flip	replace	replace	uniform

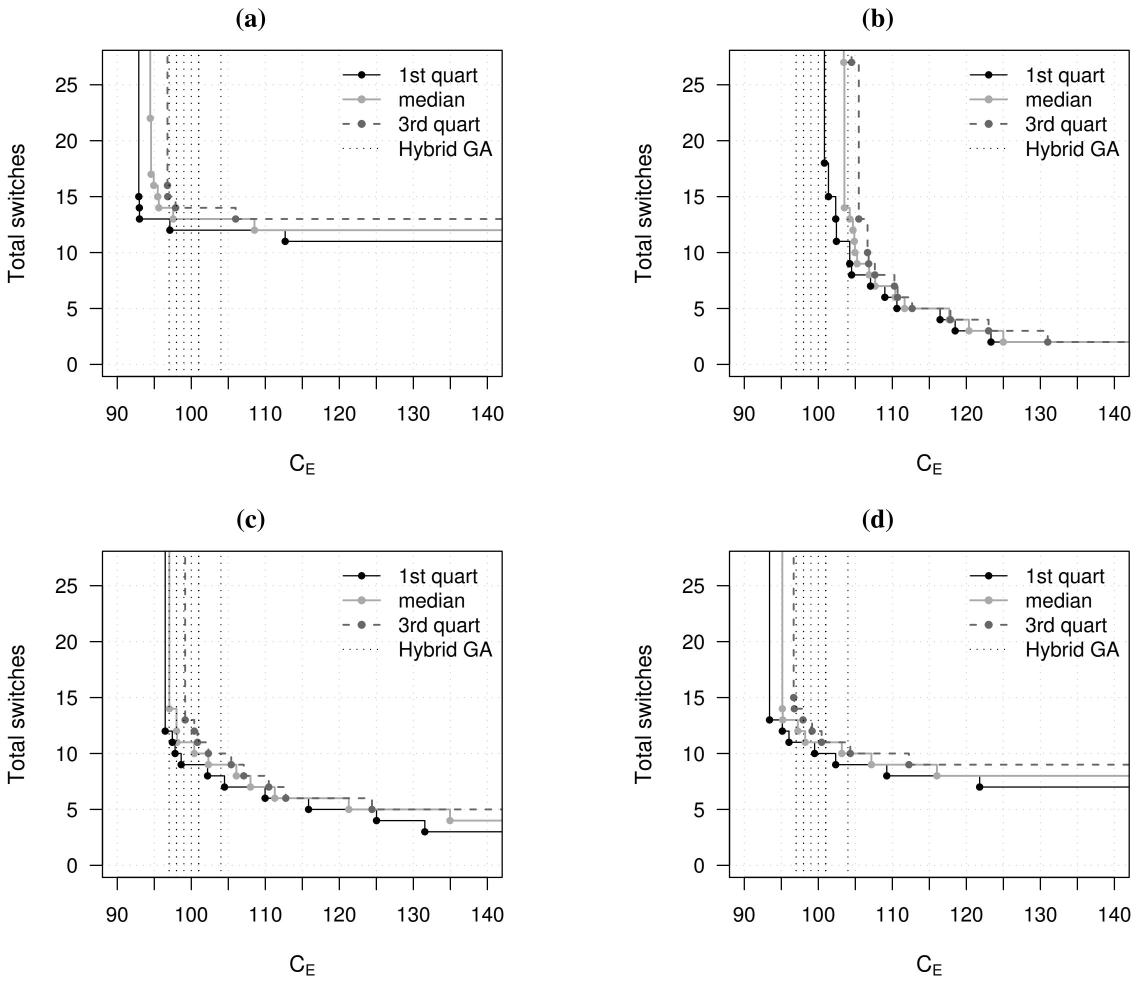


Figure 5.23: Attainment surfaces of SPEA2 using (a) the binary representation, (b) level-controlled triggers, (c) absolute time-controlled triggers and (d) relative time-controlled triggers (the Richmond network).

5.4.7 Summary

We have proposed a bi-objective approach to pump scheduling based on the minimisation of both electrical cost and number of pump switches. Different settings of SPEA2 and four representations have been tested on the two network instances. The experiments with different settings for each representation have shown that the best settings are somewhat similar to those identified for the single-objective SEA in Chapter 4. When compared with the Hybrid GA proposed by van Zyl, Savic & Walters (2004) for the optimisation of level-controlled triggers, SPEA2 using alternative representations is able to obtain lower cost schedules with the additional benefit of providing a set of schedules that model the trade-off between electrical and maintenance costs. Nonetheless, when SPEA2 uses the level-controlled triggers representation, the results of Hybrid GA are superior in terms of electrical cost.

5.5 Maximisation of Idle Intervals

The number of pump switches is the most popular surrogate measure for estimating maintenance costs. However, other factors affect maintenance costs. In particular, if the time elapsed between two operating intervals is excessively short, the fast switching of pumps can damage not only the pumps, but the whole network due to water hammer effect.[2] In order to minimise the wear and tear of pumps and protect the network, it is desirable to increase the time elapsed between two operation intervals of a pump, that is, to *maximise the shortest idle time*.

The *shortest idle time* of a schedule is the shortest time interval during which any pump in the system is idle. Certainly, system operators may accept small increments of electrical costs for longer idle time (see Fig. 5.24 for an example), because of operational flexibility to absorb pressure fluctuations. Short intervals between two active periods also cause wear and tear of pumps that is hard to estimate. Therefore, it is reasonable to assume that longer idle times reduce maintenance costs.

Formally speaking the shortest idle time (I^{T}) of a schedule S with N^{p} pumps can be defined as:

$$I^{\mathrm{T}}(S) = \min_{p=1}^{N^{\mathrm{p}}} I_p^{\mathrm{T}} \quad \text{where} \quad I_p^{\mathrm{T}} = \begin{cases} \min_{i=1}^{N_p^{\mathrm{sw}}}(t_i - t'_i) & \text{if } N_p^{\mathrm{sw}} > 0, \\ T & \text{otherwise} \end{cases} \tag{5.1}$$

[2] A water hammer is produced when the flow of water is suddenly stopped (or changed in direction) and a pressure wave propagates throughout the water distribution network.

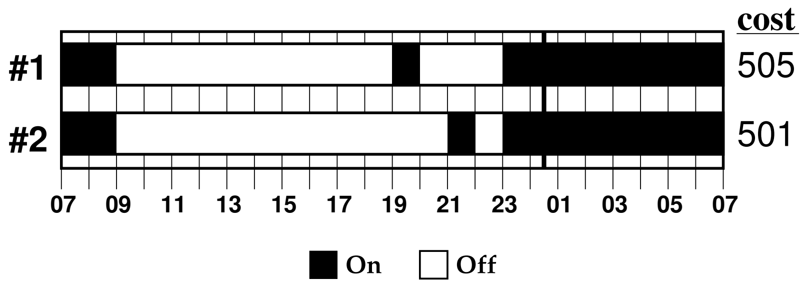


Figure 5.24: Scheduling #2 (bottom) has a lower cost than scheduling #1 (top), but #1 has a larger minimum idle time than #2.

where N_p^{sw} is the number of switches of pump p, t_i is the time when the i-th pump switch (*off*→*on*) occurred, and t'_i is the time when the previous corresponding opposite switch (*on*→*off*) occurred. Since we are interested in avoiding short idle intervals, we consider that a pump operating during the whole scheduling period T has an idle time of T.

The above definition of the shortest idle time I^{T} implies an upper bound to the number of pump switches:

$$\begin{cases} N^{\mathrm{sw}} < \dfrac{N^{\mathrm{p}} \cdot T}{I^{\mathrm{T}}} & \text{if } I^{\mathrm{T}} < T, \\ N^{\mathrm{sw}} = 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

If the representation of pump schedules defines a minimum time interval t_{min}, as it is the case in the binary and time-controlled triggers representations, then the bound is even stricter:

$$\begin{cases} N^{\mathrm{sw}} < \dfrac{N^{\mathrm{p}} \cdot T}{I^{\mathrm{T}} + t_{\mathrm{min}}} & \text{if } I^{\mathrm{T}} < T, \\ N^{\mathrm{sw}} = 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

Although a larger idle time does not directly imply a lower number of pump switches, as the example in Fig. 5.25 illustrates, the maximisation of the shortest idle time indirectly minimises the number of pump switches by reducing the upper bound described in Eq. (5.3).

The maximisation of the shortest idle time is particularly relevant in the case of level-controlled triggers representation. In level-controlled triggers representation, the shortest idle time between operating periods is only limited by the time step used by the hydraulic simulator. This may generate schedules similar to the one shown in Fig. 5.7 on page 91. Such schedule is feasible according to the constraints we have considered so far: zero volume deficit, no pressure violations, and no warnings from the simulator. On the other hand, the frequency of pump switches is excessive. Short time intervals allow to fine-tune the schedule of pumps to reduce costs. Hence, it is not strange that the optimisation

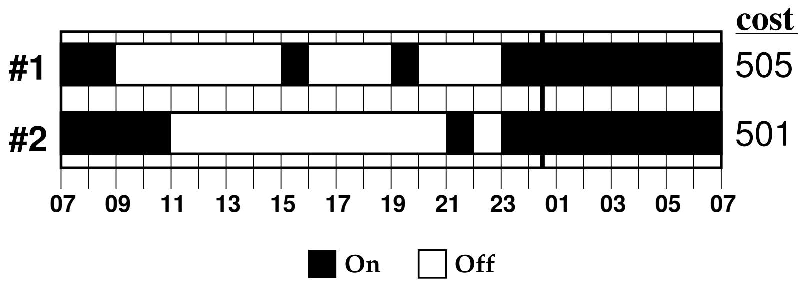


Figure 5.25: Scheduling #2 (bottom) has a lower number of pump switches than scheduling #1 (top), but #1 has a larger minimum idle time than #2.

algorithm generated such schedule only taking into account the above constraints. In reality, however, excessively short intervals are impractical and may damage the pumps and the rest of the system due to pressure surges. Therefore, maximising the length of the shortest idle time is a relevant objective in the level-controlled triggers representation.

The binary and time-controlled triggers representations have a minimum length of time interval (t_{min}), typically one hour. This minimum time interval is a lower bound of the idle time ($I_T \geq t_{\text{min}}$), and, hence, by tuning t_{min}, the minimum idle time can be made as large as desired. However, different values of t_{min} may influence the performance of the algorithm, as we investigated in Section 4.5. Our experiments showed that values of t_{min} longer than one hour do not allow sufficient flexibility to generate a satisfactory schedule in terms of cost. Experiments showed, as well, that values smaller than one hour, such as 10 minutes, may lead to better results if the algorithm is allowed to run for enough time. For this reason, operators may wish to use a value of t_{min} of one hour or smaller to obtain a good performance from the optimisation algorithm, while at the same time expect the shortest idle time to be larger than t_{min} hour. In addition to this, the shortest idle time could be a better surrogate measure of maintenance cost than the total number of pump switches in the case of time-controlled triggers, since the time-controlled triggers representation already implicitly constraints the number of switches per pump.

The following experiments will test the minimisation of electrical cost (C_{E}) and the maximisation of shortest idle time (I^{T}) as the two objectives of a multi-objective problem defined in terms of Pareto optimality. As in the previous section, we use SPEA2 as the multi-objective optimiser. Only representations based on level-controlled and relative time-controlled triggers are tested because, for the reasons given above, they are the most likely to benefit from this approach.

5.5.1 Experimental Setup

Each combination of parameters is repeated a number of times with different random seeds, exactly 25 runs for the Vanzyl network and 15 times for the Richmond network. Each run was stopped after 6000 evaluations in the case of the Vanzyl network, and 8000 evaluations for the Richmond network.

We applied an analysis based on the hypervolume indicator (HV) and the empirical attainment function (EAF), similar to the one described in the previous Section 5.4, to identify the best parameters. For brevity, we will discuss the best results only.

5.5.2 Level-controlled Triggers

We performed several experiments with $\alpha = \{50, 100, 200\}$, $\mu = 5$ (the Vanzyl network) or $\mu = 20$ (the Richmond network), *extended-intermediate* recombination and either *gaussian* or *replace* mutation.

The Vanzyl Network. The best results are obtained by *extended-intermediate* recombination, *replace* mutation and $\alpha = \{50, 100\}$. Figure 5.26 shows the corresponding attainment surfaces. A semi-log plot is used because there are numerous solutions with a minimum idle time (I^{T}) between 100 and 500 minutes, while there are far less solutions with a value higher than 500 and very few higher than 1000 minutes (the maximum being 1440 minutes). The shape of the attainment surfaces shows a small trade-off between electrical cost and idle time. In other words, big differences in idle time do not correspond to big differences in electrical cost. SPEA2 with level-controlled triggers is able to find schedules with more than 500 minutes of shortest idle time and electrical cost lower than 360.

The Richmond Network. Figure 5.27 shows the attainment surfaces corresponding to the results obtained by SPEA2 for the Richmond network with *extended-intermediate* recombination, *replace* mutation and $\alpha = \{50, 100\}$. The plots are noticeably different from the ones corresponding to the Vanzyl network. There is a sudden increase of electrical cost (C_{E}) from 140 to more than 200 for an idle time close to 800 minutes. The long idle time means that most pumps are either inactive during long periods of time or active during the whole scheduling period. The active pumps cause an important increase in electrical cost, however, they do not have idle periods, and, thus, they do not affect the total idle time of the schedule. A further improvement of the algorithm may discard such solutions as uninteresting during the optimisation run in order to focus the search on schedules with lower electrical cost.

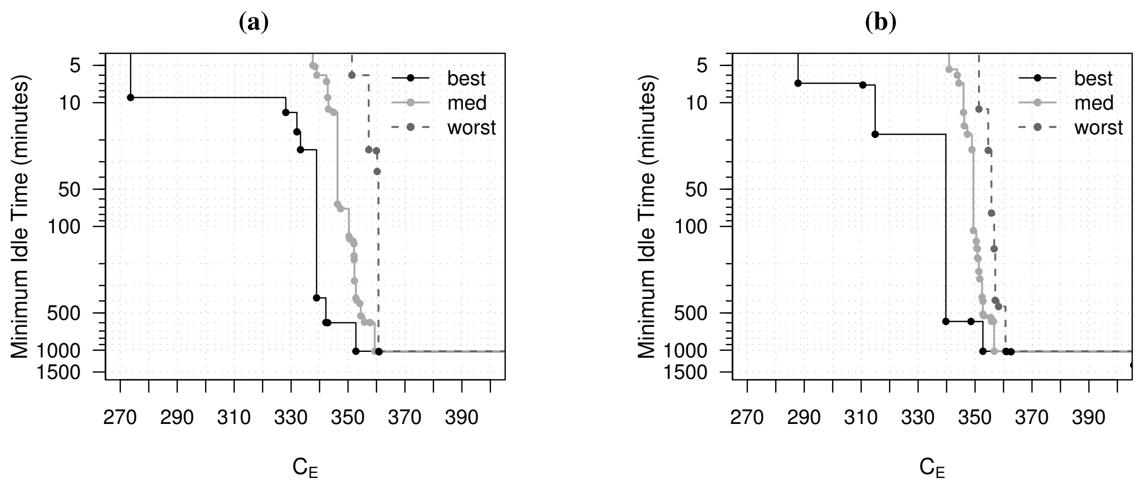


Figure 5.26: Attainment surfaces of SPEA2 using level-controlled triggers, *extended-intermediate* recombination, *replace* mutation, and (a) $\alpha = 50$ and (b) $\alpha = 100$ (the Vanzyl network).

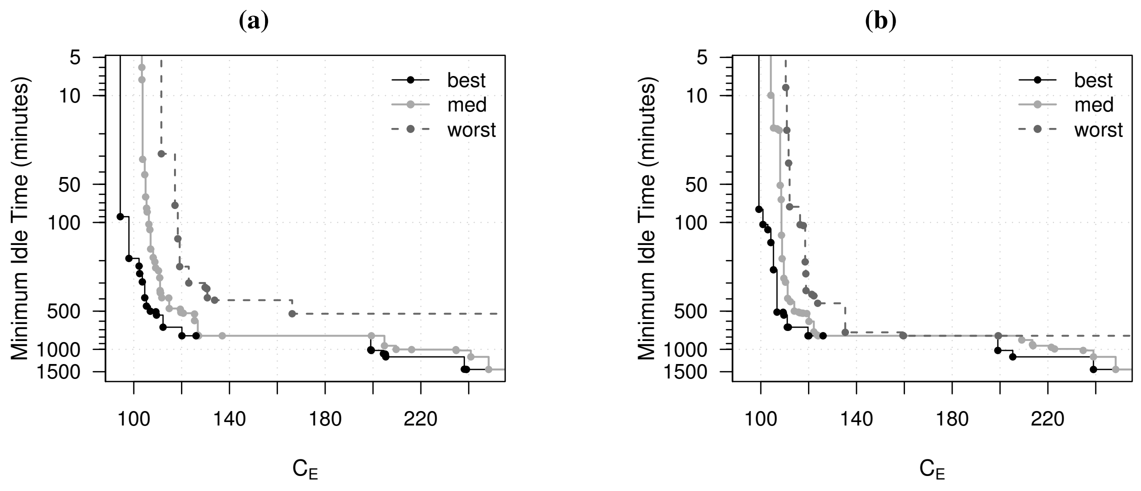


Figure 5.27: Attainment surfaces of SPEA2 using level-controlled triggers, *extended-intermediate* recombination, *replace* mutation, and (a) $\alpha = 50$ and (b) $\alpha = 100$ (the Richmond network).

5.5.3 Relative Time-controlled Triggers

We run similar experiments using relative time-controlled triggers with a limit of switches per pump of $N_p^{\text{sw}} \leq 3$ and a minimum time interval of $t_{\text{min}} = 1$ hour and $t_{\text{min}} = 1$ minute. SPEA2 parameters were *rand-arithmetical*, *two-point* and *uniform* recombination, $\mu = 5$, $\alpha = \{50, 100, 200\}$ and either *replace*, *uniform* and no mutation. The best results are further examined in the following paragraphs.

The Vanzyl Network. Figure 5.28 shows the attainment surfaces corresponding to the best results using $t_{\text{min}} = 1$ hour, which are obtained when $\alpha = 50$ and either *replace* or *uniform* mutation are used. Since the minimum time interval is one hour, the possible values of I^{T} are multiples of one hour. In the case of *replace* mutation (Fig. 5.28b), SPEA2 generates the schedules with the higher I^{T}. In particular, the resulting schedules have more than 900 minutes of idle time and electrical cost close to 360 in the median case. However, the lowest electrical cost is obtained using *uniform* mutation (Fig. 5.28a), which generates schedules with an electrical cost lower than 340 in the median case.

By setting $t_{\text{min}} = 1$ minute, we add more flexibility to the potential schedules generated by SPEA2. Among the recombination operators, *rand-arithmetical* produces the best results. For the remainder parameters, the best settings are either the combination of *uniform* mutation and $\alpha = 100$ (Fig. 5.29a), or the combination of *replace* mutation and $\alpha = 50$ (Fig. 5.29b). Similar to the results obtained for $t_{\text{min}} = 1$ hour, *uniform* mutation obtains the lowest electrical cost schedules, whereas *replace* mutation performs better in the region with longer idle time. In contrast, SPEA2 using $t_{\text{min}} = 1$ minute finds many more schedules than for $t_{\text{min}} = 1$ hour. The lowest electrical costs are found using $t_{\text{min}} = 1$ hour. The reason for this was pointed out already in Section 4.5: a smaller time interval leads to a great increment of the search space, and hence, optimisers may need more computation time to find good schedules.

The Richmond Network. The best results for the Richmond network are shown in Fig. 5.30 for $t_{\text{min}} = 1$ hour, and in Fig. 5.31 for $t_{\text{min}} = 1$ minute. From these plots, one can draw conclusions similar to those for the Vanzyl network. First, *uniform* mutation obtains the lowest electrical cost, whereas *replace* mutation finds schedules with the longest idle time. Second, the greater flexibility of $t_{\text{min}} = 1$ minute allows SPEA2 to generate many more solutions than using $t_{\text{min}} = 1$ hour. Finally, this greater flexibility comes with the burden of a larger search space, and hence, despite the flexibility, the lowest electrical costs are found when $t_{\text{min}} = 1$ hour is used.

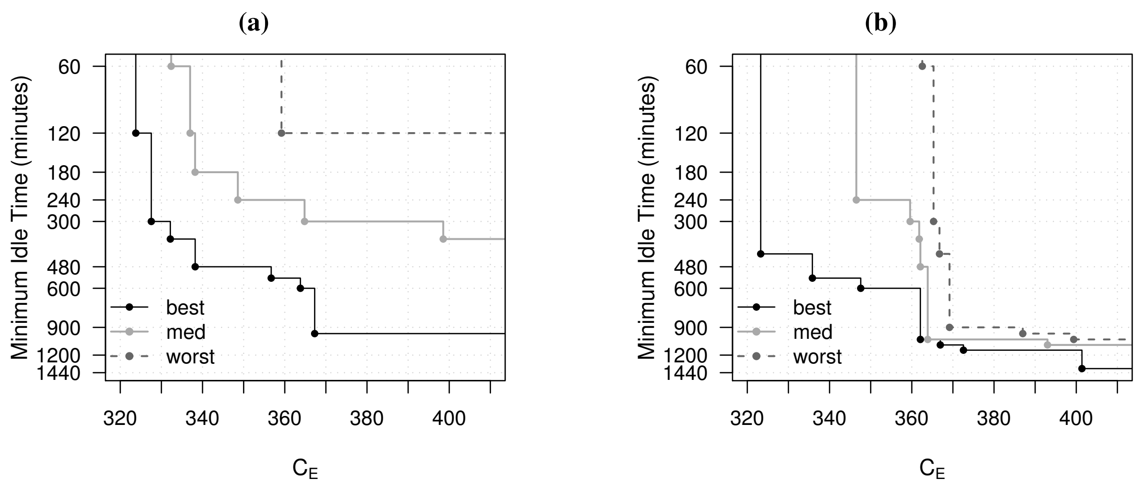


Figure 5.28: Attainment surfaces of SPEA2 using relative time-controlled triggers ($t_{\text{min}} = 1$ hour) with $\alpha = 50$ and (a) *rand-arithmetical* recombination and *uniform* mutation, and (b) *two-point* recombination and *replace* mutation (the Vanzyl network).

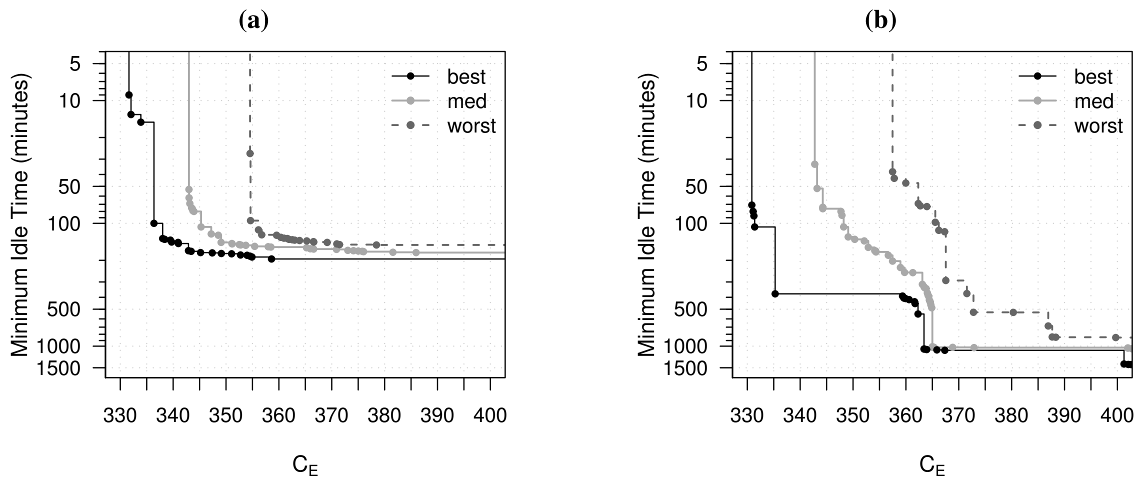


Figure 5.29: Attainment surfaces of SPEA2 using relative time-controlled triggers ($t_{\text{min}} = 1$ minute) with *rand-arithmetical* recombination and (a) $\alpha = 100$ and *uniform* mutation, and (b) $\alpha = 50$ and *replace* mutation (the Vanzyl network).

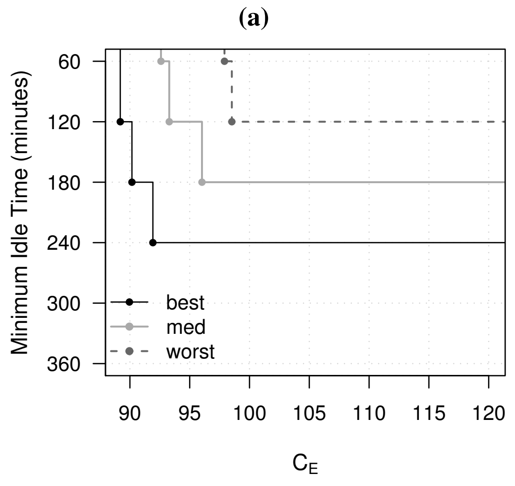


Figure 5.30: Attainment surfaces of SPEA2 using relative time-controlled triggers ($t_{\text{min}} = 1$ hour), *rand-arithmetical* recombination, $\alpha = 50$, and (a) *uniform* mutation and (b) *replace* mutation (the Richmond network).

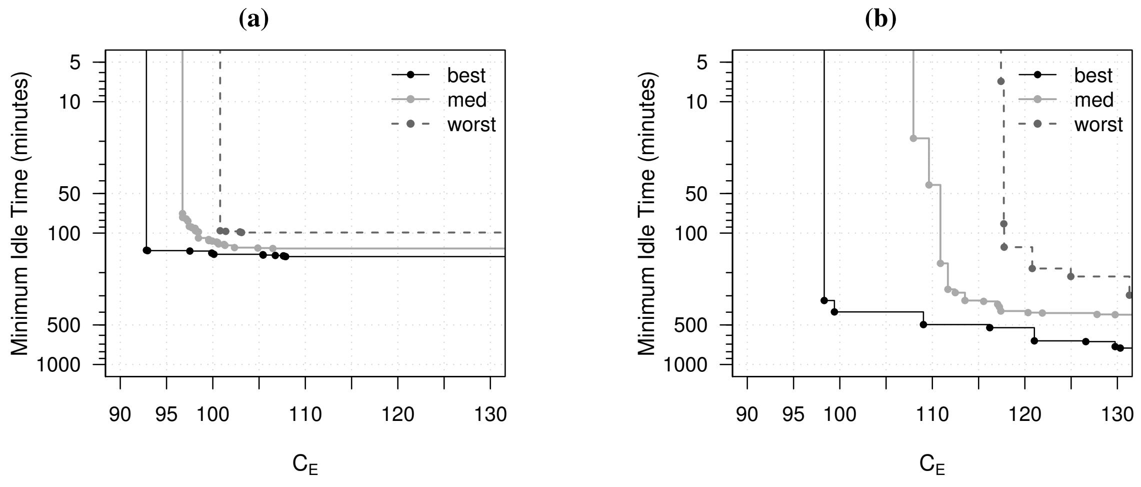


Figure 5.31: Attainment surfaces of SPEA2 using relative time-controlled triggers ($t_{\text{min}} = 1$ minute) with $\alpha = 50$ and (a) *rand-arithmetical* recombination and *uniform* mutation, and (b) *uniform* recombination and *replace* mutation (the Richmond network).

5.5.4 Summary

This section has proposed a multi-objective approach that minimises the electrical cost (C_E) and maximises the minimum idle time (I^T). Experimental results show that SPEA2 is able to find a wide range of schedules with low electrical cost and very long minimum idle time. In the case of relative time-controlled triggers, we noticed that the mutation operator has a strong influence on the shape of the resulting Pareto sets. In particular, SPEA2 with *uniform* mutation generated the lowest electrical cost, while SPEA2 with *replace* mutation focused on the region of the objective space with the longest idle time. This effect of mutation was consistently observed in the two test networks.

5.6 Summary

In this chapter we have presented the results of two different multi-objective approaches with different optimisation objectives in addition to the electrical cost: minimisation of pump switches and maximisation of shortest idle time. We adapted a modern multi-objective algorithm, SPEA2 (Zitzler, Laumanns & Thiele, 2002) to the pump scheduling problem and linked it to EPANET in order to tackle complex water distribution networks. In order to show the viability of the multi-objective approach, we compared its performance with the results obtained by a state-of-the-art single-objective algorithm (van Zyl, Savic & Walters, 2004) for the optimisation of level-controlled triggers. In our analysis of experimental results, we used up-to-date assessment methods for multi-objective algorithms. In particular, we avoided unreliable quality indicators in favour of the Pareto-compliant hypervolume indicator and the more powerful empirical attainment function.

The first objective considered was the minimisation of pump switches. Our empirical analysis showed that SPEA2 consistently obtains good schedules for the two network instances tested, in particular when using the binary and time-controlled triggers representation. A single run of SPEA2 using either binary or relative time-controlled triggers representations generates schedules with both lower electrical cost and lower number of pump switches than the singe-objective state-of-the-art algorithm designed for level-controlled triggers. However, the single-objective algorithm is still superior when compared with SPEA2 using level-controlled triggers. From a single run, one can obtain a number of schedules that model the trade-off between electrical cost and number of pump switches.

The second approach investigated in this chapter is the use of the shortest idle time between operating periods as a surrogate measure of maintenance costs. We proposed a formal definition of the concept and examined its benefits. The maximisation of the shortest idle time may solve the inherent problem of pressure surges that frequently occur in schedules generated with a level-controlled triggers approach. As for the binary and time-controlled triggers, the possibility of adding more flexibility to schedules by using smaller time intervals introduces the risk of very short idle intervals between pump switches, and hence, sudden pressure fluctuations. Maximising the shortest idle time would permit the network to accommodate pressure changes caused by pump operations. Our experimental results showed that schedules with very long idle times may be found with slightly higher electrical cost than those obtained by SEA in Chapter 4.

In conclusion, we can say that our experimental analysis showed that multi-objective optimisation is very attractive if we are interested in the trade-off between conflicting objectives. However, if the goal is to obtain the lowest electricity cost while keeping constraints within certain limits, then the single-objective SEA is more effective. The next chapter investigates the optimisation strategy called Ant Colony Optimisation as an alternative to evolutionary algorithms with the goal of further improving over the results of SEA.

Chapter 6

Ant Colony Optimisation

In the previous chapter various representations were empirically examined through the use of an evolutionary algorithm. The time-controlled triggers representation obtained the best results overall. In this chapter, the time-controlled trigger representation is further explored as the means to apply a different optimisation technique, Ant Colony Optimisation, to the pump scheduling problem. Ant Colony Optimisation (Dorigo & Stützle, 2004) is a biologically-inspired optimisation method that mimics the technique used by real ants for optimising the path between the nest and a source of food. Certain ant species follow chemical substances, called pheromones, previously laid by other ants in their trails. The trail followed by an ant depends strongly on the amount of pheromone present at each possible direction. Thus, pheromones work as a communication mechanism and also as a reinforcement learning process. Ant Colony Optimisation (ACO) has been applied to a different problem, namely the design of water distribution networks (Maier *et al.*, 2003). Here we propose to apply ACO to the operational optimisation of water distribution networks. Hence, in this chapter, the pump scheduling problem along with the proposed time-controlled triggers representation are adapted to the ACO framework. First, the approach is tested by means of an algorithm similar to the Ant System (Dorigo, Maniezzo & Colorni, 1996), which is considered to be the first ACO algorithm. Next, a more advanced ACO algorithm, specifically the Max-Min Ant System (Stützle & Hoos, 2000), is also implemented and empirically evaluated. ACO algorithms show notable improvements in comparison to previous algorithms for the pump scheduling problem.

6.1 Ant Colony Optimisation

Ant Colony Optimisation (ACO) is a meta-heuristic, a problem-independent optimisation technique, that has been successfully applied to several optimisation problems. Dorigo

& Stützle (2004) provide an extensive introduction to this technique and many examples of its application. The concept of ACO was inspired by the foraging behaviour of some species of real ants. These ants are able to find an optimal path between nest and food through a kind of indirect communication known as stigmergy, by means of trails of a chemical substance called *pheromone*. Pheromone is laid by ants along their way when moving. While ants wandering unexplored areas choose their path basically in a random fashion, they tend to follow those paths marked with pheromone by other ants, stochastically choosing, with a higher probability, those paths marked with a higher amount of pheromone. The path chosen by an ant is reinforced by its own pheromone trail, increasing the probability of other ants following the same path. Successive reinforcement of the pheromone trails results in a positive feedback mechanism. Although initially all paths are equally probable, ants travelling through shorter paths are able to reach the source of food faster. When retracing their steps, ants find that the shorter path has a higher amount of pheromone, so more ants tend to follow it. Therefore, pheromone increases faster along the shorter paths, which in turn attracts more ants to them. Meanwhile, the pheromone of paths that are not reinforced slowly *evaporates*.

Ant Colony Optimisation (ACO) is inspired by this biological behaviour. In ACO, very simple agents, analogous to *artificial ants*, stochastically build paths in a graph. Such graph represents an optimisation problem, where nodes are decision points and edges represent possible choices or *solution components*. A path over the graph defines a candidate solution to the optimisation problem. At each decision point (node), an individual ant stochastically chooses a solution component (edge) to add to its current path. Solution components are added iteratively until a candidate solution is completed. Each stochastic decision is influenced by numerical information, analogous to *pheromone trails*, associated with each edge in the graph. The higher the pheromone value of a particular solution component, the greater its probability of selection by an ant to add it to its current partial solution. Ants intensify the pheromone information associated with solution components that belong to good solutions. This increases the probability of choosing the most attractive edges of the graph when constructing new solutions. In addition, all pheromone values are decreased at every iteration, an operation known as *evaporation*. Evaporation decreases faster those pheromone values that have not been recently reinforced, that is, those pheromone values associated with solution components that do not appear frequently in good solutions. As a result, the search is directed towards the most promising regions of the search space.

Optimisation algorithms that implement this framework are called ACO algorithms. Most ACO algorithms consist of a main loop where: (*1*) ants construct solutions taking into account the pheromone information; (*2*) solutions are evaluated according to an objective function; and (*3*) pheromone information is updated, increasing the pheromone value associated with solution components that are part of the best solutions.

6.2 ACO Applied to the Pump Scheduling Problem

In order to apply ACO to the pump scheduling problem, we need to define an ACO algorithm, a representation of the pheromones and a construction mechanism used by ants to build partial solutions. Despite being a critical aspect, the particular ACO algorithm can be quite independent of the problem. This is not the case for the representation of the pheromones and the construction mechanism, which are closely related to the representation of solutions. In the next section, we describe a variant of the Ant System algorithm and introduce fundamental notation and concepts of the ACO framework. These concepts are used to explain later the particular pheromone representation and construction mechanism that we propose in this work for the pump scheduling problem. Finally, we describe two alternative pheromone update methods and a proposed heuristic information for the pump scheduling problem.

6.2.1 The Ant System Algorithm

We adapt the original Ant System (AS) (Dorigo, 1992; Dorigo, Maniezzo & Colorni, 1996), which was the first ACO algorithm, with some minor modifications. The schema of the resulting algorithm is shown in Fig. 6.1. Let $\tau_{ij}(t)$ be a numerical value called *pheromone* associated with each solution component (i, j), where the meaning of a solution component is left undefined for now. This pheromone value is updated during the algorithm run, and it depends on the current iteration t. First, pheromone values are initialised to a constant value τ_0 (Fig. 6.1, line 1). Then, a main loop is repeated until a termination criteria, such as maximum number of objective function evaluations, is met. Within this loop, a number of ants (A) construct candidate solutions to the problem. Each ant a constructs a single solution S_a by iteratively adding solution components (Fig. 6.1, lines 4–9), which is equivalent to building a path in the graph by choosing edges at each decision point. Thus, an ant a at each decision point i chooses a single solution component (i, j) among a set $\mathcal{N}_i(a)$ of possible alternatives. This set $\mathcal{N}_i(a)$ is called the feasible neighbourhood of ant a and it may vary according to the current partial solution of ant a. The probability of choosing a solution component (i, j) is given by:

$$p_{ij}(a,t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i(a)} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \quad \text{if } j \in \mathcal{N}_i(a) \tag{6.1}$$

```
1: $t \leftarrow 0$, initialise pheromones: $\tau_{ij}(0) \leftarrow \tau_0$
2: while termination criteria not met do
3:   for each ant $a \in \{1, \ldots, A\}$ do
4:     $S_a \leftarrow \varnothing$    /* empty solution */
5:     repeat
6:       calculate probability $p_{il}(a, t)$ for each $l \in \mathcal{N}_i(a)$ following Eq. (6.1)
7:       stochastically choose solution component $(i, j)$ such as $j \in \mathcal{N}_i(a)$
8:       add solution component to partial solution:  $S_a \leftarrow S_a \cup (i, j)$
9:     until $S_a$ is a complete solution
10:  end for
11:  for each ant $a \in \{1, \ldots, A\}$ do
12:    evaluate solution $S_a$
13:  end for
14:  /* identify iteration-best */
15:  $S^{\text{ib}} \leftarrow$ best solution from $\{S_1, \ldots, S_A\}$
16:  if $S^{\text{ib}}$ better than $S^{\text{bf}}$ then
17:    $S^{\text{bf}} \leftarrow S^{\text{ib}}$    /* update best-so-far */
18:  end if
19:  evaporate and update pheromone following Eq. 6.2
20:  $t \leftarrow t + 1$    /* next iteration */
21: end while
22: return $S^{\text{bf}}$
```

Figure 6.1: Algorithmic schema of ACO.

where η_{ij} is a heuristic value associated with solution component (i, j), and represents an estimation of the benefit of choosing j over the other alternatives in $\mathcal{N}_i(a)$; and α and β weigh the relative influence of the pheromone and the heuristic information on the final probability $p_{ij}(a, t)$.

After each ant has constructed a new solution, they are evaluated and ranked to identify the *iteration-best* solution (S^{ib}), i.e., the best solution among the ones constructed in the current iteration t. The best solution found in the current run of the ACO algorithm, called *best-so-far* solution (S^{bf}), is also updated accordingly.

Finally, pheromone information is updated to reflect the experience acquired by the ants through the evaluation of their solutions. The pheromone update is completed in two steps. In the first step, pheromones of all solution components are *evaporated* by decreasing pheromone values by a constant factor. In the second step, pheromones of solution components that are part of the best solutions are *reinforced* by increasing their pheromone values. The evaporation mechanism exponentially decreases the pheromone values of solution components that are not reinforced, reducing the influence of pheromones over time. A high evaporation rate implies that decisions by ants are mainly influenced by recently reinforced pheromone values. This results in a faster convergence to the best solution components that were recently constructed. On the other hand, a low evaporation rate implies a gradual differentiation between early reinforced edges and recently reinforced edges, resulting in a higher exploration of solution components not belonging to the best solutions recently constructed. Both operations, evaporation and reinforcement, can be formalised as:

$$\tau_{ij}(t+1) = \rho\,\tau_{ij}(t) + \Delta\tau_{ij} \tag{6.2}$$

where $\rho \in [0,1]$ is a parameter sometimes called *persistence factor* that determines the evaporation rate ($1-\rho$), and $\Delta\tau_{ij}$ is the amount of pheromone deposited in solution components belonging to a solution selected for update (S^{best}) as given by:

$$\Delta\tau_{ij} = \begin{cases} \Delta\tau & \text{if edge } (i,j) \text{ is part of } S^{\text{best}}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

where $\Delta\tau$ may be a function of the objective value of S^{best} or a user-defined constant. According to Dorigo & Stützle (2004), the only requirement is that $\Delta\tau$ is non-increasing with respect to the value of the objective function $f()$, that is, $f(S) < f(S') \Rightarrow \Delta\tau(S) \geq \Delta\tau(S')$. In the simplest case, $\Delta\tau$ can be the same constant value for all ants.

Selecting just one ant S^{best} for update focuses the search on the best solutions generated, rather than adding pheromone amount for every ant depending on its objective function value. This pheromone reinforcement method is different from the update method used in the original Ant System (Dorigo, Maniezzo & Colorni, 1996), where all ants in the current iteration deposited an amount of pheromone relative to its objective function value. Nevertheless, update methods that use just one solution to reinforce the pheromone values, either the *iteration-best* or *best-so-far* solutions, are widely used in many modern ACO algorithms (Dorigo & Gambardella, 1997; Dorigo & Stützle, 2004; Stützle & Hoos, 2000).

6.2.2 Pheromone Information

In the description of the Ant System above, we did not define what is a solution component (i,j), and, thus, it is unclear what a pheromone value τ_{ij} and the feasible neighbourhood of an ant $\mathcal{N}_i(a)$ actually represent. The purpose of this ambiguity is to show that the algorithm described above does not depend on a particular pheromone representation. Yet, the pheromone representation is mainly determined by the definition of solution component, and, hence by the representation of candidate solutions. Thus, a representation of solutions must be chosen at this point.

In Chapter 3, four different representations, two traditional and two new alternatives, were discussed thoroughly. Later, in Chapter 4, the four representations were compared empirically by means of an evolutionary algorithm. The one called *relative time-controlled triggers* produced better results over the other three. Thus, in the following we are going to focus on this representation and apply the ACO technique. Besides, the time-controlled triggers representation is more natural for the application of ACO, since it is a discrete representation, while level-controlled triggers are continuous variables. Unlike the binary representation, in time-controlled triggers representation there are more than two choices.

As a brief reminder of Section 3.3.2, in the time-controlled triggers representation, a candidate solution (S) of the pump scheduling problem has the following form:

$$S = \{s^1, \dots, s^{N^{\mathrm{p}}}\} \quad \text{, where each} \quad s^p = \{\phi_1, \phi'_1, \dots, \phi_{SW}, \phi'_{SW}\} \tag{6.4}$$

where s^p is the schedule of pump p, N^{p} is the number of pumps, SW is the maximum number of switches per pump, and each pair of decision variables $\{\phi_i, \phi'_i\}$ define a single pump switch. When the decision variables represent *relative* time, then each ϕ_i gives the length of time during which the pump is inactive since the previous status change, while ϕ'_i is the length of time during which the pump is active since the previous status change. The value of ϕ_0 is relative to the start of the scheduling period. The relative time-controlled triggers representation has the following constraint:

$$\sum_{i=1}^{SW} (\phi_i + \phi'_i) \; = \; T \tag{6.5}$$

where T is the scheduling period.

The strictness of the constraint on the number of switches determines the range of the decision variables. The simplest option is to force an exact number of switches per pump, $N_p^{\mathrm{sw}} = SW$ (Eq. 2.7). In this case, each integer ϕ_i (or ϕ'_i) must be greater than zero. If there are $(2 \cdot SW - 1)$ integers having a value of one hour each, then, following constraint Eq. (6.5), the remaining integers must be equal to $(T - 2 \cdot SW + 1)$ hours. Thus, $\phi_i, \phi'_i \in [1, (T - 2 \cdot SW + 1)]$. On the other hand, if a lower number of pump switches is allowed, $N_p^{\mathrm{sw}} \leq SW$ (Eq. 2.8), then $\phi_i, \phi'_i \in [0, T]$.

This formulation is completely equivalent to the one given in Section 3.3.2, except that we use ϕ_i instead of t_i to avoid any confusion with the parameter t that represents the current iteration in ACO formulation. In addition, Eq. (6.5) above is stricter than Eq. (3.5).

A few preliminary empirical tests showed better results with this approach, particularly if the constraint on the number of pump switches is not strict and decision variables may take a value of zero.

In order to define the pheromone information and the construction mechanism followed by ants, the time-controlled triggers representation has to be translated into a graph formulation. For simplicity, let us first restrict to the problem of scheduling a single pump with a maximum of SW pump switches. Following the relative time-controlled triggers representation, each pump switch involves a pair of time intervals during which a pump is, respectively, *off* and *on*. Thus, a complete schedule contains $2 \cdot SW$ intervals and the total duration of all intervals must be equal to T. A solution component (i, j) may be defined as the assignment of a duration of j time units (typically, hours) to interval ϕ_i. Associating each decision point to a time interval ϕ_i and the j^{th} edge to a duration of j time units, the result is the *multidigraph*—a directed graph which is permitted to have edges with the same source and target nodes— shown in Fig. 6.2. The pheromone information τ_{ij} is therefore defined as the desirability of assigning a duration of j hours to interval ϕ_i. The next section explains how this assignment takes place.

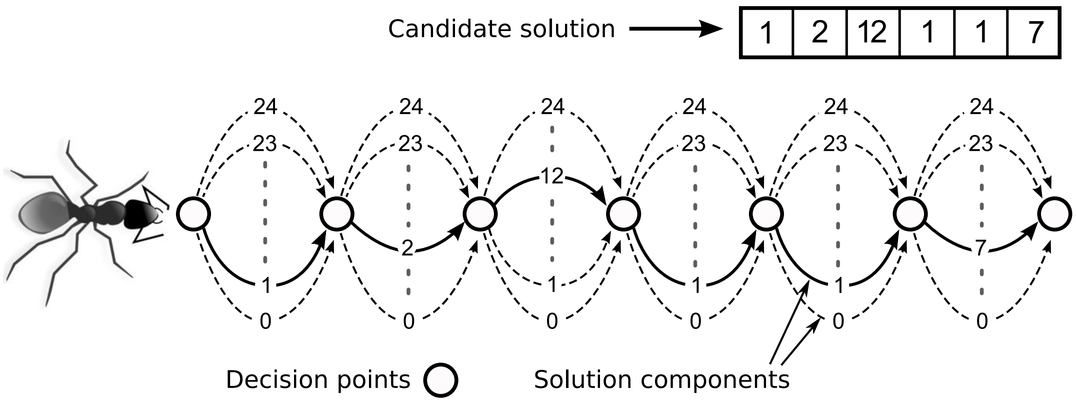


Figure 6.2: Representation of a solution as a path in a graph.

6.2.3 Constructing Schedules in ACO

Using the graph formulation described in the previous section, an ant constructs candidate solutions by moving from one decision point ϕ_i, representing a time interval, to the next and choosing at each step one of the possible edges that correspond to different durations. However, in order to satisfy constraint Eq. (6.5) on the previous page, not all edges will be available at each decision point. In fact, as the ant progresses through the graph, some edges become infeasible. The feasible neighbourhood of ant a at decision point i, as used in Eq. (6.1), is limited to:

$$\mathcal{N}_i(a) = \begin{cases} \{1, 2, \dots, (T - 2 \cdot SW + 1 - T_a)\} & \text{if } N_p^{\text{sw}} = SW, \\ \{0, 1, 2, \dots, (T - T_a)\} & \text{if } N_p^{\text{sw}} \leq SW, \end{cases} \tag{6.6}$$

where i is the current decision point (time interval), and T_a is the total duration of intervals already assigned in the partial solution being constructed by ant a. That is, after one solution component (i, j) has been chosen by ant a to be added to its partial solution, T_a is updated by adding the number of hours j. Moreover, the last solution component is not chosen stochastically. On the contrary, at the last decision point, the number of hours is directly assigned such that the total duration of the schedule is equal to T.

An important point in this schedule construction mechanism is that ants do not always visit decision points in the same order. Otherwise time intervals assigned earlier would have more chances of having a longer duration. In other words, if the last decision point were to always correspond to the same time interval (e.g., the last interval $i = SW$), such interval would have a higher probability of being assigned a shorter duration. To avoid introducing such bias, time intervals are considered in random order, so the last decision point does not necessarily correspond to the last interval of the schedule.

The formulation of the pheromone representation and construction mechanism described above for a single pump can be extended to more than one pump just associating a different pheromone matrix to each pump (τ_{ij}^p). The schedule of each pump is constructed by using the values of its own pheromone matrix and Eq. (6.1). This is equivalent to associating one independent graph to each pump, such that each ant constructs one path in each graph in order to build a complete schedule of all pumps. Similarly, each pheromone matrix is updated following Eq. (6.3) but only considering those components of solution S^{best} associated with the corresponding pump.

6.2.4 Heuristic Information for Pump Scheduling

The heuristic information η_{ij} in Eq. (6.1) is meant to give some estimation of the goodness of solution component (i, j). We consider the following approach:

$$\eta_{ij} = \begin{cases} \frac{T-j}{T} & \text{if pump is } \textit{on} \text{ during interval } i \\ \frac{j}{T} & \text{if pump is } \textit{off} \text{ during interval } i \end{cases} \tag{6.7}$$

This introduces a linear influence depending on the length of the interval, that is, shorter active intervals and longer inactive intervals are preferred. If the value of η_{ij} is zero then, it is taken as $\frac{0.1}{T}$, to allow pheromones to have some influence in the result.

This rule of thumb is far from a perfect estimator. Shorter active intervals may reduce energy usage but they also result in less water being pumped and thus may generate a deficit of water at the end of the scheduling period, or even pressure deficits during the scheduling period. Nonetheless, many ACO algorithms make use of some form of heuristic information, so it is worth testing whether it may also lead to improved results for pump scheduling. The results and conclusions of this empirical analysis are given in Section 6.3.3.

6.2.5 Pheromone Update Methods

After all ants have constructed solutions, the pheromone values are updated (Eq. 6.2). The pheromone increment $\Delta\tau$ can take many forms depending on the particular problem. It is often a function of the objective function itself, which in the pump scheduling problem is the electrical cost (C_{E}), such as in:

$$\Delta\tau(S) = \frac{Q}{C_{\mathrm{E}}(S)} \tag{6.8}$$

where Q could be a constant or the maximum for objective function ($C_{\mathrm{E}}^{\mathrm{max}}$). A simpler approach would be to use the same constant value for all ants (e.g., $\Delta\tau = 1$). Since only one ant at each iteration is allowed to update the pheromone values, and evaporation decreases the influence of previous pheromone updates at every iteration, this approach may be sufficient to achieve convergence to a good solution (Dorigo & Stützle, 2004). In the empirical analysis carried out in the following sections, we test and compare both update methods: $\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\mathrm{max}}}{C_{\mathrm{E}}(S)}$ and $\Delta\tau = 1$.

6.2.6 Constraint Handling

The constraint handling used for ACO in the Pump Scheduling problem is identical to the one used for SEA in Chapter 4. As a brief reminder, hydraulic constraints and limits on tank levels are enforced implicitly by EPANET. The constraint on the number of pump switches is implicitly satisfied by the time-controlled triggers representation. The rest of the constraints (zero total volume deficit, minimum pressure requirements at demand nodes and no warnings from the simulator) are handled by ranking solutions according to which constraints are violated and the degree of violation. That is, a solution is considered better than another if: (*i*) it generates lower pressure violations; if equal, (*ii*) it resulted in less simulation warnings; if still equal (*iii*), it produces smaller volume deficit. In case all constraint violations are equal, as would happen when comparing two feasible solutions, they are compared with respect to their objective function values.

6.3 Empirical Study of ACO in the Pump Scheduling Problem

In the following sections we perform an empirical analysis of the performance of the Ant System (AS) algorithm on the Pump Scheduling problem. The experimental setup tests different parameters of AS for both Vanzyl and Richmond networks. In order to assess the average performance of the AS algorithm, 15 runs, with different random seeds, were conducted for the Richmond network with each configuration of parameters; whereas 25 runs were performed for the Vanzyl network, since the variability of results was higher for the Vanzyl network and the extra runs were computationally cheap with respect to a single run for the Richmond network. In order to enable a fair comparison with results provided by van Zyl, Savic & Walters (2004), we stopped each run of the algorithm after 6000 evaluations (i.e., hydraulic simulations) in the case of the Vanzyl network and 8000 evaluations for the Richmond network. A limit of $SW = 3$ switches per pump is set and we evaluate both strict ($N_p^{\mathrm{sw}} = 3$) and relaxed ($N_p^{\mathrm{sw}} \leq 3$) representations. The use of heuristic information will be considered later, so for now $\alpha = 1$ and $\beta = 0$ in Eq. (6.1). Also later we will consider the Max-Min Ant System, which indicates how to set the initial pheromone value, so for now we will use $\tau_0 = 1$. A summary of the parameters considered in the experimental setup and their values is given in Table 6.1. The results of these experiments are presented in Tables B.1 and B.2 for the Vanzyl and Richmond networks, respectively.

Table 6.1: Parameter setup for Ant System experiments.

Parameter	Value
Repetitions	25 (Vanzyl network), 15 (Richmond network)
Stopping criteria (Evaluations)	6000 (Vanzyl network), 8000 (Richmond network)
Persistence factor (ρ)	0.85, 0.9, 0.95, 0.98
Number of ants (A)	10, 20, 40, 80
Selection method (S^{best})	iteration-best (ib), best-so-far (bf)
Update method ($\Delta\tau$)	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S), \quad \Delta\tau = 1$
Limit of switches per pump (N_p^{sw})	$N_p^{\mathrm{sw}} = 3$, $N_p^{\mathrm{sw}} \leq 3$
Other parameters	$\tau_0 = 1$, $\alpha = 1$, $\beta = 0$

The empirical analysis is divided in three parts. First, we investigate whether there is any significant difference between the two alternative update methods discussed in Section 6.2.5. Then, we examine the effect of each parameter and their potential interactions. Finally, we perform additional experiments on a few selected configurations to explore whether the use of heuristic information further improves the best results obtained.

6.3.1 Choosing a Pheromone Update Method

In Section 6.2.5, we discussed two alternatives for the parameter $\Delta\tau$ used for updating the pheromones: $\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\max}}{C_{\mathrm{E}}(S)}$ and $\Delta\tau = 1$. Both alternatives were tested in our experiments and statistically compared. Tables 6.2 and 6.3 compare, for the Vanzyl and Richmond networks respectively, the two update methods in terms of the best configurations of parameters. On the other hand, Table 6.4 makes a comparison with respect to all configurations of parameters tested.

Table 6.2 shows the configuration of parameters that obtained the lowest median electrical cost for each update method from all the results in the Vanzyl network. We report separately the results when using a strict constraint on the pump switches ($N_p^{\mathrm{sw}} = 3$) and a relaxed constraint ($N_p^{\mathrm{sw}} \leq 3$). We also show the p-value obtained by a Welch two-sample t-test comparing the best configurations of parameters for each of the two update methods. Table 6.3 gives the same information for the Richmond network. The conclusion from the t-tests is that there is no statistically significant difference between the two update methods in the case of the best configuration of parameters for each update method.

Table 6.2: Comparison of the best configurations of parameters for each pheromone update method for the Vanzyl network.

	Pheromone update	Best settings	p-value (t-test)
$N_p^{\mathrm{sw}} = 3$	$\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\max}}{C_{\mathrm{E}}(S)}$	$A = 80$, best-so-far, $\rho = 0.95$	0.2082
	$\Delta\tau = 1$	$A = 80$, best-so-far, $\rho = 0.98$	
$N_p^{\mathrm{sw}} \leq 3$	$\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\max}}{C_{\mathrm{E}}(S)}$	$A = 80$, best-so-far, $\rho = 0.98$	0.9725
	$\Delta\tau = 1$	$A = 40$, best-so-far, $\rho = 0.98$	

Table 6.3: Comparison of the best configurations of parameters for each pheromone update method for the Richmond network.

	Pheromone Update	Best Settings	p-value (t-test)
$N_p^{\mathrm{sw}} = 3$	$\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\max}}{C_{\mathrm{E}}(S)}$	$A = 40$, iteration-best, $\rho = 0.90$	0.358
	$\Delta\tau = 1$	$A = 80$, iteration-best, $\rho = 0.85$	
$N_p^{\mathrm{sw}} \leq 3$	$\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\max}}{C_{\mathrm{E}}(S)}$	$A = 80$, iteration-best, $\rho = 0.85$	0.8083
	$\Delta\tau = 1$	$A = 20$, iteration-best, $\rho = 0.90$	

If we consider the full results (all configurations of parameters) for each update method, we obtain p-values much higher than a critical level of 0.05 (see Table 6.4). Moreover, 95% confidence intervals of the difference in mean electrical cost obtained by the two update methods give very small extreme values. Therefore, we must conclude that for this problem there is little difference (if any) between using one or another method. In the following we will use $\Delta\tau = 1$, since it is a much simpler approach.

Table 6.4: 95% confidence interval and p-value reported by Welch two-sample t-test on the difference in mean electrical cost obtained when using update methods $\Delta\tau = 1$ versus $\Delta\tau(S) = \frac{C_{\mathrm{E}}^{\max}}{C_{\mathrm{E}}(S)}$ (the former minus the latter), over all configurations of parameters.

	$N_p^{\mathrm{sw}} = 3$		$N_p^{\mathrm{sw}} \leq 3$	
	CI	p-value	CI	p-value
Vanzyl	$[-0.5, 1.69]$	0.2866	$[-1.37, 0.60]$	0.447
Richmond	$[-0.92, 0.78]$	0.8695	$[-0.39, 1.69]$	0.2202

6.3.2 Analysis of AS Parameters

Following the conclusions from the previous section, we focus on the results obtained with constant pheromone update $\Delta\tau = 1$ and we want to investigate the effect and possible interactions between the other AS parameters, namely the number of ants A, the persistence parameter ρ and the selection method. We analyse the results by means of the Analysis of Variance (ANOVA). In terms of ANOVA, each of these parameters is considered a factor and their tested values, given in Table 6.1, are levels. We perform ANOVA for each instance and each constraint on the pump switches, to assess the effect of each parameter (or interaction thereof) in the electrical cost obtained by AS.

The Vanzyl Network

For the Vanzyl network and constraint $N_p^{\mathrm{sw}} \leq 3$, ANOVA identifies significant interactions between all parameters. We examine each interaction in detail by means of interaction plots and error bars based on Tukey's Honest Significant Difference intervals in Fig. 6.3. The conclusion of the analysis is that best results are obtained with either a high number of ants or high ρ (low evaporation), while the difference between iteration-best and best-so-far is not statistically significant.[1]

[1] It is relevant to mention that had we started the analysis with a low number of ants or persistence factor and decided first which selection method was better, we would had drawn the false conclusion that iteration-best produces the best results, when in fact that strongly depends on the value of the other parameters.

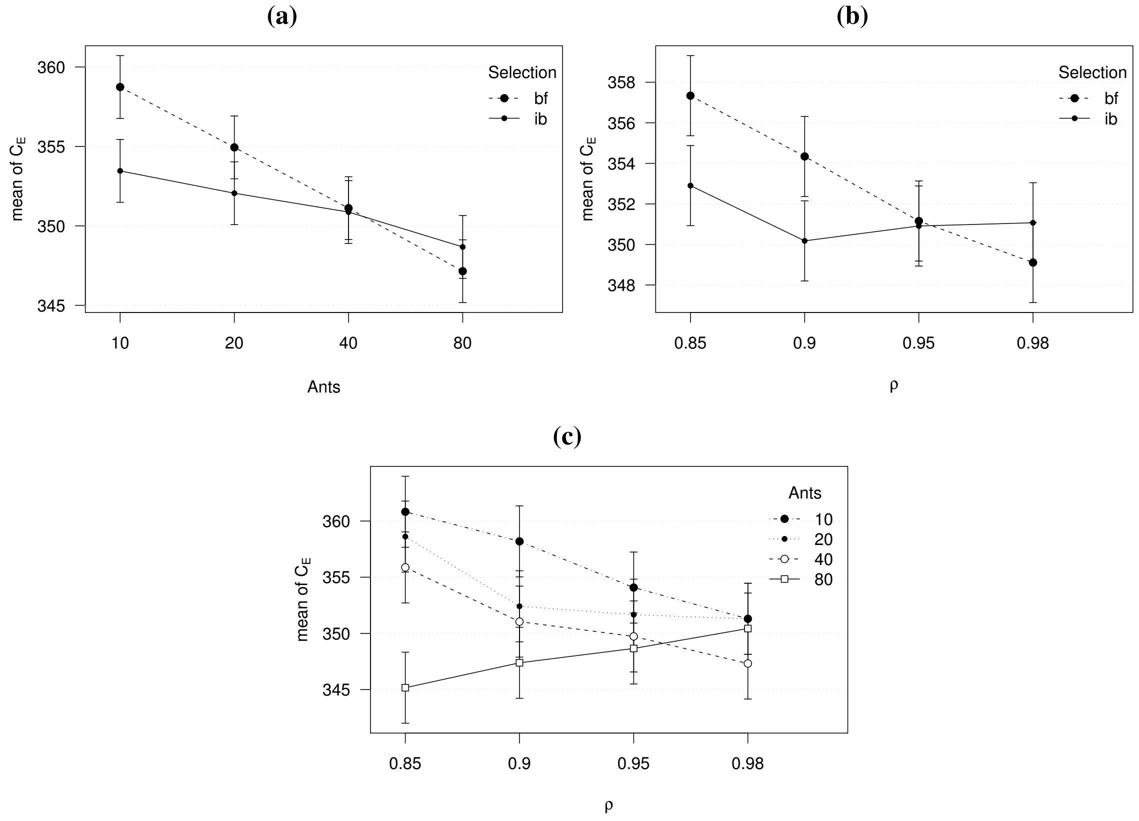


Figure 6.3: Interaction plots of AS using $N_p^{\text{sw}} \leq 3$ for the Vanzyl network.

Number of ants (A) and selection method. Selecting for update the iteration-best ant seems to give better results when the number of ants is low, as shown in Fig. 6.3a. However, for high number of ants, which is the setting that obtains the best results, the difference between iteration-best and best-so-far is not significant.

Persistence factor (ρ) and selection method. Figure 6.3b shows that there is no significant difference between the two selection methods for high values of ρ, while best-so-far performs much worse than iteration-best when combined with low values of ρ (high evaporation). More importantly, results improve steadily for higher values of ρ when using best-so-far selection.

Persistence factor (ρ) and number of ants (A). Fig. 6.3c shows that the value $\rho = 0.85$ produces better results when combined with $A = 80$ than for the other values of A. However, this anomaly is unimportant in order to find the best configuration, since Fig. 6.3c also shows that there is no significant difference with respect to other values of ρ. This suggests that the effect of ρ is not important when using high number of ants. Conversely, the effect the number of ants is not important for high values of ρ.

The above conclusions are consistent with the results for $N_p^{\text{sw}} \leq 3$ and $\Delta\tau = 1$ given in Table B.1, where configurations of parameters with either $\rho = \{0.95, 0.98\}$ or $A = \{40, 80\}$ produce the best results. This suggests that a higher exploration, which is obtained by using more ants and low evaporation, works better in this case than a faster convergence. Among these best configurations of parameters, we choose for further comparison the configuration that achieves the lowest median electrical cost, which is obtained by $A = 40$, best-so-far, $\rho = 0.98$. Similar conclusions are obtained by applying ANOVA to the results for the Vanzyl network and constraint $N_p^{\text{sw}} = 3$. In this case, the lowest median electrical cost was obtained by using $A = 80$, best-so-far and $\rho = 0.98$.

The Richmond Network

The results of ANOVA for the data corresponding to the Richmond network when using constraint $N_p^{\text{sw}} \leq 3$ also indicate significant interactions for all combinations of factors. The analysis shows that the best results are obtained when best-so-far selection is combined with a high number of ants or when iteration-best selection is combined with a low ρ. However, the analysis also suggests that the best approach is low ρ and high number of ants. To understand this apparent contradiction, we study the third-order interaction. This reveals that, overall, iteration-best and low values of ρ lead to the best results. We start the analysis studying the interactions between each pair of parameters, as shown in Fig. 6.4.

Number of ants (A) and selection method. The interaction between these two parameters is evident in Fig. 6.4a. The combination of either best-so-far and high number of ants, or iteration-best and low number of ants, give the best results.

Persistence factor (ρ) and selection method. In a similar fashion, Fig. 6.4b shows the strong interaction between evaporation and selection method. However, in this case, it is clear that iteration-best and low ρ is better than the combination of best-so-far and high ρ.

Persistence factor (ρ) and number of ants (A). Figure 6.4c shows that high $\rho = 0.98$ does not give good results for any number of ants and it is specially worse for $A = 80$. On the other hand, the results for $\rho = 0.85$ improve with increasing number of ants. This result suggests that the best approach is low ρ and high number of ants. However, the previous interaction plots showed high number of ants combines best with best-so-far selection, while low ρ performs better when combined with iteration-best. In order to reach a final conclusion, we must consider also the third-order interaction.

In ANOVA, a third-order interaction is an interaction between three factors. To study such interaction in our case, we plot the interaction between A and ρ for each selection method, as shown in Fig. 6.5.

Persistence factor (ρ) and number of ants (A) and best-so-far selection. Figure 6.5a shows that $A = \{10, 20\}$ are worse than $A = 80$ for low values of ρ, while $A = 80$ is worse for $\rho = 0.98$. However, apart from these worst-cases, most intervals overlap, that is, the overall effect of evaporation and number of ants is small when using best-so-far selection and there is no clear best configuration.

Persistence factor (ρ) and number of ants (A) and iteration-best selection. In contrast, Fig. 6.5b shows a strong influence of ρ when using iteration-best selection and low values of ρ achieve the best results. In contrast, the influence of the number of ants is small, since most intervals for different numbers of ants overlap.

Taking the above analysis into account, it is not surprising that the lowest median electrical cost in the Richmond network was obtained by $A = 20$, $\rho = 0.90$ and iteration-best. Yet, the analysis showed (and Table B.2 corroborates) that other configurations of parameters using iteration-best selection and $\rho = \{0.85, 0.90\}$ generate results that are not significantly different from the best configuration. Similar conclusions are obtained for the ANOVA of results corresponding to the Richmond network and using constraint $N_p^{\text{sw}} = 3$. In this case, the lowest median electrical cost is achieved by $A = 80$, $\rho = 0.85$ and iteration-best, but other combinations of iteration-best selection and high evaporation (low ρ) produce equally good results for various values of A.

Summary of Analysis of AS Parameters

The overall conclusions of the analysis of AS parameters are that for the Vanzyl network the best strategy is to use a large number of ants and high ρ (low evaporation), which suggests a higher exploration and slow convergence is needed to achieve the best results. Using iteration-best or best-so-far selection is not very important here, which may indicate that the iteration-best ant frequently becomes the best-so-far ant when using such combination of parameters. Completely different conclusions are obtained for the Richmond network. Here the selection method clearly influences the performance, and the combination of iteration-best selection and low values of ρ turns out to be the best configuration. Iteration-best selection implies exploration of the search space. However, this exploration probably needs to be counter-balanced with the faster convergence produced by a low value of ρ in order to produce good results within the limited number of evaluations.

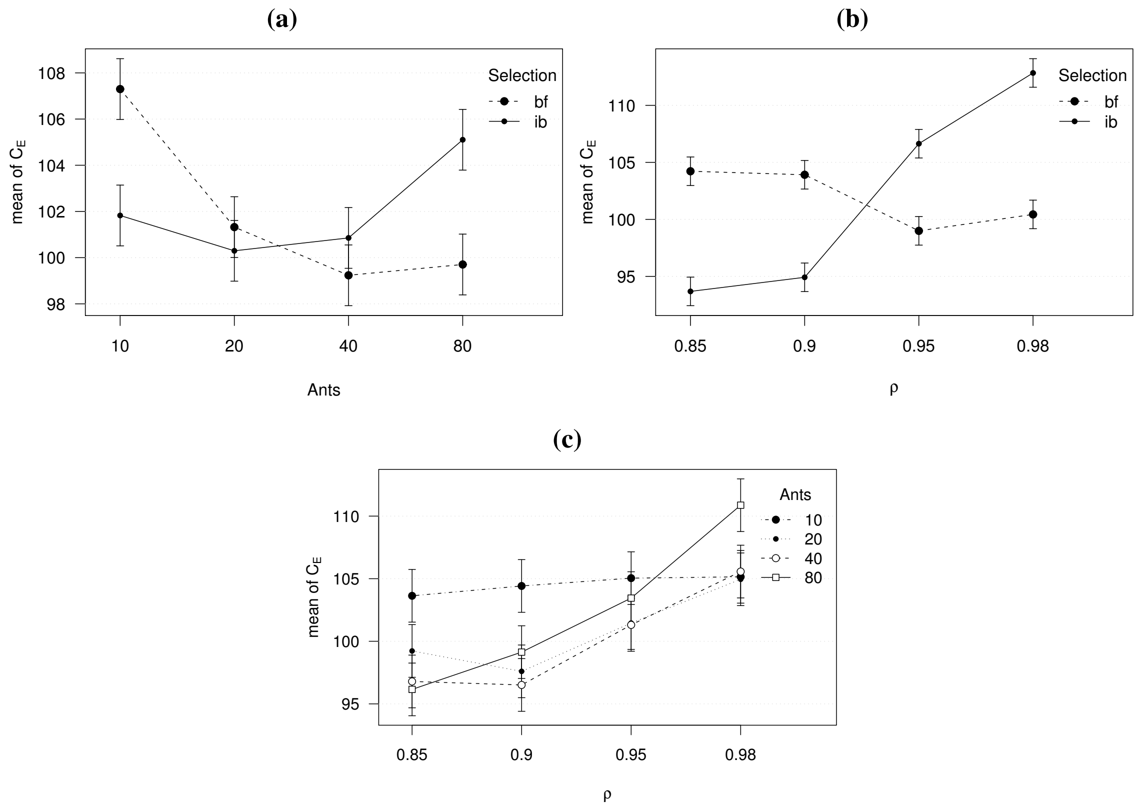


Figure 6.4: Second-order interaction plots of AS using $N_p^{\text{sw}} \leq 3$ for the Richmond network.

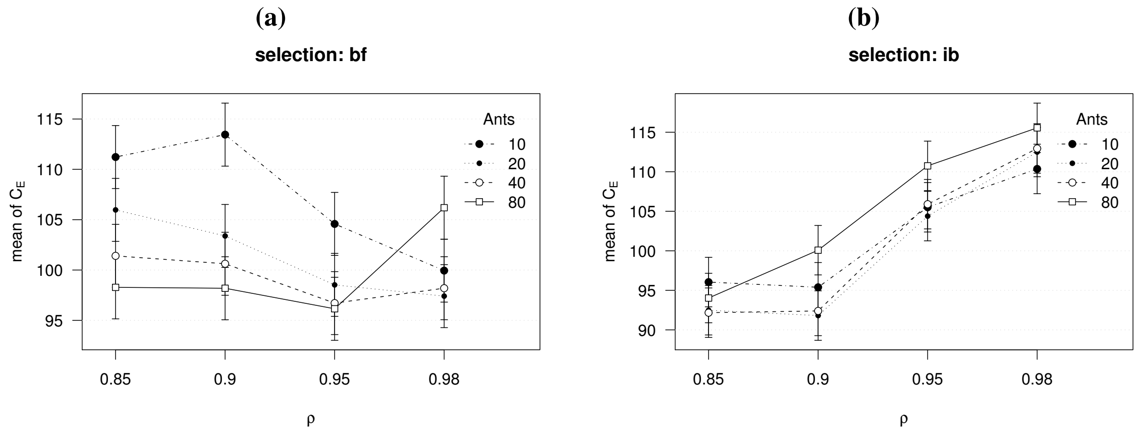


Figure 6.5: Third-order interaction plots of AS using $N_p^{\text{sw}} \leq 3$ for the Richmond network.

These differences can be attributed to the structure of the network instances. The Vanzyl network has only three pumps compared to the seven pumps of the Richmond network. Furthermore, the pumps in the Vanzyl network have very similar characteristics (in fact, two of them are parallel), which generates a search space with multiple solutions of similar cost. This may actually increase the difficulty of finding the global optimal solution since there is no clear direction of improvement and, thus, algorithms easily find themselves trapped in a sub-optimal solution. While on the contrary, the Richmond network contains different types of pumps located in various positions in the network. This reduces the number of different solutions with similar cost.

6.3.3 Heuristic Information

In this Section we evaluate the benefits of incorporating the heuristic information described in Eq. (6.7). The influence of the heuristic information is controlled with the parameter β. The higher the value, the largest the influence on the decision taken by the ants. We test the best configurations of parameters found in the previous analysis using different values of $\beta = \{0, 0.25, 0.5, 0.75, 1\}$, where $\beta = 0$ corresponds to not using any heuristic information at all. Results are reported for each constraint on pump switches in Figures 6.6 and 6.7, for the Vanzyl and Richmond networks respectively. Apart from the electrical cost, we pay attention as well to the evaluation number when the best solution of each run was found. The results with respect to the electrical cost indicate that heuristic information does not show any clear improvement for these configurations of parameters. Looking at the evaluation number, it does not seem that the use of heuristic information stagnates the search. In fact, heuristic information helps exploration (best

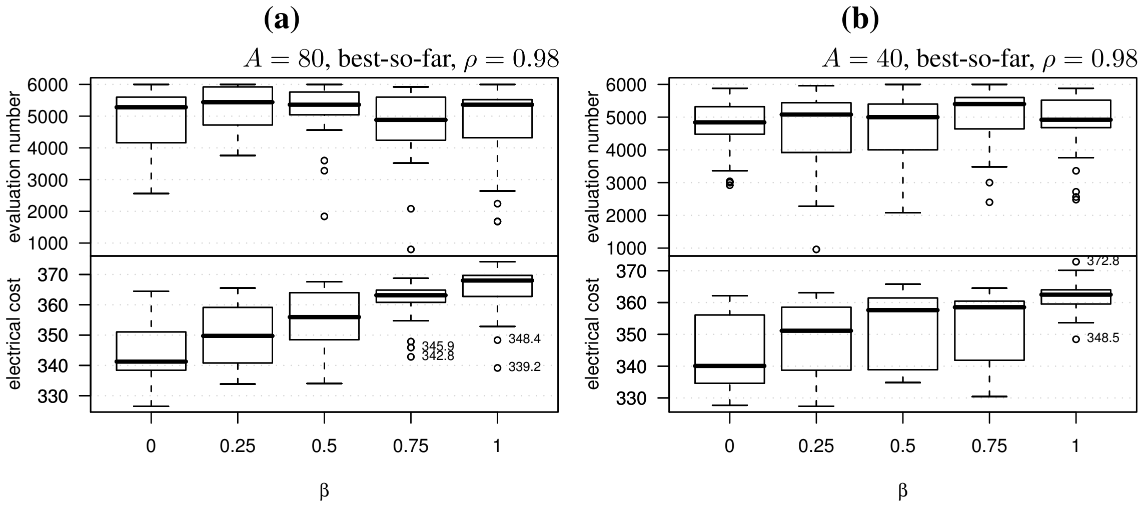


Figure 6.6: Influence of heuristic information (β) in the best configurations of parameters for the Vanzyl network (a) when $N_p^{\mathrm{sw}} = 3$ and (b) when $N_p^{\mathrm{sw}} \leq 3$.

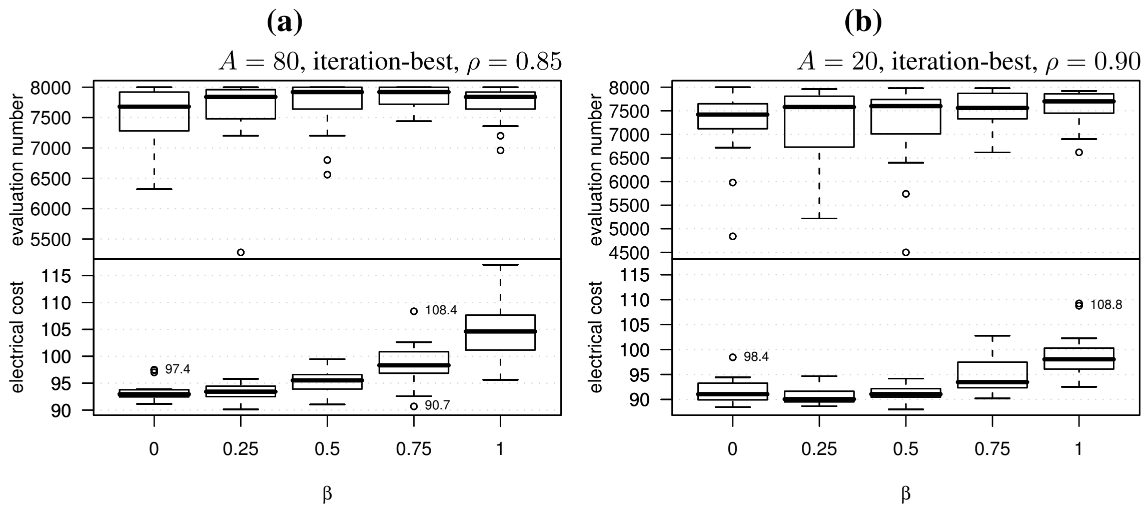


Figure 6.7: Influence of heuristic information (β) in the best configurations of parameters for the Richmond network (a) when $N_p^{\mathrm{sw}} = 3$ and (b) when $N_p^{\mathrm{sw}} \leq 3$.

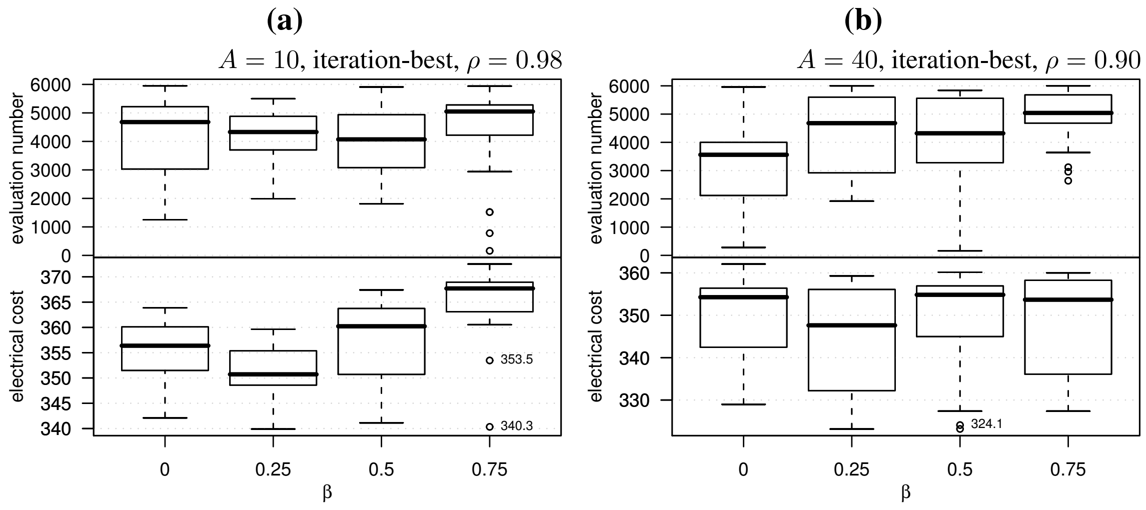


Figure 6.8: Influence of heuristic information (β) for the Vanzyl network (a) when $N_p^{\mathrm{sw}} = 3$ and (b) when $N_p^{\mathrm{sw}} \leq 3$.

solution is improved until close to the limit of evaluations) but it is not as effective as the pheromone information in directing the search to the optimal solutions. Still, heuristic information seems to slightly help the search with $\beta = 0.25$ in the Richmond network when using the relaxed constraint $N_p^{\mathrm{sw}} \leq 3$ (see Fig. 6.7). In addition, when not using the best configuration of parameters, the use of heuristic information, again with $\beta = 0.25$, may clearly improve the performance of the AS algorithm, as shown in Fig. 6.8. However, for the best configurations of parameters, the use of heuristic information does not show any advantage.

6.3.4 Comparison among Algorithms

In the previous sections, we have analysed the effect of the different parameters of AS and identified configurations of parameters that give good results for the pump scheduling problem. Now, we compare the best results obtained by AS with the state-of-the-art results from the literature, the Hybrid Genetic Algorithm (Hybrid GA) described by van Zyl, Savic & Walters (2004), and with the best results obtained by the Simple Evolutionary Algorithm (SEA) proposed in Chapter 4. Both AS and SEA are fine-tuned for relative time-controlled triggers representation, whereas Hybrid GA was designed for level-controlled triggers representation.

The Vanzyl Network

Table 6.5 shows a comparison of four algorithms according to the *median*, standard deviation (*sd*), *best* and *worst* values of the daily electrical cost (C_{E}) and total number of pump switches (N^{sw}). The algorithms compared are the AS algorithm discussed above with both strict and relaxed constraint on the number of pump switches, the Simple Evolutionary Algorithm (SEA) and the results of Hybrid Genetic Algorithm (Hybrid GA) reported by van Zyl, Savic & Walters (2004).Figure 6.9 graphically displays the comparison according to electrical cost.

Comparing the median electrical cost and the number of pump switches obtained by AS with constraint $N_p^{\text{sw}} = 3$ and constraint $N_p^{\text{sw}} \leq 3$ reveals that relaxing the constraint on the number of switches per pump reduces notably the total number of switches, although at the expense of more variation in the results (higher standard deviation, bigger box in Fig. 6.9). This could be attributed to the larger solution space explored when $N_p^{\text{sw}} \leq 3$ was used. The results obtained by AS compare well with those obtained by van Zyl, Savic & Walters (2004) using Hybrid GA, even though neither AS nor SEA use any local search method. Moreover, unlike the Hybrid GA, our AS approach did not require additional runs to tune the penalty costs. Still, SEA seems the clear winner among the four algorithms, in particular when compared with Hybrid GA, since the worst solution of SEA is better than the best solution of Hybrid GA.

Although AS obtained a lower median cost than Hybrid GA (in fact, the median of AS is below the best value of Hybrid GA), AS also obtained solutions with higher cost than the worst of Hybrid GA. This high variability of AS could mean that the observed differences are due to the stochastic nature of both algorithms. Therefore, we conducted a statistical analysis to assess whether there is a significant difference between the algorithms. First, we used a Kruskal-Wallis test to assess whether the median is statistically the same in all the algorithms or it is significantly different for at least one. As was ex-

Table 6.5: Experimental results for the Vanzyl network.

	C_{E}				N^{sw}			
Algorithm	med.	sd.	best	worst	med.	sd.	best	worst
AS ($N_p^{\mathrm{sw}} = 3$)	341.3	9.7	326.5	364.5	9	0	9	9
AS ($N_p^{\mathrm{sw}} \leq 3$)	340.1	11.5	327.7	362.1	4	1.3	3	8
Hybrid GA	347.1	4.3	344.4	354.8	4	0.8	3	5
SEA	334.1	6.1	315.9	341.4	5	1.2	3	7

AS ($N_p^{\mathrm{sw}} = 3$): $A = 80$, best-so-far, $\rho = 0.98$, $\Delta\tau = 1$, $\alpha = 1$, $\beta = 0$.
AS ($N_p^{\mathrm{sw}} \leq 3$): $A = 40$, best-so-far, $\rho = 0.98$, $\Delta\tau = 1$, $\alpha = 1$, $\beta = 0$.
SEA: relative time-controlled triggers with $N_p^{\mathrm{sw}} \leq 3$, see also Table 4.1 on page 66.
Hybrid GA: statistics are calculated from the 7 runs reported by van Zyl, Savic & Walters (2004).

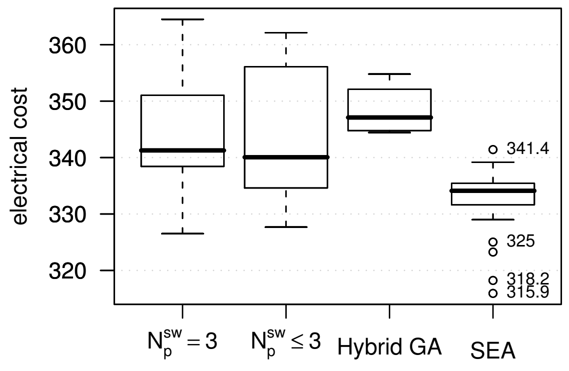


Figure 6.9: Comparison of AS with constraints ($N_p^{\mathrm{sw}} = 3$) and ($N_p^{\mathrm{sw}} \leq 3$), Hybrid GA and SEA for the Vanzyl network.

Table 6.6: Statistical comparison of algorithms for the Vanzyl network. P-values reported by pairwise Wilcoxon rank sum tests, adjusted for multiple comparison using Holm's method.

	AS $N_p^{\mathrm{sw}} = 3$	AS $N_p^{\mathrm{sw}} \leq 3$	Hybrid GA
AS $N_p^{\mathrm{sw}} \leq 3$	0.7124	—	—
Hybrid GA	0.2497	0.5478	—
SEA	3.4e-05	0.0038	3.6e-06

pected, the p-value $= 1.054e - 05$ indicates that there is a difference. Next, to identify which algorithms have a significantly different median, we use pairwise Wilcoxon rank sum tests.[2] The resulting p-values are reported in Table 6.6. The conclusions are that SEA is clearly different from the three other algorithms, while there is no statistical evidence to reject the hypothesis that the median electrical cost is equal for AS and Hybrid GA (at a confidence value of 0.05).

Finally, in order to illustrate the structure of schedules obtained, Fig. 6.10 shows three different solutions obtained by the AS algorithm for the Vanzyl network using constraint $N_p^{\text{sw}} = 3$ on the left and three schedules obtained using constraint $N_p^{\text{sw}} \leq 3$ on the right. In both cases, the best solutions take full advantage of the off-peak electricity tariff. It is also interesting that the best schedule obtained by AS ($N_p^{\text{sw}} \leq 3$) turns on the pumps at the start of the scheduling period, when tanks are almost full and pumping is supposed to be more expensive, which is a counter-intuitive result.

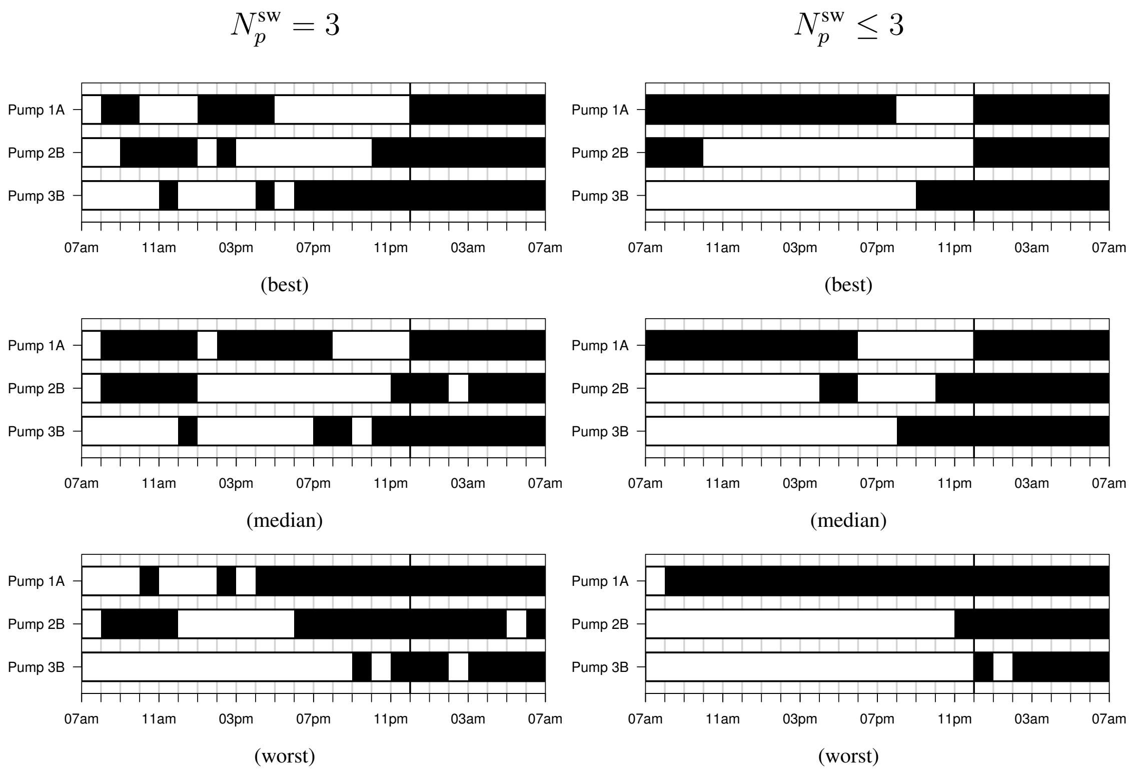


Figure 6.10: Pump schedules obtained by AS algorithm for the Vanzyl network: (top) best solution; (middle) median solution; and (bottom) worst solution.

[2] Kruskal-Wallis and Wilcoxon rank sum tests are standard non-parametric statistical tests, which do not require the data to be normally distributed (Furlong, Lovelace & Lovelace, 2000). Wilcoxon rank sum test is also called Mann-Whitney U test.

The Richmond Network

The comparison of the four algorithms for the Richmond network is given in Table 6.7 and graphically in Fig. 6.11. The algorithms compared are the Ant System algorithm proposed above with both a strict and a relaxed constraint on the number of pump switches, the Simple Evolutionary Algorithm (SEA) and the Hybrid Genetic Algorithm (Hybrid GA) proposed by van Zyl, Savic & Walters (2004). For Hybrid GA there are no numerical results available, so we estimated the electrical cost following the graphical results of 10 runs reported by the authors. Van Zyl, Savic & Walters (2004) do not provide any concrete information about the number of pump switches obtained by Hybrid GA in the Richmond network. However, we do not expect that the total number is significantly lower than the values obtained by AS ($N_p^{\text{sw}} \leq 3$) or SEA. On the contrary, the level-controlled triggers representation used by Hybrid GA tends to generate frequent switching of the pumps.

For the Richmond network, AS ($N_p^{\text{sw}} \leq 3$) not only reduces the number of total switches by a third with respect to AS ($N_p^{\text{sw}} = 3$), it also reduces the median electrical cost. The results of AS ($N_p^{\text{sw}} \leq 3$) are also better than those obtained by SEA in terms of both median electrical cost and total number of pump switches. Hybrid GA performs substantially worse than any of the other three algorithms. The worse electrical cost obtained by AS or SEA is almost always lower than the one corresponding to the best schedule generated by Hybrid GA. The differences are even more clearly shown in Fig. 6.11, where it can be observed that the worst case of AS ($N_p^{\text{sw}} = 3$) is actually an outlier and schedules generated by AS have a much lower electrical cost than the best schedule of Hybrid GA.

We repeat for the Richmond network the same statistical analysis performed above for the Vanzyl network. As expected, the Kruskal-Wallis test strongly rejects that the real median value is equal for the four algorithms. Pairwise Wilcoxon tests (see Table 6.8) indicate significant differences between all algorithms, except for SEA and AS ($N_p^{\text{sw}} = 3$). The conclusion, hence, is that AS with a relaxed constraint generates the best pump schedules of the four algorithms for the Richmond network, while the ones generated by Hybrid GA are definitely more expensive.

Now we examine the median pump schedules obtained by the AS algorithm in the Richmond network. The median schedule obtained when using the strict constraint $N_p^{\text{sw}} = 3$ is shown in the left plot in Fig. 6.12, while the one obtained using $N_p^{\text{sw}} \leq 3$ is shown in the right plot. Although both schedules seem quite similar, a careful examination shows that the off-peak period is better utilised in the right plot. Moreover, for pumps 1A and 7F, the schedules are almost similar except for a very short active interval at the start of the scheduling period. The presence of such short pumping intervals indicates that the strict constraint is forcing more switches than what would be optimal. The relaxed constraint

Table 6.7: Experimental results for the Richmond network.

	C_{E}				N^{sw}			
Algorithm	median	sd.	best	worst	median	sd.	best	worst
AS ($N_p^{\mathrm{sw}} = 3$)	92.9	2.1	91.1	97.5	21	0.0	21	21
AS ($N_p^{\mathrm{sw}} \leq 3$)	90.1	1.9	88.7	94.7	14	1.5	10	15
Hybrid GA	99.5	2.2	97.0	104.0	—	—	—	—
SEA	92.3	1.6	90.3	95.4	16	1.7	14	20

AS ($N_p^{\mathrm{sw}} = 3$): $A = 80$, iteration-best, $\rho = 0.85$, $\Delta\tau = 1$, $\alpha = 1$, $\beta = 0$.
AS ($N_p^{\mathrm{sw}} \leq 3$): $A = 20$, iteration-best, $\rho = 0.90$, $\Delta\tau = 1$, $\alpha = 1$, $\beta = 0.25$.
SEA: relative time-controlled triggers with $N_p^{\mathrm{sw}} \leq 3$, see also Table 4.2 on page 67.
Hybrid GA: energy costs are estimated from graphical results from van Zyl, Savic & Walters (2004).

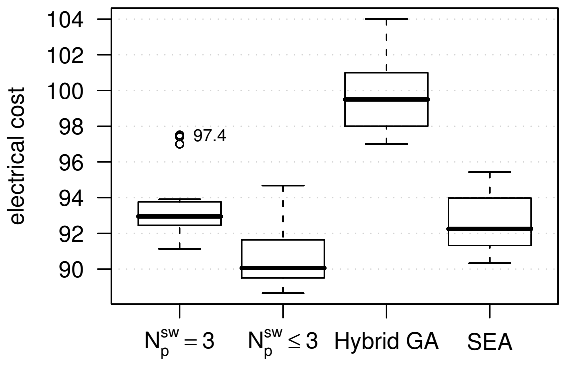


Figure 6.11: Comparison of AS with constraints ($N_p^{\mathrm{sw}} = 3$) and ($N_p^{\mathrm{sw}} \leq 3$), Hybrid GA and SEA for the Richmond network.

Table 6.8: Statistical comparison of algorithms for the Richmond network. P-values reported by pairwise Wilcoxon rank sum tests, adjusted for multiple comparison using Holm's method.

	AS $N_p^{\mathrm{sw}} = 3$	AS $N_p^{\mathrm{sw}} \leq 3$	Hybrid GA
AS $N_p^{\mathrm{sw}} \leq 3$	0.00118	—	—
Hybrid GA	0.00036	0.00021	—
SEA	0.20168	0.00988	0.00021

seems to handle well this situation and is able to generate schedules with just one switch (pump 1A) and others with 3 switches (pumps 3A and 5C).

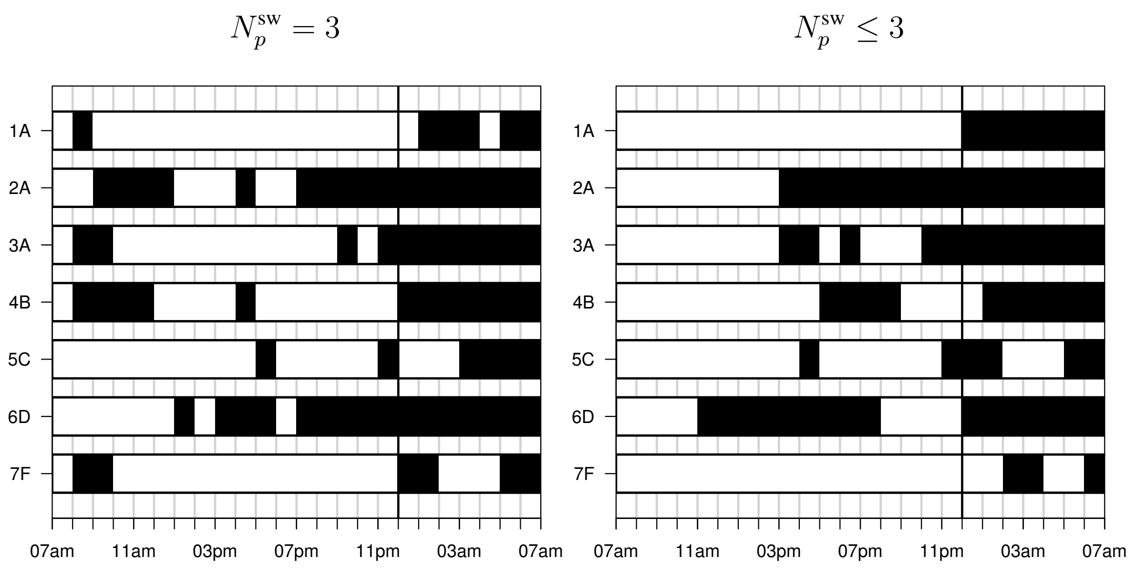


Figure 6.12: Pump schedules corresponding to the median solution obtained by AS algorithm for the Richmond network.

6.3.5 Discussion of AS Results

The results of the previous sections show the feasibility of using the ACO framework for the optimisation of pump schedules. We have covered the main aspects needed to implement an ACO algorithm for the pump scheduling problem, namely the representation of solutions, the definition of the pheromone information and the construction mechanism used by ants to generate new solutions. Update methods and heuristic information have been also discussed and analysed. An extensive empirical analysis of the Ant System algorithm has been carried out, examining in detail the effect of each parameter and their combined influence. By means of statistical testing, the best configurations of parameters have been identified. One clear conclusion of this analysis is that the Vanzyl and Richmond networks are fundamentally different instances, thus the best configuration on one of them has little resemblance with the best configuration for the other. As a matter of fact, using the Vanzyl network as a test network for fine-tuning parameters for the Richmond network will, in all certainty, lead to suboptimal configurations of parameters. It is not clear at this moment which kind of network instance could take that role, perhaps a simplification of the Richmond network itself. But it is evident that the Vanzyl network belongs to a different kind, worth of investigating by its own interest and inherent difficulty. Because it turns out that, for the ACO algorithm proposed here, the Vanzyl network seems to be more difficult than the Richmond network.

The final part of our analysis compared AS (both strict and relaxed constraint on the number of pump switches) with the Simple Evolutionary Algorithm proposed in earlier chapters and the Hybrid GA proposed by van Zyl, Savic & Walters (2004). AS with a relaxed constraint generated schedules with a much lower number of switches than when using a strict constraint. In the case of the Richmond network, the relaxed constraint produced lower cost schedules. Whereas in the case of the Vanzyl network there was no significant difference between using one or the other constraint in terms of median electrical cost. Therefore, it seems more beneficial to focus on AS with relaxed constraint.

Ant System obtained clearly the best results in case of the Richmond network, both in terms of number of pump switches and electrical cost. Even the worst solutions given by AS were better than the best solution obtained by Hybrid GA. In addition, statistical tests showed a significant difference between the medians of AS and SEA, in favour of the former. In the case of the Vanzyl network, however, the median electrical cost obtained by AS was not statistically different from the one of Hybrid GA. Moreover, AS for the Vanzyl network showed a high variability. In this network, SEA was clearly the winner. These results suggest that some characteristic of the Vanzyl network makes this instance particularly difficult for AS. In fact, pumps for the Richmond network have distinct features and locations, whereas pumps in the VanZyl network have similar features and nearby locations. The consequence of the latter is fundamentally different schedules with similar electrical cost.

One of the consequences of adapting the Pump Scheduling to the ACO framework rather than creating an ad hoc algorithm is that the proposed approach can be straightforwardly extended to more complex and modern ACO algorithms. In particular, the next sections study the application of Max-Min Ant System (Stützle & Hoos, 2000) with the goal of further improving the results obtained so far.

6.4 Max-Min Ant System

In the previous sections we have discussed the application of ACO to the pump scheduling problem. Empirical results showed that a variant of the Ant System algorithm generates schedules with lower electrical cost than those published in the literature. In the case of a real-world instance (the Richmond network), our AS algorithm also improved the results over those obtained by the Simple Evolutionary Algorithm (SEA) presented in Chapter 4. These are promising results, yet the AS algorithm examined in earlier sections is probably not the best ACO algorithm. In fact, literature shows that other ACO approaches typically outperform the original AS (Dorigo & Stützle, 2004). In this section we focus on the empirical application of Max-Min Ant System (MMAS), which is considered one of the best performing ACO algorithms (Dorigo & Stützle, 2004), with two goals in mind. First, showing by example that once the pump scheduling problem has been adapted to the ACO framework, the adaptation to new ACO algorithms is typically trivial. Our second goal is to investigate what performance improvements, if any, MMAS shows in the test instances.

Max-Min Ant System was proposed by Stützle & Hoos (2000) and its most notable characteristic is the use of maximum and minimum limits for pheromone values, which prevents extremely high or small pheromone values, and thus, avoids premature convergence. The maximum pheromone limit is inversely proportional to the evaporation factor $(1 - \rho)$ and directly proportional to the amount of pheromone added at each iteration ($\Delta\tau$). The minimum pheromone limit depends on the maximum pheromone limit and a parameter p_{best} which represents the probability of constructing the best solution found once MMAS has converged. For $p_{\text{best}} = 1$, the minimum pheromone limit is zero, whereas the difference between the maximum and minimum gets smaller with decreasing values of p_{best}. Another interesting aspect is that MMAS sets the initial pheromone values (τ_0) to the maximum pheromone limit in order to achieve a higher exploration. We refer to the original publication Stützle & Hoos (2000) for a complete description of how pheromone limits are calculated. Another characteristic of MMAS, widely used by other modern ACO algorithms, is that only a single ant is allowed to deposit pheromone at each iteration. This strategy was already used by the AS algorithm proposed in this study.

6.4.1 Empirical Setup

The empirical analysis of MMAS follows a setup similar to the one previously used for Ant System. We performed 25 runs of 6000 evaluations for the Vanzyl network and 15 runs of 8000 evaluations for the Richmond network. The parameters of the MMAS algorithm in this study are: the number of ants $A = \{10, 20, 40, 80\}$, the pheromone persistence parameter $\rho = \{0.85, 0.9, 0.95, 0.98\}$, the best ant used for selection is either the iteration-best (*ib*) or the best-so-far ant (*bf*), and $p_{\text{best}} = \{0.05, 0.5, 0.7, 0.9, 0.9999\}$. The initial pheromone was set to $\tau_0 = \tau_{\text{max}}$ while the amount of pheromone deposited by ants during pheromone updating was set to $\Delta\tau = 1$.

The previous analysis of AS showed that relaxing the constraint on pump switches does not prevent the algorithm from finding schedules with low electrical cost, while it has the additional benefit of reducing maintenance costs caused by frequent pump switching. Therefore, we focus solely on a relaxed constraint of less than three switches per pump ($N_p^{\text{sw}} \leq 3$) for both Vanzyl and Richmond networks. In the next two sections we analyse the results for both network instances.

6.4.2 Analysis of Results for the Vanzyl Network

The complete results for the Vanzyl network are shown in Table B.3. The ANOVA method used in the previous sections relies in some assumptions, one of them is the normality of the data, which is not satisfied in this case. Normality is typically tested by means of a Normal Q-Q plot, which displays points corresponding to the experimental data and a straight line representing the ideal normal distribution. If the experimental data resembles a normal distribution, the points should fall close to the straight line. Figure 6.13 shows the Normal Q-Q plot of the results of MMAS for the Vanzyl network. In this case, a substantial number of data points show a high deviation from the normal distribution. Sometimes transforming the data, for example, by taking the logarithm of the data, may alleviate this deviation. However, in this particular case, no transformation satisfyingly solves the lack of normality. Therefore, we are unable to apply the ANOVA method. Nevertheless, we still can observe some trends in the data by means of boxplots.

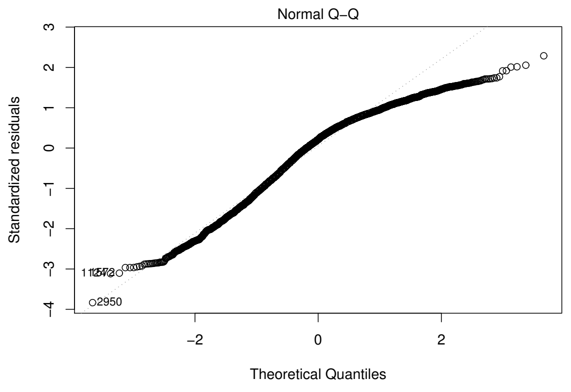


Figure 6.13: Normal Q-Q plot of MMAS results for the Vanzyl network

In the following plots, each boxplot summarises the results of 25 runs for a particular configuration of parameters. The plots summarise the electrical cost of the best schedule found by each run and the evaluation number at which each run found this solution. By keeping constant the value of all parameters except one, we can assess the effect of this parameter on the results.

Probability of constructing the best solution (p_{best}). Figure 6.14 shows the effect of p_{best} for a configuration of MMAS using best-so-far selection (Fig. 6.14a) and the same configuration but using iteration-best selection (Fig. 6.14b). Figure 6.14a shows how dramatically the performance degrades as p_{best} is increased. The parameter p_{best} controls the distance between the upper and lower pheromone limits, and as p_{best} approaches 1, the lower limit converges to zero, which is the minimum pheromone value. The explanation here is that best-so-far selection produces a fast convergence that may lead to stagnation, while low p_{best} (and tighter pheromone limits) favour exploration of new solutions. Stagnation is evident for $p_{\text{best}} = 0.999$, where the algorithm finds a schedule with electrical cost close to 360 in less than 1000 evaluations but it is not able to improve that solution thereafter. On the other hand, Fig. 6.14b shows a comparatively more uniform behaviour. Iteration-best selection is more exploratory, so it is not affected as much by the pheromone limits. However, the excess of exploration and lack of direction of the search also mean that this configuration is not able to obtain as good solutions as the combination of best-so-far selection and $p_{\text{best}} = 0.05$.

Persistence factor (ρ). Figure 6.15 illustrates that the value of ρ affects the resulting electrical cost. Moreover, the influence of ρ varies depending on the value of p_{best}. Figure 6.15a shows how for a low p_{best}, the performance degrades with an increased value of ρ. In contrast, Fig. 6.15b shows the opposite effect for a high p_{best}, except for $\rho = 0.98$, which obtains the worst results independently of the value of p_{best}. This demonstrates a strong interaction between ρ and p_{best}. High values of ρ indicate a low pheromone evaporation rate, which results in a slower convergence. When combined with tight pheromone limits (low p_{best}), which favour exploration of new solutions, the consequence is an excessively slow convergence. On the other hand, for high p_{best} (Fig. 6.15b), low values of ρ make the algorithm converge to a good solution relatively early in terms of evaluations and stagnate because of the wide pheromone limits, so higher values of ρ work better by slowing convergence. Yet, in the case of $\rho = 0.98$ the convergence is too slow and the search is not focused around the best solutions, despite the wide pheromone limits.

Number of ants (A). Finally, for some configurations of parameters, the number of ants have a strong influence on the results, whereas for others the influence is less clear. This is illustrated in Fig. 6.16. Since the number of evaluations is fixed, a higher number of ants means a lower number of iterations. If the pheromone evaporation is too slow ($\rho = 0.98$, Fig. 6.16a), the algorithm does not stagnate but solutions obtained do not improve fast enough because past solutions keep influencing current decisions. Therefore, less ants ($A = 10$) and more iterations produce better results. For a high evaporation rate ($\rho = 0.85$, Fig. 6.16b), this is not true anymore and the influence of the number of ants is not evident.

According to the above analysis, the best configurations of parameters are those using best-so-far selection. Furthermore, for low number of ants $A = 10$, the best approach is low evaporation $\rho = 0.9$ and low $p_{\text{best}} = 0.05$. These settings favour a slow convergence and more exploration, while the low number of ants enables the MMAS algorithm to

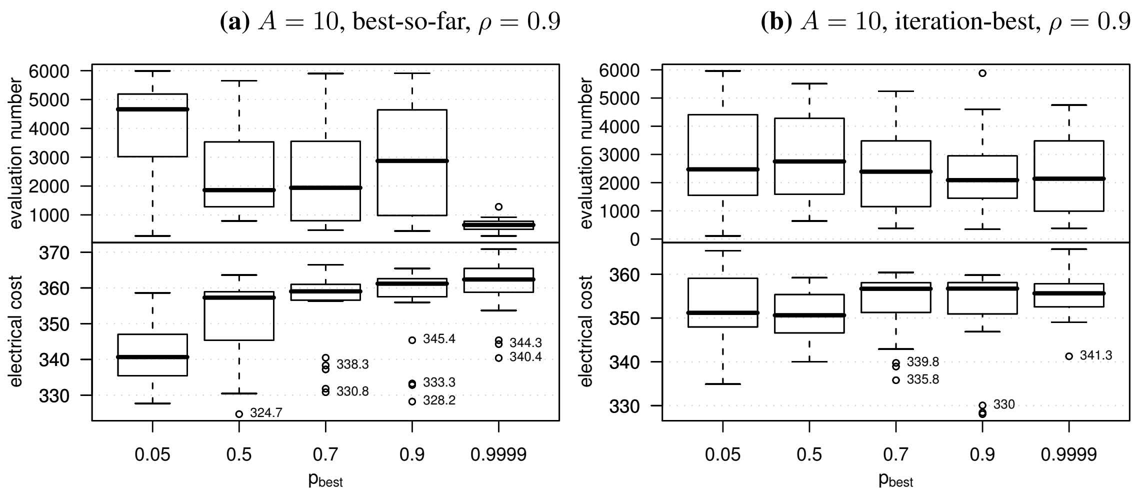


Figure 6.14: Effect of p_{best} in the results of MMAS for the Vanzyl network.

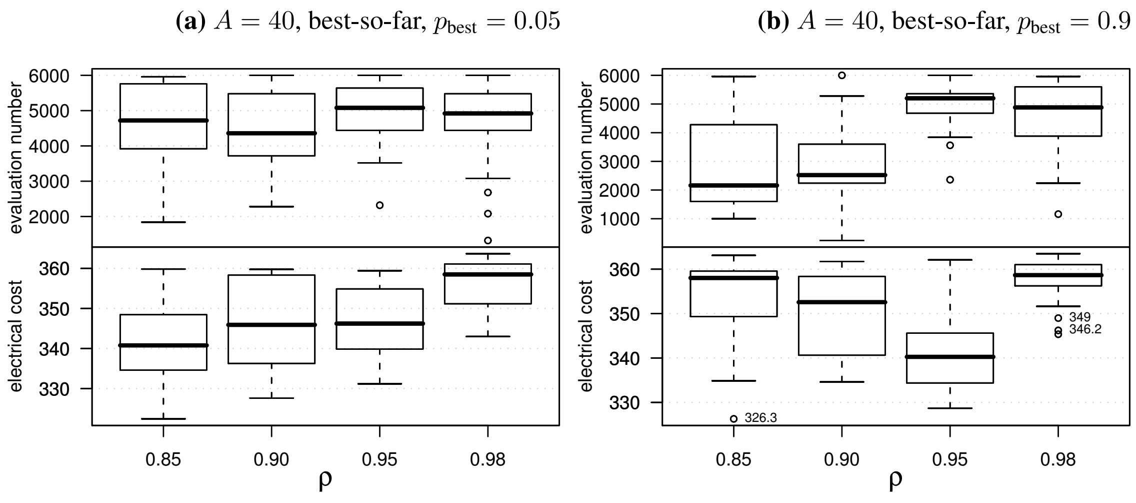


Figure 6.15: Effect of ρ in the results of MMAS for the Vanzyl network.

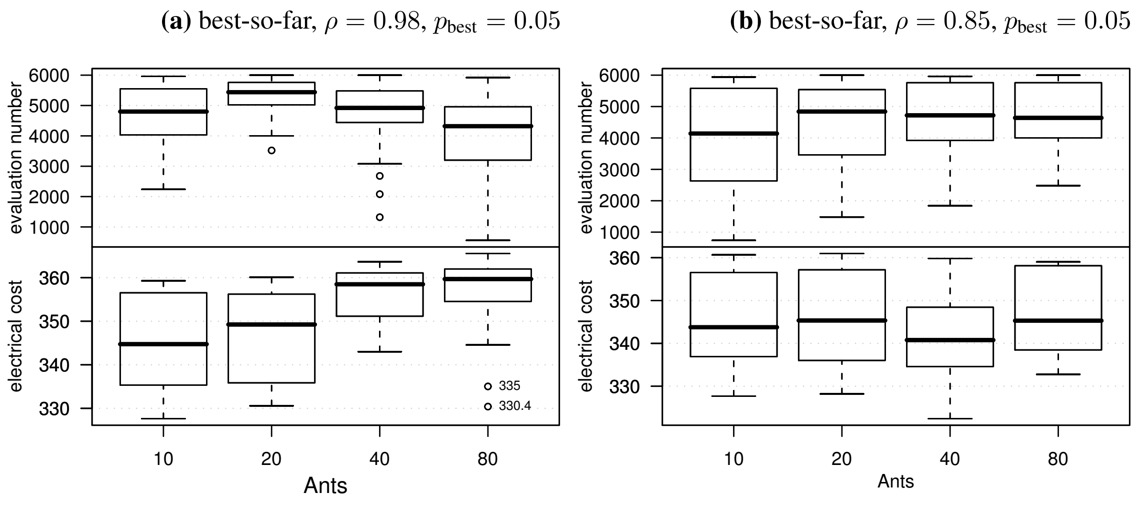


Figure 6.16: Effect of the number of ants in the results of MMAS for the Vanzyl network.

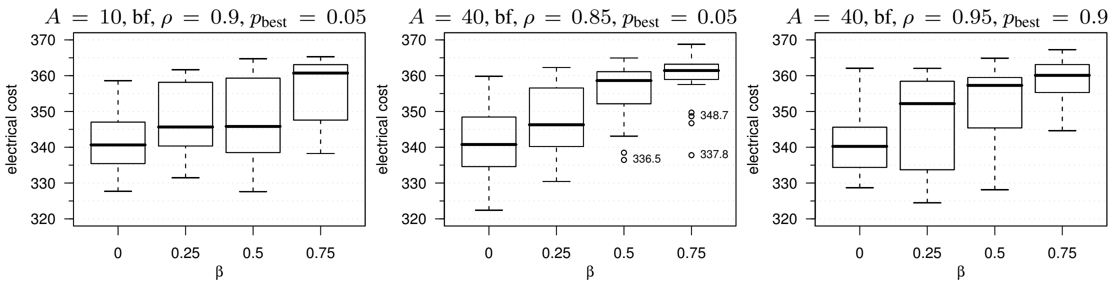


Figure 6.17: Influence of heuristic information (β) in several configurations of parameters for the Vanzyl network.

perform more iterations. By contrast, for high number of ants ($A = 40$), it is better to use either low evaporation $\rho = 0.95$ and high $p_{\text{best}} = 0.9$, which gives a balance between slow convergence but less exploration, or high evaporation $\rho = 0.85$ and low $p_{\text{best}} = 0.05$, which results in faster convergence but high exploration. Although the configuration of parameters that obtains the lowest median electrical cost is $A = 10$, best-so-far, $\rho = 0.9$ and $p_{\text{best}} = 0.05$, these three configurations produce similarly good results according to Table B.3 on page 183. Therefore, we will consider the three of them when studying the effect of heuristic information next, in case any of them is affected more strongly by the heuristic information than the others.

Heuristic Information

We now test whether heuristic information could further improve the results of the three configurations of MMAS identified above as the best ones for the Vanzyl network, The heuristic function is the one described in Section 6.2.4 and its intensity is controlled by the parameter β in Eq. (6.1). Experimental results for different values of β are shown in Fig. 6.17. As it happened before with the AS algorithm, setting $\beta = 0$ gives the best results in the case of the Vanzyl network, thus the use of heuristic information does not improve the results of fine-tuned parameters.

Comparison among Algorithms

Table 6.9 (and the corresponding Fig. 6.18) compares the Ant System (AS) algorithm proposed earlier in this chapter, the Max-Min Ant System, the Simple Evolutionary Algorithm proposed in Chapter 4 and the Hybrid Genetic Algorithm proposed by van Zyl, Savic & Walters (2004). The results obtained by MMAS are clearly better in the average and best-case than those obtained using Hybrid GA but not as good as SEA. Our conclusion is that our ACO algorithms (both AS and MMAS) in the Vanzyl network have difficulties to converge fast to the best solutions, yet if exploration is sacrificed to speedup convergence, then they have a tendency to stagnate in sub-optimal solutions with a cost in the range of $(350, 365)$. Still, Fig. 6.18 shows that the “box” corresponding to MMAS is below the line that crosses the “box” corresponding to Hybrid GA. This means that more than 75% of the runs of MMAS obtained a lower electrical cost than the median cost obtained by Hybrid GA.

In order to test whether the differences observed in Fig. 6.18 are statistically significant, we first use Kruskal-Wallis test, which rejects the hypothesis that the medians of the four algorithms are all equal with a p-value < 0.0001. Next, we perform Wilcoxon rank sum tests for all pairs. The results shown in Table 6.10 indicate that: (1) the median cost obtained by SEA is significantly different than the median cost obtained by other algorithms, but (2) we cannot reject the hypothesis that AS, MMAS and Hybrid GA obtain the same median electrical cost. If we had to choose one among the three, we would select MMAS, given the lower empirical median with respect to Hybrid GA and smaller variability and lower worst case with respect to AS. SEA, nonetheless, is clearly better than the other three algorithms.

6.4.3 Analysis of Results for the Richmond Network

We now examine the results of MMAS for the Richmond network. The complete results are summarised in Table B.4. We again rely on the ANOVA method to identify which parameter values have a significant effect on the results. However, we can make one conclusion before even applying ANOVA. By studying the results with respect to the values of each parameter, we observe that for persistence factor (ρ), the value $\rho = 0.98$ typically generates much worse results that the other values of ρ (Fig. 6.19). If we eliminate this particular parameter value from the analysis before applying ANOVA, we can better focus on the significance of the other parameter values.

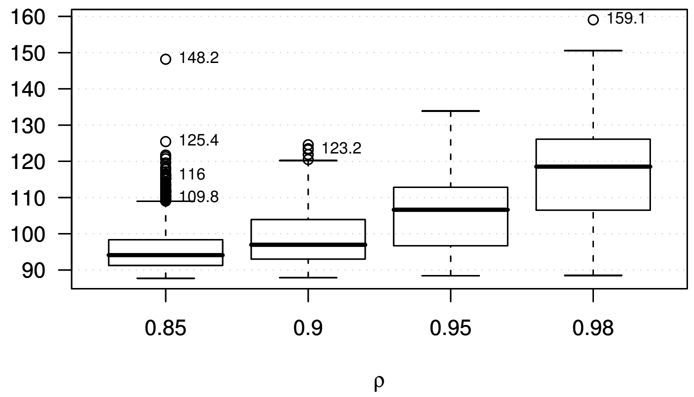


Figure 6.19: Results of MMAS for the Richmond network with respect to persistence factor (ρ).

Table 6.9: Experimental results for the Vanzyl network.

	C_{E}				N^{sw}			
Algorithm	med.	sd.	best	worst	med.	sd.	best	worst
AS	340.1	11.5	327.7	362.1	4	1.3	3	8
MMAS	340.7	9.4	327.7	358.6	5	0.9	3	6
Hybrid GA	347.1	4.3	344.4	354.8	4	0.8	3	5
SEA	334.1	6.1	315.9	341.4	5	1.2	3	7

AS ($N_p^{\text{sw}} \leq 3$): $A = 40$, best-so-far, $\rho = 0.98$, $\alpha = 1$, $\beta = 0$.
MMAS ($N_p^{\text{sw}} \leq 3$): $A = 10$, best-so-far, $\rho = 0.90$, $p_{\text{best}} = 0.05$, $\alpha = 1$, $\beta = 0$.
SEA ($N_p^{\text{sw}} \leq 3$): relative time-triggers, see also Table 4.1 on page 66.
Hybrid GA: results from the 7 runs reported by van Zyl, Savic & Walters (2004).

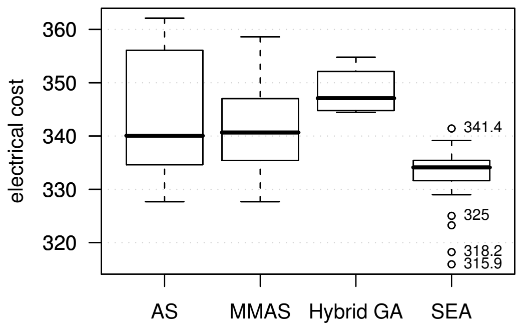


Figure 6.18: Comparison between AS, MMAS, Hybrid GA and SEA for the Vanzyl network.

Table 6.10: Statistical comparison of algorithms for the Vanzyl network. P-values reported by pairwise Wilcoxon rank sum tests, adjusted for multiple comparison using Holm's method.

	AS	MMAS	Hybrid GA
MMAS	0.97678	—	—
Hybrid GA	0.54780	0.12909	—
SEA	0.00375	0.00045	3.6e-06

We perform a three-way ANOVA, that is, we expect that the value of one parameter may change the way two other parameters interact. This seems to adjust better to the data as we shall see. The ANOVA identifies all third-order interactions between the parameters, with the exception of the interaction between A, p_{best} and ρ, as causing a significant effect on the data. Since all significant third-order interactions contain the selection parameter, we inspect the interaction plots for each type of selection method, that is, iteration-best (**ib**) and best-so-far (**bf**) selection. The main conclusions are that a moderate number of ants (lower than $A = 80$) and high evaporation (low ρ) works best in general, while whether to use a high or low p_{best} depends on the particular selection method. High p_{best} works better with iteration-best selection, while low p_{best} is better when using best-so-far selection.

Persistence factor (ρ) and number of ants (A). The interaction plot (Fig. 6.20) indeed show different behaviours for each selection method. Figure 6.20a shows that for **ib** selection a low evaporation (high ρ) produces typically worse results and even worse as the number of ants increase. While for high evaporation (low ρ), a high number of ants produces better results than 10 ants. On the other hand, Fig. 6.20a shows that for **bf** selection the only significant aspect is that configurations of $A = 80$ combined with $\rho = 0.95$ result in particularly bad performance. This is the most extreme combination (highest number of ants, highest ρ) and Table B.4 shows that further increasing ρ makes results much worse. Therefore, this combination of parameters suggests a threshold at which the MMAS algorithm loses its efficacy.

Probability of constructing the best solution (p_{best}) and number of ants (A). As for the relation between p_{best} and the number of ants, Fig. 6.21a shows that using tight pheromone limits (low p_{best}) is generally bad. However, this only applies for **ib** selection, while for **bf** selection (Fig. 6.21b) the results are quite the opposite and low p_{best} is the best approach. In addition, for both selection methods, $A = 80$ results in significantly worse schedules.

Probability of constructing the best solution (p_{best}) and persistence factor (ρ). The most significant interaction occurs between p_{best} and ρ, shown in Fig. 6.22. For **ib** selection, the combination of high p_{best} and low ρ produces the best results (Fig. 6.22a). In fact, values of ρ higher than 0.85 result in clearly worse schedules. On the other hand, for **bf** selection (Fig. 6.22b), a low value of p_{best} is better. Low ρ is also desirable but high values do not produce as bad results as for **ib** selection.

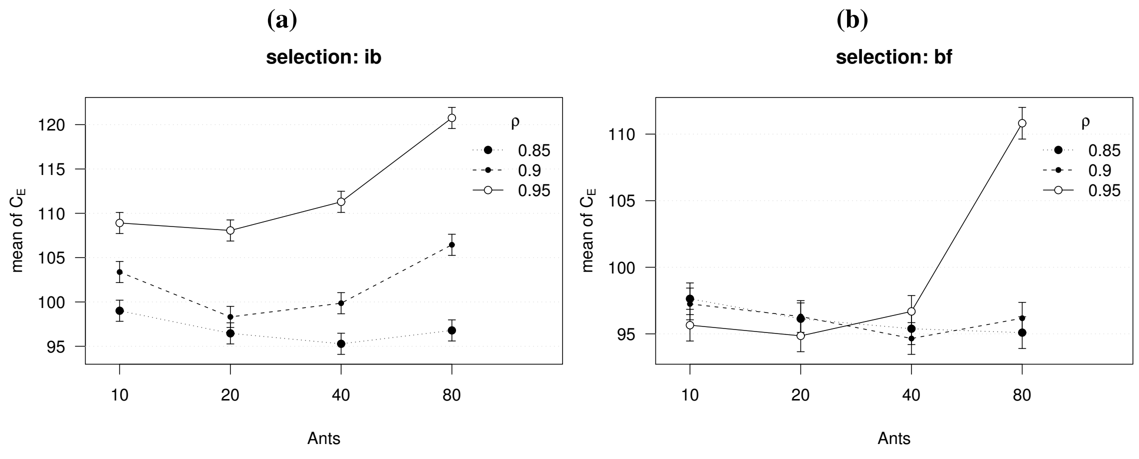


Figure 6.20: Interaction between the number of ants and persistence factor (ρ) in the results of MMAS for the Richmond network.

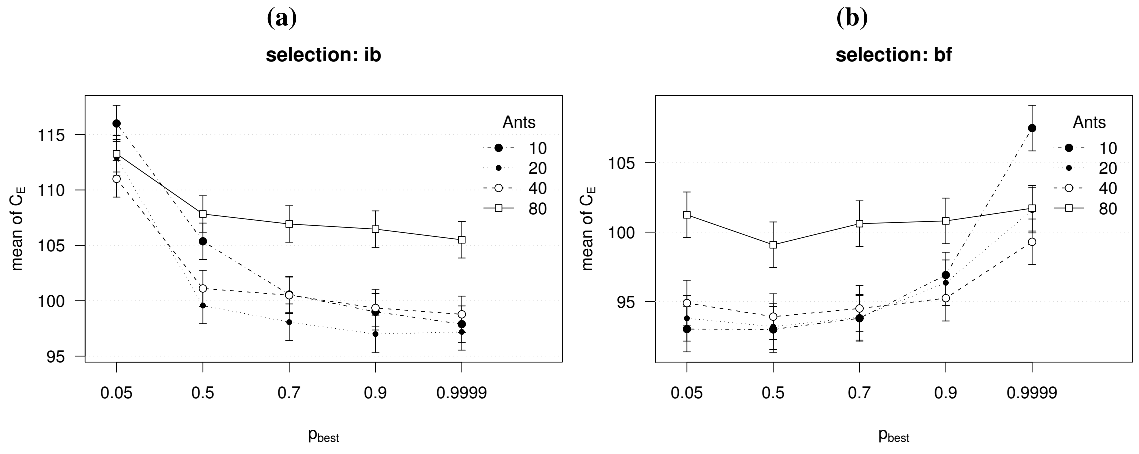


Figure 6.21: Interaction between the number of ants and p_{best} in the results of MMAS for the Richmond network.

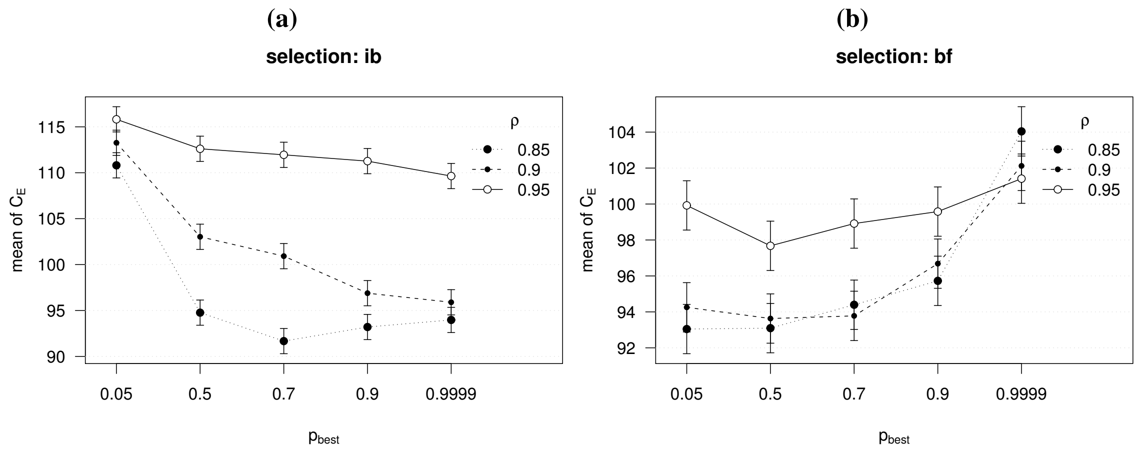


Figure 6.22: Interaction between p_{best} and persistence factor (ρ) in the results of MMAS for the Richmond network.

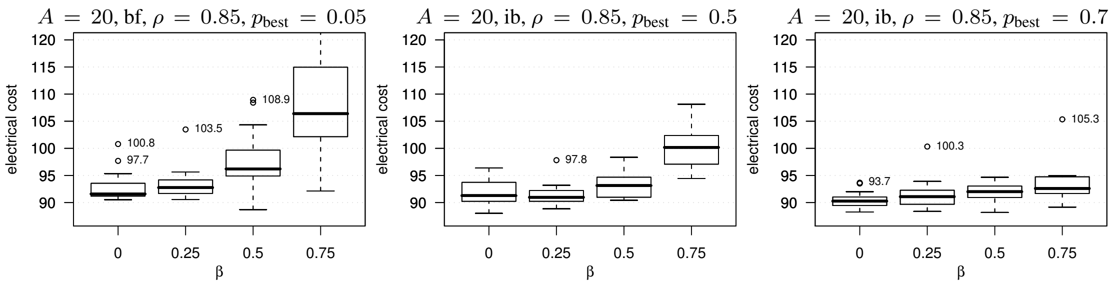


Figure 6.23: Influence of heuristic information (β) in three of the best configurations of parameters for the Richmond network.

Taking into account the above analysis, we identify as the best configurations of parameters a combination of iteration-best selection, high p_{best}, low ρ and a number of ants such as $A = 20$ or $A = 40$. Configurations of MMAS using best-so-far selection are better when using a number of ants different from $A = 80$, a low ρ and a low p_{best}. In particular, the configuration of parameters $A = 20$, iteration-best, $\rho = 0.85$ and $p_{\text{best}} = 0.7$ is the best with respect to the median electrical cost shown in Table B.4. Nevertheless, other configurations of parameters such as $A = 20$, iteration-best, $\rho = 0.85$ and $p_{\text{best}} = 0.5$, or $A = 20$, best-so-far, $\rho = 0.85$ and $p_{\text{best}} = 0.05$, produce similarly good results.

Heuristic Information

We focus on the three configurations identified above to study the effect of using heuristic information. Figure 6.23 shows the effect of the parameter β for three different configurations of MMAS. Although a small improvement is obtained for the second configuration (middle plot) with $\beta = 0.25$, the overall conclusion is that the use of heuristic information does not improve the results of MMAS with fine-tuned parameters. In fact, a high value of β may result in much worse quality.

Comparison among Algorithms

Finally, we compare the results of AS, MMAS, Simple Evolutionary Algorithm (SEA) and Hybrid Genetic Algorithm (van Zyl, Savic & Walters, 2004) in the Richmond network. From Table 6.11 and the corresponding Fig. 6.24, we conclude that Hybrid GA obtains the worst results out of the four algorithms, while the difference between SEA and MMAS is not so clear. A Wilcoxon rank sum test rejects the hypothesis that SEA and MMAS obtain equal median electrical cost with a p-value $= 0.001408$. We also calculate a non-parametric 95% confidence interval around the true difference between the median of SEA minus the median of MMAS equal to $[0.843, 3.409]$, which suggests that the real difference is relatively small but clearly in favour of MMAS. Compared with AS,

Table 6.11: Experimental results for the Richmond network.

	C_E				N^sw			
Algorithm	med.	sd.	best	worst	med.	sd.	best	worst
AS	90.1	1.9	88.7	94.7	14	1.5	10	15
MMAS	90.3	1.6	88.3	93.7	12	2.0	9	15
Hybrid GA	99.5	2.2	97.0	104.0	—	—	—	—
SEA	92.3	1.6	90.3	95.4	16	1.7	14	20

AS ($N_p^\text{sw} \leq 3$): $A = 20$, iteration-best, $\rho = 0.90$, $\alpha = 1$, $\beta = 0.25$.
MMAS ($N_p^\text{sw} \leq 3$): $A = 20$, iteration-best, $\rho = 0.85$, $p_\text{best} = 0.7$, $\alpha = 1$, $\beta = 0$.
SEA ($N_p^\text{sw} \leq 3$): relative time-triggers, see also Table 4.2 on page 67.
Hybrid GA: energy costs are estimated from graphical results from van Zyl, Savic & Walters (2004).

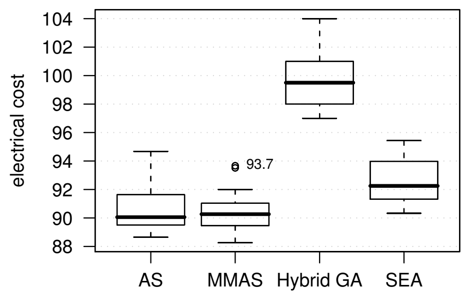


Figure 6.24: Comparison of AS, MMAS, Hybrid GA and SEA for the Richmond network.

MMAS obtains quite similar results but a lower variability. Therefore, we must conclude that MMAS is an overall improvement over the other three algorithms in the case of the Richmond network instance.

6.5 Reducing the Computation Time of ACO

6.5.1 Computational Effort of ACO

The proposed ACO approach for the Pump Scheduling problem has some characteristics that would not be appropriate for other problems where hundreds of thousands of evaluations can be executed in a relatively short time. In particular, we used multiple pheromone matrices, one for each pump, and each matrix has dimensions *intervals* $\times$ *durations*. For example, in the Richmond network there are seven pumps and, if we establish a limit of three switches per pump, there are six decision variables (intervals) per pump. For a scheduling period of 24 hours and a relaxed constraint on the number of

Table 6.12: Computation time of AS for different network instances and different constraints of the number of pump switches.

Instance	Constraint	Time	mean	sd.	min.	max.
Vanzyl	$N_p^{\text{sw}} = 3$	Total (s)	128.01	74.46	20.80	330.10
		Overhead (s)	0.29	0.05	0.09	0.44
		Overhead (%)	0.34	0.23	0.07	1.43
	$N_p^{\text{sw}} \leq 3$	Total (s)	161.73	94.06	27.28	389.70
		Overhead (s)	0.34	0.06	0.09	0.91
		Overhead (%)	0.34	0.27	0.05	2.34
Richmond	$N_p^{\text{sw}} = 3$	Total (s)	4273.64	892.70	2554.00	6833.00
		Overhead (s)	0.89	0.10	0.67	1.21
		Overhead (%)	0.02	0.01	0.01	0.04
	$N_p^{\text{sw}} \leq 3$	Total (s)	5725.38	1412.59	2432.00	11020.00
		Overhead (s)	1.29	0.23	0.78	1.98
		Overhead (%)	0.02	0.01	0.01	0.05

pump switches ($N_p^{\text{sw}} \leq 3$), the duration of each interval is within $[0, 24]$. Therefore, there are $7 \times 6 \times 25 = 1050$ pheromone values that need to be considered when constructing each solution and when performing evaporation. Is the resulting overhead in computation time significant?

To answer this question, we measured the computation time required by AS when running on an Intel Pentium 4 ($3.20\,\text{GHz}$) with $1024\,\text{KB}$ of cache size under GNU/Linux 2.4.20. Table 6.12 shows the mean, standard deviation, minimum and maximum of total CPU-time, overhead time and overhead percentage for 875 runs of AS for the Vanzyl network for each constraint and 480 runs for the Richmond network for each constraint with the parameter values analysed in Section 6.3.2. Total CPU-time is the seconds that the CPU has spent executing each run, overhead is the seconds that the CPU has spent outside the evaluation function, and overhead percentage is just the percentage of the total time represented by the overhead time. The time spent by AS evaluating solutions is the total time minus the overhead time. The worst overhead percentage for the Vanzyl is slightly more than 2% of the total time, no more than one second. For the Richmond network the difference is even stronger: the worst percentage is merely 0.05% of the total and, on average, only one second is consumed by AS itself, while the rest of the one hour and a half of execution time is spent evaluating schedules. The computational overhead of the AS algorithm is, therefore, negligible, and becomes even less relevant as the network instance grows in complexity. This result indicates that the largest gains in computation time will be achieved by reducing either the number of function evaluations or the time required by each evaluation.

6.5.2 Parallel Evaluation of Schedules in ACO

In the ACO algorithms described in this chapter, once all ants have constructed one solution each, all solutions must be evaluated before updating the pheromones. Solutions are evaluated sequentially as shown in Fig. 6.25. However, these evaluations could be performed in parallel by using as many threads as the number of ants.[3] The speedup of a parallel algorithm is defined as the time required by a sequential algorithm divided by the time required by its parallel variant. The maximum speedup that can be achieved by evaluating each solution in parallel is limited by the number of ants. The number of ants typically ranges from 10 to a few hundreds, which is often larger than the number of CPUs available in a multi-core computer. Hence, the number of ants is not a limitation to the potential speedup of the algorithm in practice.

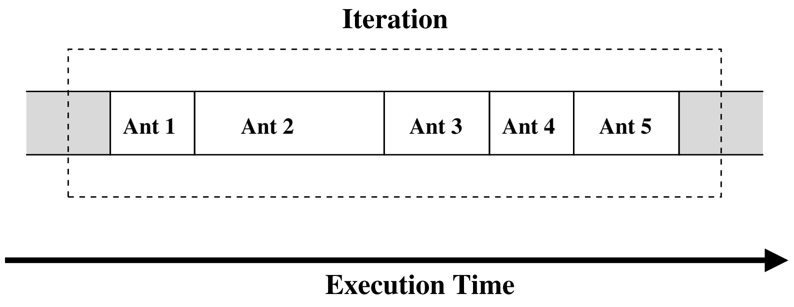


Figure 6.25: Sequential evaluation of solutions in ACO.

We must, as well, take into account that some schedules may require more simulation time than others. This may be due to several factors, such as the time required to find a solution to the hydraulic equations. System constraints, such as pressure constraints, may be violated early into the simulation, and, thus, the schedule would be considered infeasible without requiring a complete simulation. Whatever the reason, the fact is that some schedules will require more computation time than others and, hence, depending on the assignment of schedules to threads, some threads may take considerably more time to finish. The effect of long simulation times can be minimised by making the assignment dynamic, instead of equally distributing the solutions among the threads. In a dynamic assignment, one solution is assigned to each thread and the rest of solutions are assigned dynamically as threads finish evaluating previous solutions. A higher ratio of solutions per thread would also tend to minimise the impact of particularly long simulations. Figure 6.26 shows a timeline of the execution of one iteration of the parallel ACO algorithm using three threads. In this example, ant 5 is assigned to the third thread because the other threads are still busy evaluating the other ants' solutions.

[3] Threads are lightweight processes that can execute concurrently within the same computer program.

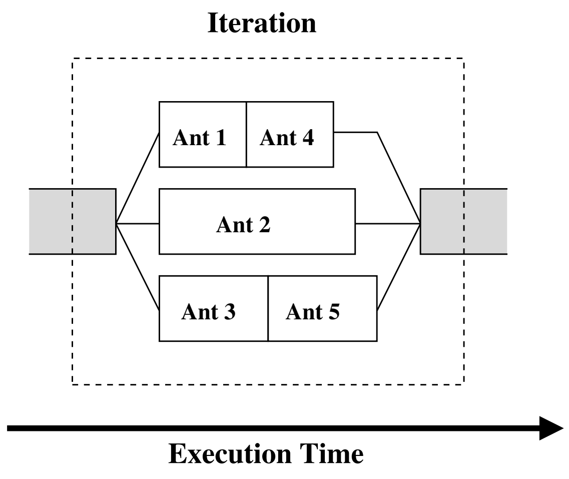


Figure 6.26: Parallel evaluation of solutions in ACO.

Finally, the time required by the sequential code, which, by definition, is not reduced by using a higher number of parallel threads, is another factor that influences the maximum speedup. If the stopping criteria of ACO are a maximum number of evaluations or a time limit, then a smaller number of ants would increase the number of iterations of the algorithm. This, in turn, increases the time required by the sequential part of the algorithm. However, as we concluded in the previous section, the time required by ACO is negligible compared with the time required by the hydraulic simulations. Hence, we make no effort to reduce the time required by the sequential parts of the algorithm.

6.5.3 Experimental Evaluation of Parallel ACO

We empirically test the benefits of the parallel ACO approach described above. The underlying ACO algorithm is the AS algorithm described in this chapter. This algorithm is modified to incorporate the parallel evaluation of solutions by dynamically assigning solutions to a number of threads. This algorithm is linked to a thread-safe version of EPANET. Appendix E provides more details abut the development of this thread-safe variant of EPANET.

The goal of our empirical study is to analyse the performance, in terms of wall-clock time, of the algorithm. The performance in terms of solution quality is not considered here because the parallel variant generates the same sequence of solutions as the sequential ACO algorithm. In other words, given the same parameters, the quality of solutions generated by both algorithms is the same and only the computation time is smaller in the parallel variant. Our objective is to determine how much computation time is reduced by the use of multiple concurrent threads. We have argued that the number of ants may have some effect on the computation time. Thus, several values for the number of ants are tested. We apply the parallel ACO algorithm to the optimisation of pump schedules in the Richmond network and the algorithm is stopped at $8,000$ evaluations. Experiments are performed on a 4-CPU machine (2 dual-core AMD64 Opteron 275, 2.2 GHz and 64KB/1MB of cache memory per core) running GNU/Linux. The algorithm is implemented in C and uses POSIX threads (Kerrisk, 2005).

We conducted several runs of ACO with different number of ants (5, 10, 20, 40, 80) and different number of threads $(1, 2, 3, 4, 5, 6)$. Figure 6.27a shows the wall-clock time taken by ACO for each combination of parameters, while Fig. 6.27b gives the corresponding speedup. Figure 6.27a shows that the sequential ACO (corresponding to using one thread) requires almost two hours of computation time. The differences in computation time for the sequential case are explained by the different sequence of solutions generated when using different number of ants. As explained above, for the same number of ants, the same sequence of solutions is generated independently of the number of threads. However, different number of ants will generate different results and, thus, there will be variations in the computation time.

The speedup for each value of the number of ants in Fig. 6.27b is calculated with respect to the computation time required when using the sequential ACO algorithm and the same number of ants. Therefore, variations in the time required by the sequential algorithm do not translate into variations in the speedup. It is an interesting result that the speedup decreases with decreasing number of ants, since this indicates that parallelism is better exploited by using a high number of ants. The explanation for this result was already given in the previous section: the higher ratio of ants to threads allows a better utilisation of the multiple CPUs and minimises the impact of schedules that require particularly long simulation time. The overall, conclusion is that, in the parallel ACO algorithm, a higher number of ants reduces the computation time. This is an encouraging result because, as reported in Section 6.4.3 above, the ACO algorithm obtains the best schedules, in terms of quality, when using a number of ants equal to 20 or 40.

6.6 Summary

In the present chapter we have gone a step forward in tackling the Pump Scheduling problem with the adoption of the ACO framework. This has been possible by means of the new time-controlled triggers representation. Definitions for the pheromone and heuristic information, the idea of using one pheromone matrix for each pump and an iterative construction mechanism for building pump schedules have been proposed throughout this chapter. These elements have enabled us to apply the Ant System (AS) algorithm to the problem of pump scheduling. Empirical results have shown the importance of the balance between exploration and convergence.

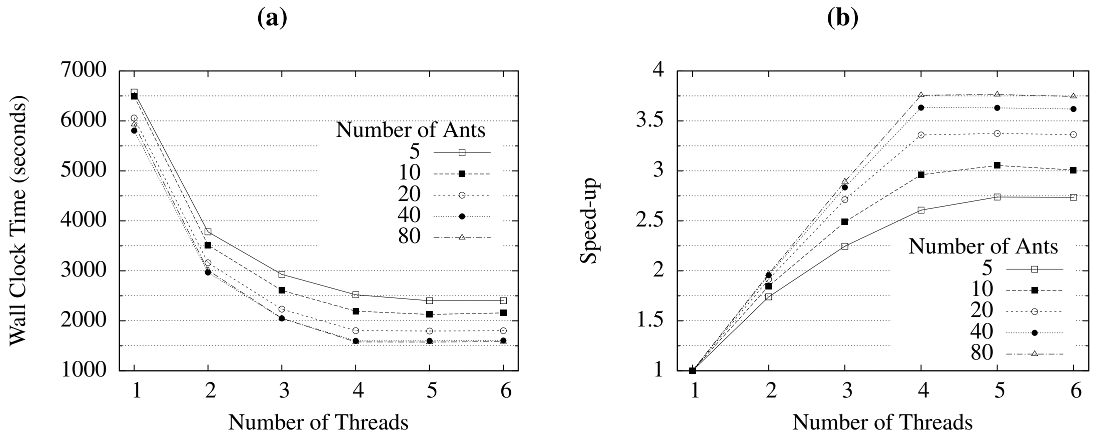


Figure 6.27: Runtime in seconds (a) and speedup (b) of AS algorithm with parallel evaluation of schedules.

The best configurations of parameters have been identified by using a full factorial empirical analysis for each network instance. The AS algorithm was able to outperform its competitors with the Richmond instance. However, for the Vanzyl network the results of AS showed a high variability. This is an indication that the Vanzyl network should not be used as a test instance to fine-tune the Richmond network, since the best configurations for each of them are notably different.

Another interesting finding is that, in the case of the Richmond network, the relaxed constraint not only obtains a lower number of pump switches, as was expected, but it is able to obtain a lower electrical cost than the strict constraint, despite the increase in the size of the search space. Since the AS algorithm has found the best-known solutions for the Richmond instance, it may well be that the extra flexibility of the relaxed constraint allows the algorithm to find better schedules that would violate the strict constraint.

We have also implemented the Max-Min Ant System (MMAS) algorithm and empirically studied its performance. The analysis showed that MMAS reduces the variability of the results with respect to those obtained by the simpler AS algorithm. In particular, it typically improves the worst-case. In spite of this, SEA is still better than MMAS for the Vanzyl network, whereas for the Richmond network, MMAS is the clear winner.

Finally, we have considered the potential overhead in computation time caused by the proposed pheromone representation and the use of multiple pheromone matrices. The conclusion of our analysis is that the time required by the ACO algorithm itself is insignificant when compared with the time required by the simulations of the schedules. In particular, for large real-world networks, such as the Richmond network, where a run of eight thousand evaluations may take several hours, the fraction corresponding to the ACO algorithm is just a few seconds. Therefore, optimising the ACO algorithm is not worth-while and we would rather chose to incorporate new techniques that lead to reducing the number of evaluations or improving the quality of the schedules.

Following these conclusions, we have proposed to evaluate pump schedules in parallel by means of a thread-safe variant of EPANET. Experimental results showed that the running time of ACO can be greatly reduced by this technique: from almost two hours to less than half and hour when using four CPUs. Our experiments with the parallel ACO algorithm indicated as well that using a large number of ants and more threads than CPUs usually gives the shortest computation time. The quality of the results is not affected by the parallel approach, since the ACO algorithm generates the exact same schedules but in a much shorter time. Therefore, the conclusions obtained from the experimental analysis of the sequential ACO algorithm are still valid for the parallel variant.

Chapter 7

Conclusion

The optimisation of pump operations in water distribution networks is an important problem in practice since it may result in substantial energy and monetary savings. By means of complex hydraulic simulation models and automatic data collection, a very precise model of a water distribution system can be obtained. However, it is the task of an optimisation algorithm to generate candidate pump schedules that minimise energy and maintenance costs while providing a reliable and satisfactory service to customers.

General-purpose optimisation algorithms, such as Evolutionary Algorithms, have been successfully applied to practical cases of the pump scheduling problem. However, the research literature is lacking on experimental analysis and comparison of alternative approaches in order to assess the effectiveness of existing methods.

The main subject of this thesis has been the development, application and experimental analysis of optimisation approaches to the pump scheduling problem. The following sections review the main conclusions obtained from our research. Next, we briefly summarise the contributions of the thesis and publications arising from or related to this work. The final section gives some insight on how our conclusions can be applied beyond this thesis and how our work could be extended in new directions.

7.1 Summary and Conclusions of this Thesis

The work in this thesis can be divided into three main parts. First, the formulation of pump scheduling as an optimisation problem. Second, the application of various optimisation approaches to pump scheduling problem. And third, the exhaustive experimental analysis and comparison of these approaches.

7.1.1 Pump Scheduling as an Optimisation Problem

The formulation of the pump scheduling problem is discussed in detail in Chapter 2 of this thesis. Within the context of this discussion, we have examined a very general and flexible formulation of the objectives and constraints of the problem. This formulation may be applied to multiple instances of the problem. For example, we have proposed concepts, such as *volume deficit tolerance*, that permit adapting the general formulation to more precise applications. We introduced, as well, a distinction between the total number of pump switches and the number of switches per pump. From our assessment, the latter is more relevant in order to limit maintenance costs. In Chapter 2, we have additionally carried out a thorough review of previous approaches for automated pump scheduling. Chapter 2 also summarised the numerous improvements we have incorporated into the original hydraulic simulator (EPANET). A more complete list of our modifications was given in Appendix D.

Two new representations of pump schedules based on the concept of time-controlled triggers were proposed in Chapter 3. One representation is based on absolute time with respect to the start of the scheduling period. The other uses relative time to indicate elapsed time between pump switches. The new representations implicitly constraint the number of switches per pump, leading to a significant reduction in the search space. In addition, we have developed several recombination and mutation operators as the means to use the new representations in evolutionary algorithms. Exhaustive experimental analysis has conclusively shown that an evolutionary algorithm using the new representations is able to obtain better results than using the traditional representations, while at the same time ensuring a maximum limit of switches per pump

It is clear from our experiments that the level-controlled triggers representation may lead to rapid switching of pumps and, hence, produce sudden pressure fluctuations that may cause damage to the pipes connected to the pump. For this reason, we have proposed the maximisation of the shortest idle time as a surrogate measure of maintenance costs. In Chapter 5, the shortest idle time is precisely defined and experimental tests are performed by means of multi-objective optimisation.

7.1.2 Optimisation Algorithms for Pump Scheduling

During this work, various optimisation approaches were implemented: a single-objective evolutionary algorithm, a multi-objective evolutionary algorithm and two ant colony optimisation algorithms.

Firstly, a single-objective evolutionary algorithm called the Simple Evolutionary Algorithm (SEA) was developed in Chapter 4. This algorithm uses no penalty values to handle constraints and it is linked to a hydraulic simulator (EPANET) to evaluate potential schedules. SEA is adapted to 4 different representations of pump schedules: two traditional representations, binary and level-controlled triggers, and the proposed absolute and relative time-controlled triggers.

Secondly, the well-known multi-objective evolutionary algorithm SPEA2 was applied to the pump scheduling problem. By means of SPEA2, we addressed two alternative formulations of a multi-objective pump scheduling problem in terms of Pareto optimality. The first alternative minimised both electrical cost and number of pump switches. The second variant, replaced the minimisation of pump switches by the maximisation of the shortest idle time.

Finally, the proposed time-controlled triggers representation allowed us to apply Ant Colony Optimisation to pump scheduling. To this end, we have developed suitable definitions for the pheromone information, solution (pump schedule) construction, pheromone update and heuristic information. We first tested the approach using the original Ant System algorithm and later improved it by using a modern algorithm, Max-Min Ant System.

7.1.3 Experimental Analysis and Comparison

To the best of our knowledge, we have performed the most exhaustive experimental analysis of the pump scheduling problem available in the literature. Our experimental setup has compared, for the first time, multiple optimisation techniques using various representations and different network instances. We have relied on sound statistical techniques from the field of Design of Experiments to assess whether observed differences between algorithms were, in fact, statistically significant or could be attributed to the stochastic nature of the algorithms.

In the case of multi-objective optimisation, we have used state-of-art methods. Some of the tools utilised in the analysis were developed in parallel to this thesis in collaboration with other researchers. In particular, during the course of this thesis, we have collaborated in the development of a faster algorithm for the calculation of the hypervolume (Fonseca, Paquete & López-Ibáñez, 2006), and a method to visualise the differences between the empirical attainment functions of multi-objective algorithms (López-Ibáñez, Paquete & Stützle, 2006). These techniques have been applied in this thesis.

Experiments on a Single-Objective Evolutionary Algorithm

An exhaustive experimental analysis was performed on SEA to identify the best settings for each representation. Then, a comparison among representations was carried out. Our analysis showed that results obtained by SEA with either the binary representation or relative time-controlled triggers outperformed results obtained by a more complex algorithm from the literature designed for level-controlled triggers.

The effect of the constraint on the number of switches per pump was empirically studied for the binary representation. Our results suggest that schedules with low electrical cost typically have a moderate number of pump switches. This observation contradicts the intuitive notion that a higher number of pump switches would provide more flexibility and lead to lower cost schedules. We also concluded that the constraint on pump switches has an effect on the electrical cost that greatly depends on the particular limit and the network instance. For some networks, a lower limit may reduce the electricity cost of the schedules generated by the optimisation algorithm. In other cases, a excessively strict limit may hinder the search and result in sub-optimal schedules. A maximum limit of three switches per pump was found to be appropriate for the Vanzyl network. For the Richmond network, on the other hand, some experiments suggested that this limit was too strict and a maximum limit of four or five switches per pump would result in lower electricity cost. Nonetheless, even a limit of three switches per pump generated state-of-the-art results for the Richmond network.

With regard to the new representation based on time-controlled triggers, we investigated various settings for the implicit constraint on number of pump switches. We first searched for optimal schedules with different number of switches per pump. The best schedules in terms of electrical cost had two or three switches per pump. Schedules with one switch per pump produced a very high electrical cost due to the lack of flexibility of the schedule. Schedules with more than three switches per pump produced increasingly higher electrical costs due to the exponential growth of the search space. Next, we performed experiments using a relaxed constraint on the number of pump switches that allowed less switches per pump than the limit. Results showed that, when using the relaxed constraint, SEA is very robust to different settings of the maximum limit and, in fact, the number of pump switches of the resulting schedules grows far slower than the maximum number allowed by the limit.

Lastly, we tested various settings of the minimum time interval in the time-controlled triggers representation. The overall conclusion from our experiments is that intervals larger than one hour do not have enough flexibility and typically produce higher electrical cost. As for intervals smaller than one hour, they may lead to significant reductions in electrical cost if the algorithm is given sufficient time. However, for a run of typical duration, the growth of the search space hinders the search and produces worse results. Therefore, we concluded that a minimum time interval of one hour was the adequate setting.

Experiments on a Multi-Objective Evolutionary Algorithm

We next experimented with multi-objective variants of the pump scheduling problem by means of SPEA2. Firstly, we considered a multi-objective approach to minimise both electrical cost and number of pump switches. An exhaustive experimental analysis was carried out involving various representations and network instances. The results of the multi-objective approach was compared with the state-of-the-art algorithm in the literature. Our experimental results showed that the multi-objective approach using time-controlled triggers outperformed the algorithm proposed in the literature designed for level-controlled triggers. The multi-objective algorithm generated schedules with lower electrical cost and a lower number of pump switches within the same computation time. In addition, a single run of the multi-objective algorithm provides system operators with multiple optimal schedules that model the trade-off between electrical and maintenance costs.

We also proposed the maximisation of the shortest idle time between operating periods as an alternative to the minimisation of pump switches. After an experimental assessment of this approach, we believe that this approach does not lead to the lowest electricity cost in general. However, our experimental results showed that schedules with very long idle times may be found with slightly higher electrical cost than those obtained by the single-objective algorithm.

Experiments on Ant Colony Optimisation

In Chapter 6, the pump scheduling problem was adapted to the Ant Colony Optimisation (ACO) framework by means of the time-controlled triggers representation. The proposed approach was implemented using two different ACO algorithms: Ant System and Max-Min Ant System.

First, we empirically tested various settings for the parameters of Ant System. Results showed that Ant System outperformed, in the Richmond network, both the state-of-the-art algorithm from the literature for level-controlled triggers, and the single-objective evolutionary algorithm (SEA) proposed in this work. However, the analysis was not conclusive in the case of the Vanzyl network. The application of Max-Min Ant System further improved the results of Ant System on both networks. Nonetheless, SEA was still the best approach for the Vanzyl network.

Heuristic information for the pump scheduling problem was proposed as well. This heuristic information assumed that schedules with low energy cost have long idle periods and short operating periods. After an examination of our experimental results, we concluded that this heuristic information does not help the ACO algorithms to reduce energy cost, as long as the algorithms are configured with adequate parameter settings.

Experiments on Computation Effort

The potential overhead in computation time caused by the proposed ACO approach was thoroughly explored in Chapter 6. After the analysis of empirical measures of computation time, our conclusion is that the computation time consumed by the ACO algorithm itself is insignificant when compared with the time spent on the simulation of candidate schedules. In particular, for large real-world networks, such as Richmond, where a typical run of eight thousand evaluations may require more than one hour, the computation time corresponding to the ACO algorithm itself amounted to a few seconds.

As a consequence of this, we proposed to evaluate pump schedules in parallel taking advantage of multi-core computers. Hence, we modified the Ant System algorithm to simulate multiple schedules in parallel at each iteration by means of a new thread-safe variant of EPANET. Experimental results showed a significant reduction of computation time when running on a four CPUs computer, without any loss of solution quality.

The thread-safe version of EPANET utilised in the above experiments was developed during the course of this thesis. This thread-safe library may be linked to a wide range of parallel optimisation algorithms for water engineering problems, enabling concurrent simulation of different schedules. Appendix E discusses the limitations of the original EPANET that motivated our work, the strategy followed to implement the new library and some experimental results obtained by a parallel random search algorithm. From our experimental testing, we believe that the thread-safe version has no significant overhead with respect to the original EPANET.

7.2 Contributions of this Thesis

A brief summary of the main contributions of this thesis is:

1. A very general and flexible definition of the pump scheduling problem and its constraints that may be adapted to multiple instances of the problem. By comparison, the objectives and constraints of the pump scheduling problem are loosely defined or implicitly assumed in other works. Our definition explicitly introduces concepts such as *volume tolerance*. Moreover, we explicitly distinguish between the total number of pump switches and the number of switches per pump, arguing that the latter is better suited to model maintenance costs.

2. The proposal of two new representations for pump schedules based on the concept of time-controlled triggers. The new representations implicitly constraint the num-

ber of switches per pump, avoiding the need of an explicit constraint and leading to a significant reduction in the search space. Moreover, several recombination and mutation operators were developed for the new representations. Exhaustive experimental analysis has conclusively shown that an evolutionary algorithm using the new representations is able to obtain better results than using the traditional representations.

3. An evolutionary algorithm called SEA was developed. This algorithm uses no penalty values to handle constraints and is linked to a hydraulic simulator (EPANET) to evaluate potential schedules. We concluded that the representation based on level-controlled triggers, which is the current practice in UK, performs significantly worse than the rest. In fact, SEA using either the binary representation or relative time-controlled triggers obtained state-of-the-art results for two network instances from the literature.

4. The effect of the constraint on the number of switches per pump was empirically investigated for the binary representation. Our results suggest that schedules with low electrical cost typically have a moderate number of pump switches. This observation contradicts the intuitive notion that higher number of pump switches would provide more flexibility and lead to lower cost schedules. The particular constraint on pump switches may have different effects on the resulting electrical cost. For some networks, a lower limit may reduce the electricity cost of the schedules generated by the optimisation algorithm. In other cases, a excessively strict limit may hinder the search and result in sub-optimal schedules.

5. We investigated, as well, the effect of various settings of the new representation based on time-controlled triggers. In the case of the implicit constraint on the number of pump switches, our conclusion is that if less switches than the limit are allowed, then the time-controlled triggers representation is able to focus on the best schedules even if the limit is higher than the optimal limit. On the other hand, an excessively strict limit (e.g., one switch per pump) will generate poor results. As for the effect of the minimum time interval, we can say that intervals smaller than one hour may produce schedules that further minimise the electrical cost if the algorithm is allowed to run for sufficient time. Otherwise, a setting of one hour generates the best results.

6. We have shown the viability of multi-objective optimisation for solving the Pump Scheduling problem, which allows system operators to examine a range of Pareto-optimal solutions and choose one solution with regard to additional criteria. The

importance of this has already been noticed by Ormsbee & Reddy (1995).[1] We have investigated two different multi-objective formulations.

7. We tested a multi-objective approach that minimises both electrical cost and number of pump switches in terms of Pareto optimality. We empirically showed that such multi-objective approach using relative time-controlled triggers generates schedules with lower electrical cost and a lower number of pump switches than the state-of-the-art algorithm available in the literature for level-controlled triggers, within similar computation time. A single run of the multi-objective algorithm, in addition, provides system operators with multiple optimal schedules that model the trade-off between electrical and maintenance costs.

8. We also proposed the maximisation of the shortest idle time as a surrogate measure of maintenance costs, and as an additional objective to the minimisation of electrical cost. Experiments with this multi-objective formulation generated schedules with very long idle times with slightly higher electrical cost than those obtained when only minimising the electrical cost.

9. The pump scheduling problem was adapted to the Ant Colony Optimisation framework. Our conclusion is that ACO algorithms, in particular Max-Min Ant System, using time-controlled triggers are able to generate lowest cost pump schedules in comparison to other algorithms available in the literature for level-controlled triggers optimisation.

10. We have developed a thread-safe variant of EPANET. The thread-safe EPANET enables the parallel evaluation of different pump schedules for the same network instance in a efficient manner. Backwards-compatibility was an important goal and, hence, the numerical results are equivalent as those produced by the original EPANET.

11. We have shown that most of the computation effort is spent on the simulation of potential schedules. We empirically showed that the overhead of ACO was negligible. Consequently, we proposed a variant of the ACO algorithm that takes advantage of parallel processors to significantly reduce the total computation time required by a single run. This result also applies to the other algorithms considered in this thesis.

[1] The exact quote is "Indeed, not only is an *optimal* solution obtained, but all resulting feasible solutions are available for examinations by the system operator. As a result, the operator is provided with an increased flexibility with regard to selection of alternative solutions that may not be optimal from a purely cost-savings objective but may provide a superior solution based on additional more subjective operational considerations." (Ormsbee & Reddy, 1995).

7.2.1 Publications Arising from this Thesis

- López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2005a. Multi-objective optimisation of the pump scheduling problem using SPEA2. **In**: *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 435–442. Edinburgh, UK.

- López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2005b. Optimal pump scheduling: Representation and multiple objectives. **In**: Savic, D.A., Walters, G.A., King, R. & Thiam-Khu, S., eds., *Proceedings of the Eighth International Conference on Computing and Control for the Water Industry (CCWI2005)*, vol. 1, pp. 117–122. University of Exeter, UK.

- López-Ibáñez, M., 2007. High performance ant colony optimisation of the pump scheduling problem. **In**: Alberigo, P., Erbacci, G., Garofalo, F. & Monfardini, S., eds., *Science and Sumpercomputing in Europe*, pp. 371–375. CINECA. ISBN 978-88-86037-21-1.

- López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2008a. Ant colony optimisation for the optimal control of pumps in water distribution networks. *Journal of Water Resources Planning and Management, ASCE*, **134** (4) pp. 337–346.

- López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2008b. Parallel optimisation of pump schedules with a thread-safe variant of EPANET toolkit. **In**: van Zyl, J.E., Ilemobade, A.A. & Jacobs, H.E., eds., *Proceedings of the 10th Annual Water Distribution Systems Analysis Conference WDSA2008*. ASCE.

7.2.2 Other Publications Related to this Thesis

The subject of these publications is not directly related to this thesis, however, the techniques and methods developed in these publications have been used in this thesis.

- Paquete, L., Stützle, T. & López-Ibáñez, M., 2005. Towards the empirical analysis of SLS algorithms for multiobjective combinatorial optimization problems through experimental design. **In**: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F. & Reimann, M., eds., *6th Metaheuristics International Conference (MIC 2005)*, pp. 739–746. Vienna, Austria.

- López-Ibáñez, M., Paquete, L. & Stützle, T., 2006. Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, **5** (1) pp. 111–137.

- Fonseca, C.M., Paquete, L. & López-Ibáñez, M., 2006. An improved dimension-sweep algorithm for the hypervolume indicator. **In**: *IEEE Congress on Evolutionary Computation*, pp. 1157–1163. IEEE Press.
- Paquete, L., Stützle, T. & López-Ibáñez, M., 2007. Using experimental design to analyze stochastic local search algorithms for multiobjective problems. **In**: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F. & Reimann, M., eds., *Metaheuristics: Progress in Complex Systems Optimization*, *Operations Research / Computer Science Interfaces*, vol. 39, pp. 325–344. Springer-Verlag, New York.

7.2.3 Application and Extensions of this Work

The present work provides the groundwork for both practical applications of its results and further research studies based on its conclusions.

Although level-controlled triggers is the current practice in UK, the results in this thesis show that alternative representations may provide significant energy and cost savings. Moreover, existing algorithms can be adapted to use the time-controlled triggers representation in order to enforce an implicit limit of switches per pump and restrict the search space to those schedules that satisfy this limit. Finally, the results obtained by means of the proposed thread-safe variant of EPANET should encourage a more wide-spread usage of parallel computation in order to decrease response time of existing automatic pump scheduling systems.

With regard to further research extending the present work, the work described in this thesis provides a starting point for further empirical analysis and comparison of algorithms for pump scheduling in additional network instances. We hope that the encouraging results obtained by the new representations motivates research on alternative representations. In addition, refinements and new applications of the proposed representations should be possible.

This work has illustrated the use of statistical analysis techniques and performance assessment methods that indicate the actual factors that determine the performance of an algorithm. These techniques indicate, as well, whether perceived differences between algorithms can be deemed significant. We believe that future empirical research on operational optimisation of water distribution networks should employ similar techniques.

Given its scope, this thesis can only explore a small part of the potential applications of multi-objective optimisation. Many-objectives formulations, and interactive or goal-based multi-objective optimisation may be areas of further investigation for the purpose of obtaining pump schedules that satisfy the expectations of system operators.

Although the results of ACO presented in this work outperform those available in the literature, we believe that they may be further improved. In particular, the heuristic information utilised in this work was shown to be mostly ineffectual. Hence, an extension to this work may investigate alternative heuristics and compare their results with those presented in this thesis.

Finally, we have observed that most of the computation time is spent on the hydraulic simulation of potential schedules. Therefore, future algorithms should be developed with the goal of decreasing the number of simulations required to achieve a satisfactory solution. An alternative and complementary goal would be shortening the time required by the hydraulic simulations. We have shown that a significant reduction of the computation time can be achieved by using parallel computers. Other techniques worth of consideration would be partial simulation of schedules and approximated evaluation of candidate schedules.

The application of our thread-safe variant of EPANET to other optimisation algorithms, such as evolutionary algorithms, and more complex parallel algorithms is a logical next step. Closely related to the pump scheduling problem are water quality problems, which can also be evaluated through EPANET. Water quality simulation requires even longer execution times, thus the benefits of extending the thread-safe library to handle water quality simulation in parallel would be substantial.

Appendix A

Results of Simple Evolutionary Algorithm

Table A.1: SEA with the binary representation and constraint $N_p^{\text{sw}} \leq 3$ for the Vanzyl network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	flip	333.0	11.2	324.7	359.6	7	1.2	5	9
50	5	one-point	none	396.6	16.5	377.3	431.4	13	1.7	10	17
50	5	two-point	flip	341.9	12.2	327.4	361.9	8	1.2	4	9
50	5	two-point	none	406.7	19.0	369.7	452.3	14	2.3	9	17
50	5	uniform	flip	333.9	11.1	323.0	358.8	7	1.1	5	9
50	5	uniform	none	393.2	21.9	363.6	429.6	9	1.5	8	13
50	20	one-point	flip	335.6	12.0	327.7	361.9	7	1.1	5	9
50	20	one-point	none	390.6	15.2	364.7	423.1	12	2.2	9	18
50	20	two-point	flip	349.4	11.2	328.2	359.8	7	1.2	5	9
50	20	two-point	none	404.8	14.0	378.1	435.7	14	1.6	11	17
50	20	uniform	flip	343.2	11.5	324.5	356.4	7	0.8	5	8
50	20	uniform	none	380.0	19.8	351.2	433.8	9	0.9	7	11
100	5	one-point	flip	337.6	11.1	323.0	361.1	7	1.2	4	9
100	5	one-point	none	389.9	13.2	360.2	422.6	11	1.8	7	15
100	5	two-point	flip	343.5	10.7	326.6	360.5	7	1.3	4	9
100	5	two-point	none	397.7	17.8	367.3	432.0	11	1.5	8	14
100	5	uniform	flip	352.4	12.6	327.2	374.4	7	1.3	4	9
100	5	uniform	none	371.1	12.2	348.6	410.4	8	1.2	5	9
100	20	one-point	flip	341.8	12.0	328.4	361.0	7	1.0	4	9
100	20	one-point	none	385.7	13.9	347.8	413.7	11	1.5	7	12
100	20	two-point	flip	343.6	11.8	323.1	362.9	7	1.0	6	9
100	20	two-point	none	398.6	16.8	378.8	431.6	11	1.7	8	14
100	20	uniform	flip	345.9	11.8	326.4	364.2	7	1.3	4	9
100	20	uniform	none	363.7	14.3	334.1	386.3	7	1.3	5	9
200	5	one-point	flip	357.9	12.2	329.6	369.8	8	1.4	4	9
200	5	one-point	none	369.5	15.8	342.7	406.6	9	1.0	6	10
200	5	two-point	flip	362.1	8.7	345.3	377.6	7	1.2	4	9
200	5	two-point	none	380.5	14.6	353.0	414.2	9	1.2	7	12
200	5	uniform	flip	360.3	9.5	345.6	385.3	7	1.3	4	9
200	5	uniform	none	351.1	12.0	328.9	375.2	7	1.1	3	8
200	20	one-point	flip	351.7	11.1	330.7	373.8	7	1.0	5	9
200	20	one-point	none	368.5	9.0	351.7	388.1	8	1.0	6	11
200	20	two-point	flip	352.9	10.8	335.8	373.7	8	0.9	5	9
200	20	two-point	none	383.7	14.0	349.9	406.2	9	1.2	7	12
200	20	uniform	flip	365.7	8.4	349.3	385.1	8	0.7	6	9
200	20	uniform	none	355.9	11.0	327.8	368.7	6	1.1	3	8

Table A.2: SEA with the binary representation and constraint $N_p^{\text{sw}} \leq 3$ for the Richmond network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	flip	93.8	2.6	91.4	100.2	26	3.1	19	32
50	5	one-point	none	163.5	15.9	122.3	178.9	39	3.0	36	46
50	5	two-point	flip	96.2	3.5	90.6	102.6	25	3.8	19	34
50	5	two-point	none	150.4	17.9	129.8	195.5	40	4.1	36	51
50	5	uniform	flip	95.4	4.0	90.2	107.5	25	3.5	18	29
50	5	uniform	none	134.9	21.0	112.8	185.4	37	4.6	29	46
50	20	one-point	flip	96.0	2.7	92.8	102.7	26	2.6	21	30
50	20	one-point	none	148.6	22.9	128.6	205.9	39	2.8	31	43
50	20	two-point	flip	98.0	4.5	92.9	110.1	28	2.1	25	31
50	20	two-point	none	138.0	10.2	124.5	161.0	40	3.5	33	46
50	20	uniform	flip	98.2	4.4	91.3	103.9	23	3.4	17	29
50	20	uniform	none	125.1	16.7	103.5	162.4	33	2.9	31	39
100	5	one-point	flip	102.2	5.7	98.1	119.9	24	3.5	19	31
100	5	one-point	none	145.1	13.6	121.8	169.2	38	3.4	32	42
100	5	two-point	flip	101.9	4.6	95.4	110.1	24	3.4	21	35
100	5	two-point	none	135.1	19.9	114.7	192.7	38	3.5	31	42
100	5	uniform	flip	102.2	6.2	92.2	115.7	20	2.7	15	25
100	5	uniform	none	121.9	10.5	110.8	152.6	30	4.7	21	41
100	20	one-point	flip	100.6	5.4	94.5	111.6	25	3.6	18	30
100	20	one-point	none	152.5	20.7	120.4	193.2	38	4.9	33	47
100	20	two-point	flip	103.8	4.8	95.6	112.5	24	3.7	20	32
100	20	two-point	none	133.7	12.3	122.6	167.5	34	4.2	27	41
100	20	uniform	flip	107.9	6.3	98.2	117.1	19	2.6	16	27
100	20	uniform	none	118.6	8.7	105.5	133.1	26	2.5	20	31
200	5	one-point	flip	112.4	6.9	102.3	127.8	25	3.2	19	28
200	5	one-point	none	127.5	9.4	118.0	153.2	32	3.3	25	41
200	5	two-point	flip	113.1	9.8	101.4	136.5	24	2.6	20	27
200	5	two-point	none	132.2	9.1	117.0	156.0	34	1.1	32	36
200	5	uniform	flip	121.8	14.0	112.1	154.3	22	2.7	19	27
200	5	uniform	none	119.9	12.1	102.0	138.1	20	2.2	18	24
200	20	one-point	flip	115.0	11.0	104.3	137.6	26	2.9	19	28
200	20	one-point	none	125.6	8.3	110.9	137.2	32	2.0	28	37
200	20	two-point	flip	109.8	8.7	104.0	135.3	25	2.7	20	28
200	20	two-point	none	127.9	11.7	112.7	153.0	33	2.9	29	39
200	20	uniform	flip	126.4	5.7	118.0	135.9	24	2.1	19	26
200	20	uniform	none	111.7	10.4	102.4	138.2	20	2.6	17	26

Table A.3: SEA with level-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Vanzyl network. All results use *extended-intermediate* recombination.

SEA parameters			C_E				N^sw			
α	μ	mutation	med	sd	min	max	med	sd	min	max
50	5	none	362.6	6.7	344.9	377.4	3	0.6	2	5
50	5	gaussian	353.9	11.6	336.6	382.3	3	0.9	2	5
50	5	replace	346.9	5.3	337.2	357.1	3	0.9	2	6
50	5	uniform	349.8	5.7	337.0	358.2	3	0.8	2	5
50	20	none	360.1	10.2	343.9	391.9	3	0.9	2	6
50	20	gaussian	353.6	7.7	340.7	372.0	3	1.3	2	7
50	20	replace	347.7	7.2	330.5	360.1	4	1.2	2	7
50	20	uniform	351.9	7.9	323.3	360.4	3	0.9	3	6
100	5	none	358.6	6.8	337.2	369.0	3	0.8	2	5
100	5	gaussian	351.7	7.2	329.9	360.4	3	1.0	2	5
100	5	replace	346.1	6.7	328.5	358.2	3	0.8	3	5
100	5	uniform	347.1	7.3	334.5	357.8	3	1.1	3	6
100	20	none	353.8	7.5	339.4	364.3	3	0.6	2	5
100	20	gaussian	352.7	5.8	339.3	361.2	3	0.9	2	5
100	20	replace	346.9	5.8	338.6	357.8	3	0.9	2	5
100	20	uniform	350.1	7.3	337.2	360.0	3	1.1	2	7
200	5	none	354.8	7.3	338.2	363.4	3	0.8	2	5
200	5	gaussian	353.1	5.1	341.9	360.4	4	0.9	2	5
200	5	replace	346.4	7.6	319.5	355.4	3	0.8	3	5
200	5	uniform	350.7	5.2	340.6	360.0	3	0.7	3	5
200	20	none	353.7	6.4	337.5	363.8	3	0.9	2	5
200	20	gaussian	347.0	6.2	336.8	360.4	3	0.8	3	5
200	20	replace	347.4	4.0	338.8	353.1	3	0.9	3	6
200	20	uniform	350.4	5.4	342.5	360.4	4	0.8	3	5

Table A.4: SEA with level-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Richmond network. All results use *extended-intermediate* recombination.

SEA parameters			C_E				N^sw			
α	μ	mutation	med	sd	min	max	med	sd	min	max
50	5	none	136.0	36.4	106.6	247.1	6	2.5	4	13
50	5	gaussian	120.9	38.5	106.4	215.5	8	2.6	4	13
50	5	replace	101.5	3.1	98.2	108.0	10	1.1	9	12
50	5	uniform	103.3	3.6	97.2	112.2	10	0.9	9	11
50	20	none	122.5	30.4	107.0	209.2	7	2.2	3	10
50	20	gaussian	124.2	25.4	103.2	201.6	9	2.3	4	12
50	20	replace	100.0	2.5	99.1	107.0	10	1.5	8	13
50	20	uniform	103.6	3.6	99.9	111.1	10	1.7	7	13
100	5	none	123.1	32.0	102.8	236.4	7	1.4	5	10
100	5	gaussian	121.0	7.9	110.3	133.2	6	1.6	4	9
100	5	replace	102.1	2.6	97.9	106.3	10	1.6	7	12
100	5	uniform	103.2	2.6	98.7	110.2	10	1.6	7	12
100	20	none	121.6	19.8	111.4	193.2	7	2.0	4	10
100	20	gaussian	117.2	22.5	100.8	194.5	8	1.9	4	11
100	20	replace	102.2	2.4	97.5	105.6	10	1.7	7	13
100	20	uniform	102.9	2.9	100.7	111.8	10	1.5	8	13
200	5	none	118.9	7.0	107.0	128.3	7	1.6	6	11
200	5	gaussian	118.5	8.8	101.9	140.1	8	1.9	5	11
200	5	replace	103.6	1.3	101.7	105.5	11	1.7	7	13
200	5	uniform	105.0	2.3	101.3	110.0	9	1.5	7	12
200	20	none	115.8	6.5	104.6	124.3	8	1.7	5	11
200	20	gaussian	114.8	6.4	105.5	125.0	9	1.9	5	11
200	20	replace	103.7	2.3	100.3	109.7	9	1.4	7	12
200	20	uniform	106.1	3.7	100.2	113.4	9	2.0	7	14

Table A.5: SEA with absolute time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Vanzyl network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	none	377.0	15.1	351.5	423.1	6	1.4	4	9
50	5	one-point	replace	336.4	8.2	327.2	358.5	6	0.9	4	8
50	5	one-point	uniform	353.7	13.5	327.1	367.5	5	0.9	4	7
50	5	rand-arithmetical	none	398.3	17.8	369.4	437.6	8	0.8	6	9
50	5	rand-arithmetical	replace	338.2	11.4	324.8	362.4	7	0.9	6	9
50	5	rand-arithmetical	uniform	344.2	10.8	329.2	363.8	9	0.8	7	9
50	5	two-point	none	369.3	11.3	347.0	394.3	6	1.4	4	9
50	5	two-point	replace	340.5	9.3	327.7	362.9	6	1.3	4	8
50	5	two-point	uniform	343.9	11.6	322.2	359.4	5	1.2	3	7
50	5	uniform	none	377.3	8.2	361.6	393.6	6	1.5	4	9
50	5	uniform	replace	358.6	9.1	339.3	373.1	7	1.0	4	8
50	5	uniform	uniform	361.8	10.1	331.8	372.8	6	1.2	4	9
50	20	one-point	none	373.1	9.8	359.2	395.8	7	1.1	5	8
50	20	one-point	replace	340.0	8.7	328.0	362.1	6	0.9	5	7
50	20	one-point	uniform	341.9	10.8	329.3	361.9	6	1.4	3	9
50	20	rand-arithmetical	none	388.3	14.0	370.3	425.8	8	0.8	6	9
50	20	rand-arithmetical	replace	342.1	9.4	330.0	363.8	8	1.0	5	9
50	20	rand-arithmetical	uniform	344.3	11.7	330.9	365.9	9	0.5	8	9
50	20	two-point	none	369.5	11.3	349.4	387.1	5	1.5	3	9
50	20	two-point	replace	338.7	5.6	325.3	351.2	6	1.3	4	8
50	20	two-point	uniform	337.7	12.5	325.3	365.9	5	1.4	3	8
50	20	uniform	none	371.3	9.8	355.9	393.1	5	1.2	3	8
50	20	uniform	replace	360.6	7.9	340.1	370.1	6	1.0	4	8
50	20	uniform	uniform	362.1	10.0	341.5	370.9	6	1.2	4	8
100	5	one-point	none	365.1	10.1	331.9	382.8	6	1.0	5	8
100	5	one-point	replace	340.9	8.0	329.7	364.3	6	0.9	4	8
100	5	one-point	uniform	337.3	8.5	323.4	357.6	6	1.0	4	8
100	5	rand-arithmetical	none	388.7	12.6	371.1	414.8	8	0.7	7	9
100	5	rand-arithmetical	replace	342.9	7.4	332.0	361.3	8	0.8	6	9
100	5	rand-arithmetical	uniform	348.8	10.8	332.1	365.5	9	0.5	7	9
100	5	two-point	none	356.7	10.4	341.5	380.2	6	0.9	4	8
100	5	two-point	replace	344.2	7.0	331.7	362.7	6	1.1	4	8
100	5	two-point	uniform	337.9	8.7	327.4	360.3	5	0.9	4	7
100	5	uniform	none	366.0	8.3	350.5	381.4	6	1.3	3	9
100	5	uniform	replace	367.5	7.7	348.4	378.5	7	0.8	5	8
100	5	uniform	uniform	365.6	4.8	345.4	370.5	6	1.4	3	8

(continues in next page . . .)

Table A.5: SEA with absolute time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Vanzyl network (continued from previous page).

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
100	20	one-point	none	368.7	8.2	356.6	390.4	7	1.5	4	9
100	20	one-point	replace	338.1	7.0	328.2	356.0	6	1	4	8
100	20	one-point	uniform	350.7	11.8	327.6	365.3	6	1.3	3	8
100	20	rand-arithmetical	none	388.1	14.0	360.1	410.7	9	0.6	7	9
100	20	rand-arithmetical	replace	341.8	8.8	332.4	364.5	8	0.8	6	9
100	20	rand-arithmetical	uniform	345.0	9.5	332.3	365.1	9	0.5	7	9
100	20	two-point	none	361.3	9.4	341.7	381.4	5	1.0	4	7
100	20	two-point	replace	344.6	7.6	331.0	359.0	6	1.0	3	8
100	20	two-point	uniform	339.2	9.8	326.8	359.1	5	1.1	3	8
100	20	uniform	none	369.4	9.3	343.6	384.3	6	1.0	4	8
100	20	uniform	replace	368.2	5.9	353.6	376.4	6	1.1	4	8
100	20	uniform	uniform	366.0	7.6	348.8	377.4	6	1.1	4	8
200	5	one-point	none	358.8	9.1	342.7	371.2	6	1.3	3	8
200	5	one-point	replace	345.3	6.8	333.2	361.1	6	1.3	4	9
200	5	one-point	uniform	347.8	8.3	328.2	362.0	6	1.3	3	8
200	5	rand-arithmetical	none	375.3	12.6	356.0	417.1	9	0.4	8	9
200	5	rand-arithmetical	replace	356.6	6.6	339.2	365.6	8	0.9	6	9
200	5	rand-arithmetical	uniform	351.9	9.9	338.1	367.3	9	0.5	7	9
200	5	two-point	none	349.4	10.2	333.8	367.2	5	0.7	4	6
200	5	two-point	replace	351.3	7.0	332.9	364.9	6	1	4	8
200	5	two-point	uniform	345.3	7.8	327.5	358.6	5	1.2	3	8
200	5	uniform	none	369.6	5.6	350.7	381.6	5	1.0	4	8
200	5	uniform	replace	367.6	7.4	352.7	381.1	6	1.1	4	7
200	5	uniform	uniform	369.8	4.2	356.5	377.6	7	1.1	5	8
200	20	one-point	none	357.9	8.7	335.2	370.7	6	1.2	3	8
200	20	one-point	replace	345.8	5.7	334.8	362.1	6	1.3	3	8
200	20	one-point	uniform	340.9	6.6	331.6	355.9	6	1.3	4	8
200	20	rand-arithmetical	none	378.2	8.6	367.1	399.7	9	0.5	7	9
200	20	rand-arithmetical	replace	351.2	7.5	339.5	366.5	8	0.9	6	9
200	20	rand-arithmetical	uniform	353.2	10.1	331.0	369.2	9	0.6	7	9
200	20	two-point	none	350.6	7.8	336.3	368.7	6	1.0	4	8
200	20	two-point	replace	348.3	6.6	333.1	358.4	6	0.8	5	7
200	20	two-point	uniform	347.2	8.6	330.0	362.5	5	1.2	3	7
200	20	uniform	none	367.6	5.8	353.6	376.9	6	1.0	4	7
200	20	uniform	replace	367.8	7.3	354.0	382.2	6	1.2	4	9
200	20	uniform	uniform	365.1	6.2	350.7	375.5	6	1.3	3	8

Table A.6: SEA with absolute time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Richmond network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	none	157.2	21.1	124.5	199.2	15	2.0	11	19
50	5	one-point	replace	95.5	3.4	90.4	104.2	13	1.1	12	15
50	5	one-point	uniform	97.1	3.9	91.7	106.2	16	1.1	14	17
50	5	rand-arithmetical	none	161.1	21.2	133.5	214.6	20	1.8	16	21
50	5	rand-arithmetical	replace	105.0	4.7	96.9	110.6	19	0.9	19	21
50	5	rand-arithmetical	uniform	96.1	7.8	94.1	124.1	21	0.4	20	21
50	5	two-point	none	159.9	17.1	117.1	174.2	14	2.0	11	18
50	5	two-point	replace	95.4	2.9	91.0	102.5	14	1.0	12	15
50	5	two-point	uniform	94.3	2.9	90.9	99.9	14	1.2	13	17
50	5	uniform	none	131.4	26.7	114.1	198.7	14	1.8	11	17
50	5	uniform	replace	109.8	7.2	100.2	124.1	17	1.5	14	19
50	5	uniform	uniform	103.6	4.2	93.6	109.9	15	1.6	12	18
50	20	one-point	none	158.4	14.7	129.6	178.5	15	1.8	13	19
50	20	one-point	replace	96.1	2.5	91.8	100.1	14	0.9	13	16
50	20	one-point	uniform	95.5	4.6	91.2	107.4	15	2.3	11	18
50	20	rand-arithmetical	none	162.7	19.1	124.5	199.3	21	0.9	18	21
50	20	rand-arithmetical	replace	102.6	5.8	96.9	113.2	20	1.2	17	21
50	20	rand-arithmetical	uniform	97.5	2.9	92.6	101.1	20	0.5	20	21
50	20	two-point	none	141.1	22.6	124.2	198.3	14	1.6	11	17
50	20	two-point	replace	95.5	2.2	92.1	100.9	13	1.6	10	16
50	20	two-point	uniform	98.3	5.8	90.0	106.3	14	2.0	11	19
50	20	uniform	none	122.1	31.5	111.2	211.4	13	2.0	10	17
50	20	uniform	replace	104.8	4.9	94.8	111.5	15	2.3	11	19
50	20	uniform	uniform	103.1	4.7	98.3	116.3	14	1.7	12	18
100	5	one-point	none	135.3	19.4	115.3	191.3	14	2.3	10	17
100	5	one-point	replace	96.4	3.1	91.0	102.3	14	1.5	12	17
100	5	one-point	uniform	98.1	5.0	94.1	112.7	15	2.6	12	20
100	5	rand-arithmetical	none	175.7	19.3	136.9	197.5	21	0.5	20	21
100	5	rand-arithmetical	replace	106.9	7.1	98.0	129.3	20	1.0	18	21
100	5	rand-arithmetical	uniform	97.7	3.0	92.4	102.3	21	0.4	20	21
100	5	two-point	none	126.7	10.5	112.2	148.7	13	1.1	12	15
100	5	two-point	replace	103.0	3.7	96.2	108.8	13	1.6	10	16
100	5	two-point	uniform	96.5	4.4	91.0	106.6	14	1.2	12	17
100	5	uniform	none	147.7	24.7	113.5	213.4	14	2.3	11	18
100	5	uniform	replace	116.3	7.8	107.4	135.4	15	2.1	10	19
100	5	uniform	uniform	116.8	7.3	105.0	131.1	14	1.7	12	18
100	20	one-point	none	129.8	13.3	109.8	160.3	15	2.0	12	18
100	20	one-point	replace	97.0	2.8	91.7	100.5	13	1.9	11	17
100	20	one-point	uniform	96.0	3.6	90.4	103.8	15	2.0	13	20
100	20	rand-arithmetical	none	159.8	15.4	129.5	176.9	20	1.0	18	21
100	20	rand-arithmetical	replace	104.7	12.5	98.1	148.7	20	0.9	18	21
100	20	rand-arithmetical	uniform	97.3	3.3	93.6	105.4	21	0.4	20	21
100	20	two-point	none	125.3	13.6	112.6	162.8	13	2.2	10	18
100	20	two-point	replace	101.9	2.9	96.9	106.7	14	1.7	11	17
100	20	two-point	uniform	98.0	4.2	88.8	107.1	14	1.6	13	18
100	20	uniform	none	126.0	21.7	103.4	168.8	15	2.0	10	18
100	20	uniform	replace	121.8	12.7	110.0	157.2	15	1.2	13	17
100	20	uniform	uniform	115.8	6.8	102.6	131.9	14	1.6	11	17
200	5	one-point	none	122.3	10.4	106.1	149.9	14	1.6	12	17
200	5	one-point	replace	105.8	3.7	99.0	111.4	16	1.6	13	19
200	5	one-point	uniform	101.0	5.0	94.8	113.5	15	1.5	13	19
200	5	rand-arithmetical	none	166.0	20.0	134.5	222.4	21	1.4	16	21
200	5	rand-arithmetical	replace	114.6	5.1	105.2	122.3	20	0.6	19	21
200	5	rand-arithmetical	uniform	106.1	4.1	97.7	109.9	21	0.5	19	21
200	5	two-point	none	117.7	9.9	97.7	134.4	13	1.4	11	15
200	5	two-point	replace	113.9	3.7	106.6	121.0	15	1.3	13	18
200	5	two-point	uniform	108.3	5.2	96.4	115.3	14	2.2	10	18
200	5	uniform	none	142.2	13.9	120.2	165.8	14	1.6	12	18
200	5	uniform	replace	135.5	14.8	115.2	164.9	15	1.9	12	19
200	5	uniform	uniform	140.8	11.2	119.7	157.0	15	2.0	11	19
200	20	one-point	none	119.1	7.8	109.8	138.4	14	2.4	10	18

(continues in next page ...)

Table A.6: SEA with absolute time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Richmond network (continued from previous page).

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
200	20	one-point	replace	104.1	3.7	100.9	112.0	15	2.2	12	18
200	20	one-point	uniform	101.6	4.0	96.3	111.2	16	1.9	12	18
200	20	rand-arithmetical	none	160.3	14.0	130.9	179.7	21	1.0	18	21
200	20	rand-arithmetical	replace	115.1	3.6	110.7	124.6	21	0.5	20	21
200	20	rand-arithmetical	uniform	105.0	3.3	100.4	113.3	21	0.3	20	21
200	20	two-point	none	117.5	7.7	103.1	131.1	12	1.5	11	16
200	20	two-point	replace	111.0	4.8	102.9	119.2	14	1.1	13	16
200	20	two-point	uniform	109.4	4.6	99.7	115.0	14	1.4	12	17
200	20	uniform	none	139.5	14.8	115.9	171.8	15	2.2	11	19
200	20	uniform	replace	135.4	9.4	121.5	152.2	14	1.3	12	17
200	20	uniform	uniform	137.3	18.5	113.7	174.8	15	1.9	11	17

Table A.7: SEA with relative time-controlled triggers representation and constraint $N_p^{\mathrm{sw}} \leq 3$ for the Vanzyl network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	none	371.1	11.5	340.6	394.2	5	2.2	1	9
50	5	one-point	replace	340.7	7.4	331.3	359.2	4	0.7	3	5
50	5	one-point	uniform	344.8	10.3	330.4	361.1	5	0.9	3	7
50	5	rand-arithmetical	none	374.7	7.2	358.9	387.9	9	0.6	7	9
50	5	rand-arithmetical	replace	336.7	10.8	325.0	361.0	5	1.1	4	7
50	5	rand-arithmetical	uniform	331.6	8.1	322.5	357.0	7	0.9	4	8
50	5	two-point	none	372.6	12.9	350.5	404.1	5	2.1	2	9
50	5	two-point	replace	352.5	9.6	328.1	362.0	4	0.8	3	5
50	5	two-point	uniform	342.9	11.7	328.3	363.0	5	0.9	3	6
50	5	uniform	none	361.4	8.7	341.4	385.8	5	1.5	2	8
50	5	uniform	replace	341.2	9.9	328.8	362.1	4	1.0	2	6
50	5	uniform	uniform	348.2	9.3	329.3	358.1	5	1.2	3	8
50	20	one-point	none	379.5	11.1	354.0	405.4	6	1.6	3	9
50	20	one-point	replace	344.5	11.2	323.4	362.8	3	0.7	2	5
50	20	one-point	uniform	339.5	12.6	323.1	363.6	5	1.0	3	6
50	20	rand-arithmetical	none	369.9	9.4	355.7	389.2	8	0.9	7	9
50	20	rand-arithmetical	replace	334.1	6.1	315.9	341.4	5	1.2	3	7
50	20	rand-arithmetical	uniform	332.2	10.8	324.6	356.0	6	1.1	4	9
50	20	two-point	none	374.7	9.7	356.1	400.5	5	1.7	4	9
50	20	two-point	replace	345.8	9.4	327.2	362.0	4	0.8	3	6
50	20	two-point	uniform	355.3	10.7	325.6	359.9	5	1.0	3	7
50	20	uniform	none	363.4	8.2	338.6	378.7	5	1.1	3	8
50	20	uniform	replace	344.4	9.3	332.2	360.0	4	0.6	3	5
50	20	uniform	uniform	353.5	10.9	328.2	361.3	5	0.6	4	6
100	5	one-point	none	368.6	8.2	348.5	387.7	4	1.6	2	9
100	5	one-point	replace	348.1	7.3	335.9	360.5	4	0.9	3	6
100	5	one-point	uniform	355.9	7.9	335.2	362.7	5	1.2	4	8
100	5	rand-arithmetical	none	364.2	7.6	348.5	375.6	9	0.8	7	9
100	5	rand-arithmetical	replace	335.4	7.2	323.2	355.0	6	1.3	3	8
100	5	rand-arithmetical	uniform	332.8	8.2	325.8	356.7	7	0.9	6	9
100	5	two-point	none	366.4	10.2	338.5	389.0	5	1.2	3	8
100	5	two-point	replace	349.0	7.8	331.1	361.0	4	0.8	3	5
100	5	two-point	uniform	355.4	9.1	331.5	365.2	5	0.8	3	6
100	5	uniform	none	359.3	7.2	339.9	368.1	4	0.8	2	5
100	5	uniform	replace	346.6	8.4	325.8	359.8	4	0.6	3	5
100	5	uniform	uniform	354.2	9.1	328.2	364.0	5	1.1	3	6
100	20	one-point	none	364.4	9.4	336.4	379.5	5	1.8	2	9
100	20	one-point	replace	351.5	8.3	333.0	363.1	4	0.8	3	5
100	20	one-point	uniform	353.7	10.1	330.5	362.9	5	0.9	3	6
100	20	rand-arithmetical	none	367.2	9.5	340.5	375.0	9	0.5	8	9
100	20	rand-arithmetical	replace	336.9	8.1	326.6	358.6	7	1.1	4	8
100	20	rand-arithmetical	uniform	330.5	4.8	323.0	342.8	7	0.8	6	9
100	20	two-point	none	364.6	8.4	339.3	382.5	5	1.8	2	9
100	20	two-point	replace	349.5	6.3	335.8	361.2	4	0.7	3	5
100	20	two-point	uniform	356.9	6.3	339.6	362.1	5	1.4	3	8
100	20	uniform	none	358.4	6.5	336.6	364.0	4	1.5	2	8
100	20	uniform	replace	351.7	5.9	340.1	360.3	4	0.7	3	6
100	20	uniform	uniform	356.3	5.9	341.1	363.9	5	1.4	3	9
200	5	one-point	none	363.4	8.6	334.3	371.1	4	1.3	2	7
200	5	one-point	replace	351.8	9.0	329.3	363.4	4	1.0	3	7
200	5	one-point	uniform	359.6	5.9	344.3	364.0	5	1.0	3	7
200	5	rand-arithmetical	none	355.3	9.0	339.6	373.2	9	0.7	7	9
200	5	rand-arithmetical	replace	341.9	6.0	331.2	353.8	6	1.1	4	8
200	5	rand-arithmetical	uniform	338.2	5.6	334.5	354.4	8	0.9	5	9
200	5	two-point	none	360.5	6.5	338.4	370.3	4	1.0	2	6
200	5	two-point	replace	351.4	8.8	327.2	364.4	4	0.8	3	6
200	5	two-point	uniform	358.6	5.1	349.5	368.1	4	1.0	3	7
200	5	uniform	none	358.3	5.4	344.1	365.8	4	0.7	3	5
200	5	uniform	replace	352.6	6.9	333.2	361.0	4	0.8	3	6
200	5	uniform	uniform	359.0	5.5	339.0	364.4	4	1.1	3	7
200	20	one-point	none	362.5	7.5	338.2	367.2	4	1.7	2	8

(continues in next page . . .)

Table A.7: SEA with relative time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Vanzyl network (continued from previous page).

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
200	20	one-point	replace	349.4	7.3	335.1	361.5	4	0.7	3	5
200	20	one-point	uniform	356.3	5.9	341.0	365.7	5	1.1	3	6
200	20	rand-arithmetical	none	352.4	8.8	340.3	375.2	9	0.5	7	9
200	20	rand-arithmetical	replace	340.8	5.3	325.9	349.7	6	1.3	4	9
200	20	rand-arithmetical	uniform	339.6	7.3	327.8	358.1	8	1.2	5	9
200	20	two-point	none	359.4	10.3	333.2	377.3	4	1.0	3	7
200	20	two-point	replace	352.5	7.1	334.5	362.3	4	0.7	3	5
200	20	two-point	uniform	361.8	5.9	345.5	367.0	5	1.2	3	7
200	20	uniform	none	357.3	6.7	343.3	365.1	4	0.8	3	5
200	20	uniform	replace	350.1	7.5	339.4	361.5	4	0.9	3	6
200	20	uniform	uniform	354.9	6.5	342.6	363.4	4	1.0	3	6

Table A.8: SEA with relative time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Richmond network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	none	174.7	24.5	136.6	221.6	19	1.0	17	21
50	5	one-point	replace	101.3	3.8	97.0	112.1	19	1.0	18	21
50	5	one-point	uniform	99.2	11.0	94.8	125.2	20	0.6	19	21
50	5	rand-arithmetical	none	140.3	21.3	125.9	189.6	20	1.1	18	21
50	5	rand-arithmetical	replace	93.9	3.5	90.5	104.0	13	1.7	11	16
50	5	rand-arithmetical	uniform	92.3	1.6	90.3	95.4	16	1.7	14	20
50	5	two-point	none	161.3	26.9	141.6	220.1	16	3.3	9	21
50	5	two-point	replace	96.9	2.7	92.7	101.7	10	1.4	8	13
50	5	two-point	uniform	100.4	3.9	91.0	104.3	13	2.4	9	16
50	5	uniform	none	154.5	32.4	123.9	243.6	13	3.6	8	20
50	5	uniform	replace	102.3	4.5	92.5	108.6	10	1.5	7	12
50	5	uniform	uniform	101.8	7.1	92.5	121.3	13	2.5	9	19
50	20	one-point	none	175.2	27.0	130.7	216.4	20	1.2	18	21
50	20	one-point	replace	100.2	3.6	97.1	108.2	19	0.9	18	21
50	20	one-point	uniform	98.3	5.2	93.6	110.6	20	0.6	19	21
50	20	rand-arithmetical	none	147.3	20.9	124.0	193.5	21	0.9	19	21
50	20	rand-arithmetical	replace	94.4	3.5	89.3	104.3	13	1.8	11	18
50	20	rand-arithmetical	uniform	93.6	1.5	91.6	97.4	16	1.7	12	18
50	20	two-point	none	168.2	29.6	132.9	226.2	17	3.0	10	20
50	20	two-point	replace	100.4	4.4	93.2	110.2	10	0.9	9	12
50	20	two-point	uniform	102.3	7.3	90.4	122.6	14	1.5	11	16
50	20	uniform	none	150.4	22.0	120.1	192.4	13	2.4	10	19
50	20	uniform	replace	102.0	5.4	92.9	109.6	10	1.3	8	12
50	20	uniform	uniform	101.8	6.3	94.8	121.5	14	2.2	10	18
100	5	one-point	none	168.5	24.7	127.0	211.8	19	1.1	17	21
100	5	one-point	replace	113.4	5.2	107.7	123.5	19	1.0	17	20
100	5	one-point	uniform	114.4	6.5	106.3	128.9	20	0.8	18	21
100	5	rand-arithmetical	none	140.3	14.5	123.8	173.9	21	0.5	20	21
100	5	rand-arithmetical	replace	100.1	4.9	93.6	111.2	15	2.5	11	19
100	5	rand-arithmetical	uniform	94.6	1.6	92.6	97.0	18	1.4	15	19
100	5	two-point	none	151.9	31.6	119.8	225.8	16	2.9	11	20
100	5	two-point	replace	112.8	4.7	108.2	125.7	10	1.9	6	13
100	5	two-point	uniform	117.5	5.8	107.8	125.2	12	2.4	7	15
100	5	uniform	none	132.6	13.8	112.4	164.0	11	3.6	9	20
100	5	uniform	replace	117.2	5.8	108.6	127.8	10	1.7	8	13
100	5	uniform	uniform	120.7	4.8	109.5	125.1	13	2.0	9	16
100	20	one-point	none	146.8	19.7	127.6	189.9	19	1.2	17	21
100	20	one-point	replace	112.2	5.1	104.4	120.4	18	1.3	17	21
100	20	one-point	uniform	115.5	6.9	107.6	128.7	20	1.0	18	21
100	20	rand-arithmetical	none	139.4	11.0	129.0	168.1	21	0.8	19	21
100	20	rand-arithmetical	replace	97.4	5.1	89.8	109.4	16	1.6	13	18
100	20	rand-arithmetical	uniform	95.1	2.0	91.5	98.9	18	1.4	16	20
100	20	two-point	none	168.0	25.3	130.5	209.2	15	4.1	6	20
100	20	two-point	replace	114.7	4.2	105.7	118.5	9	1.7	8	13
100	20	two-point	uniform	117.2	6.1	104.2	123.2	13	2.4	8	17
100	20	uniform	none	136.5	13.3	117.8	166.2	12	2.7	7	17
100	20	uniform	replace	114.8	4.0	109.6	125.5	9	1.5	6	11
100	20	uniform	uniform	120.4	5.7	102.6	125.4	12	1.5	10	16
200	5	one-point	none	135.1	14.4	120.9	165.3	14	3.3	8	19
200	5	one-point	replace	123.8	6.0	116.9	136.6	9	4.3	7	20
200	5	one-point	uniform	127.1	8.6	118.7	147.9	13	1.8	9	15
200	5	rand-arithmetical	none	134.5	9.1	120.6	155.9	21	0.4	20	21
200	5	rand-arithmetical	replace	106.8	3.7	97.8	111.8	17	2.3	10	19
200	5	rand-arithmetical	uniform	100.6	3.0	97.5	108.7	20	0.8	18	21
200	5	two-point	none	147.6	21.0	125.6	200.0	14	3.1	9	21
200	5	two-point	replace	127.3	6.3	116.3	137.8	9	2.5	6	17
200	5	two-point	uniform	131.1	8.1	116.0	143.3	14	2.6	10	21
200	5	uniform	none	135.1	13.4	122.8	172.3	13	4.0	9	21
200	5	uniform	replace	130.3	3.8	120.7	136.9	11	2.8	7	20
200	5	uniform	uniform	132.5	6.7	123.4	142.9	12	2.3	9	16
200	20	one-point	none	135.6	16.8	122.3	187.7	14	3.4	6	18

(continues in next page ...)

Table A.8: SEA with relative time-controlled triggers representation and constraint $N_p^{\text{sw}} \leq 3$ for the Richmond network (continued from previous page).

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
200	20	one-point	replace	130.9	5.6	121.6	140.9	11	2.1	7	15
200	20	one-point	uniform	127.2	8.2	104.4	138.9	12	2.3	8	17
200	20	rand-arithmetical	none	129.0	9.7	122.0	159.8	21	0.6	19	21
200	20	rand-arithmetical	replace	107.9	4.2	102.6	118.0	17	2.0	14	20
200	20	rand-arithmetical	uniform	100.5	3.1	94.9	105.8	19	1.0	18	21
200	20	two-point	none	132.9	12.5	116.8	156.6	15	2.9	7	19
200	20	two-point	replace	127.3	7.6	113.9	142.2	9	1.3	7	11
200	20	two-point	uniform	128.8	6.3	117.7	139.9	12	3.4	8	20
200	20	uniform	none	132.6	10.6	123.2	156.6	11	2.8	7	18
200	20	uniform	replace	129.4	6.7	114.6	138.6	10	4.0	8	20
200	20	uniform	uniform	132.1	6.2	118.1	140.2	11	2.3	8	16

Table A.9: Results of SEA with the binary representation and not constraint on number of pump switches for the Vanzyl network.

SEA parameters				C_{E}				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	20	one-point	flip	344.0	4.7	332.6	351.0	12	1.6	9	16
50	20	one-point	none	380.8	10.4	362.6	399.5	15	1.9	12	19
50	20	two-point	flip	345.7	4.3	338.7	357.8	13	2.1	9	18
50	20	two-point	none	381.3	16.2	355.9	424.6	17	1.7	12	20
50	20	uniform	flip	345.4	3.4	336.8	354.0	13	2.1	9	17
50	20	uniform	none	352.7	10.5	333.3	375.1	14	2.8	8	19
100	20	one-point	flip	350.7	4.3	343.6	358.3	13	1.6	10	16
100	20	one-point	none	371.5	10.6	351.2	394.8	15	2.0	11	19
100	20	two-point	flip	351.0	5.0	344.4	365.5	13	1.6	10	17
100	20	two-point	none	373.4	9.1	358.1	394.4	16	2.5	11	21
100	20	uniform	flip	349.7	4.6	337.5	357.4	13	1.6	10	16
100	20	uniform	none	338.7	9.9	327.8	358.4	11	1.8	8	16
200	20	one-point	flip	357.2	3.9	349.2	362.7	14	2.0	11	18
200	20	one-point	none	350.9	9.5	331.1	364.8	13	2.1	9	20
200	20	two-point	flip	358.4	3.9	350.6	365.6	15	2.5	10	19
200	20	two-point	none	360.9	7.1	349.6	372.3	14	1.8	10	18
200	20	uniform	flip	356.5	4.8	348.5	364.6	14	1.6	11	16
200	20	uniform	none	328.2	2.6	323.5	333.5	10	2.0	6	14

Table A.10: Results of SEA with the binary representation and not constraint on number of pump switches for the Richmond network.

SEA parameters				C_E				N^{sw}			
α	μ	recomb	mutation	med	sd	min	max	med	sd	min	max
50	5	one-point	flip	93.2	1.9	91.7	98.0	33	3.3	28	38
50	5	one-point	none	146.4	15.0	124.0	179.1	40	2.6	36	44
50	5	two-point	flip	94.0	1.3	91.8	95.7	31	3.0	28	38
50	5	two-point	none	146.9	20.0	119.2	183.1	42	3.4	37	46
50	5	uniform	flip	91.7	1.9	90.4	97.0	30	2.7	26	37
50	5	uniform	none	120.4	7.7	109.8	136.4	40	4.0	33	48
100	5	one-point	flip	94.9	1.5	91.8	96.6	32	2.4	28	37
100	5	one-point	none	126.7	12.6	113.6	155.8	40	3.5	34	45
100	5	two-point	flip	95.1	2.0	92.5	99.1	35	2.9	28	40
100	5	two-point	none	121.7	19.6	113.1	173.8	39	3.3	35	47
100	5	uniform	flip	93.6	1.0	92.3	95.9	31	2.6	26	35
100	5	uniform	none	106.4	8.5	97.8	124.3	38	4.3	31	46
200	20	one-point	flip	122.7	4.0	116.4	131.2	41	2.9	36	48
200	20	one-point	none	115.0	9.0	106.9	142.8	40	2.9	34	48
200	20	two-point	flip	117.4	2.1	114.1	121.8	40	2.3	36	44
200	20	two-point	none	115.1	5.1	108.2	129.8	39	2.4	33	43
200	20	uniform	flip	123.5	1.9	120.3	126.6	40	3.1	34	44
200	20	uniform	none	94.8	3.0	91.1	102.3	31	2.7	28	36

Appendix B

Results of Ant Colony Optimisation

Table B.1: Results of ACO for the Vanzyl network.

ACO parameters				C_{E} ($N_p^{\mathrm{sw}} = 3$)				C_{E} ($N_p^{\mathrm{sw}} \leq 3$)			
A	$Sel.$	ρ	Update	med.	sd.	best	worst	med.	sd.	best	worst
10	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	368.8	5.2	355.3	376.7	363.6	4.4	355.7	374.2
10	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	361.8	8.8	330.6	366.3	359.6	6.0	336.1	367.7
10	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	365.3	7.7	346.9	375.9	362.3	7.5	342.4	378.5
10	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	358.4	11.1	324.9	362.0	356.7	4.9	341.9	361.3
10	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	362.8	8.2	338.5	371.8	360.0	7.7	334.1	366.5
10	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	352.7	8.3	329.6	361.4	354.1	7.0	329.3	359.6
10	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	362.1	9.4	337.6	367.9	357.6	12.3	328.1	365.9
10	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	353.2	5.9	336.8	362.4	353.0	7.0	330.0	358.9
20	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	364.5	8.0	341.5	373.0	360.2	6.5	342.0	369.6
20	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	359.6	11.4	327.4	363.9	357.5	9.0	331.1	362.8
20	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	361.7	9.3	340.4	370.1	359.8	9.9	332.7	365.1
20	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	353.2	12.3	328.3	363.6	356.0	9.0	327.4	363.3
20	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	360.8	11.3	331.7	368.0	359.0	8.0	336.0	366.3
20	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	356.2	9.0	329.8	363.0	349.9	7.8	330.5	361.1
20	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	358.4	13.1	325.3	363.2	356.9	10.1	330.2	360.2
20	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	355.6	5.6	343.7	362.2	351.2	7.2	335.0	360.3
40	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	361.5	9.9	326.0	366.5	360.0	8.6	335.8	366.3
40	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	359.5	10.2	330.1	365.2	355.5	8.2	330.6	361.4
40	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	357.3	11.5	327.1	367.0	358.6	11.7	330.6	365.1
40	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	357.6	14.4	326.7	362.0	354.9	9.2	324.8	361.8
40	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	359.0	11.5	332.4	364.3	356.7	11.8	326.7	362.3
40	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	352.9	10.3	327.6	361.4	351.8	7.9	327.2	359.9
40	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	341.9	12.7	326.9	365.5	354.3	10.3	330.0	360.8
40	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	352.7	5.7	338.3	358.3	350.9	6.8	337.0	361.6
80	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	356.3	9.6	335.8	366.4	357.4	10.7	329.8	363.9
80	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	353.1	13.3	321.6	361.9	355.8	11.4	327.4	361.7
80	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	346.0	11.3	328.6	365.3	356.7	11.0	333.3	364.3
80	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	343.6	11.2	329.3	360.2	349.3	11.1	324.8	358.9
80	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	338.5	9.7	327.5	362.8	345.6	12.4	318.2	361.3
80	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	348.0	6.5	334.8	360.7	352.9	7.3	329.8	360.2
80	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	344.3	9.4	327.4	363.7	339.9	11.1	326.8	359.4
80	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	352.1	5.2	339.5	363.2	353.7	6.2	337.1	361.5
10	bf	0.85	$\Delta\tau = 1$	366.6	4.4	354.9	377.2	362.4	6.2	351.7	377.5
10	ib	0.85	$\Delta\tau = 1$	360.5	5.8	343.7	367.3	360.0	6.6	334.4	364.4
10	bf	0.90	$\Delta\tau = 1$	365.2	6.0	351.0	373.3	362.6	5.9	344.1	372.4
10	ib	0.90	$\Delta\tau = 1$	359.5	11.7	330.1	367.1	356.4	7.4	328.2	362.0
10	bf	0.95	$\Delta\tau = 1$	364.6	7.7	340.8	369.7	358.7	7.2	338.4	366.2
10	ib	0.95	$\Delta\tau = 1$	354.7	7.9	332.1	361.8	352.0	6.2	335.5	359.8
10	bf	0.98	$\Delta\tau = 1$	361.4	9.7	334.9	367.9	357.6	12.1	319.8	367.3
10	ib	0.98	$\Delta\tau = 1$	356.4	6.4	342.1	363.9	347.8	7.1	334.1	359.4
20	bf	0.85	$\Delta\tau = 1$	365.4	6.5	337.0	371.3	361.8	5.6	344.1	368.4
20	ib	0.85	$\Delta\tau = 1$	359.9	11.5	327.7	367.1	358.4	6.3	332.7	364.1
20	bf	0.90	$\Delta\tau = 1$	363.7	8.7	341.2	370.4	359.3	9.6	335.8	364.7
20	ib	0.90	$\Delta\tau = 1$	355.1	13.9	325.5	363.8	355.6	10.4	323.9	359.3
20	bf	0.95	$\Delta\tau = 1$	359.6	10.3	334.4	365.4	358.4	11.7	328.2	365.0
20	ib	0.95	$\Delta\tau = 1$	350.1	8.3	330.7	361.7	352.2	6.7	334.4	358.7
20	bf	0.98	$\Delta\tau = 1$	358.7	13.0	329.3	365.3	357.5	11.1	329.6	362.1
20	ib	0.98	$\Delta\tau = 1$	351.7	6.2	336.8	362.2	351.6	6.5	338.3	360.1
40	bf	0.85	$\Delta\tau = 1$	361.3	11.0	331.2	368.1	361.6	7.5	339.4	367.3
40	ib	0.85	$\Delta\tau = 1$	355.2	11.5	331.0	366.2	357.8	10.7	325.4	362.1
40	bf	0.90	$\Delta\tau = 1$	361.0	9.3	334.1	369.1	357.3	11.4	327.2	363.9
40	ib	0.90	$\Delta\tau = 1$	351.4	11.4	327.0	361.0	354.3	10.4	329.0	362.1
40	bf	0.95	$\Delta\tau = 1$	344.1	10.4	334.3	363.9	356.0	11.1	328.7	362.6
40	ib	0.95	$\Delta\tau = 1$	351.0	9.0	329.0	361.7	348.1	6.8	332.7	360.1
40	bf	0.98	$\Delta\tau = 1$	350.4	10.4	331.6	365.4	340.1	11.5	327.7	362.1
40	ib	0.98	$\Delta\tau = 1$	352.5	7.0	328.7	361.8	351.0	6.7	338.4	360.3
80	bf	0.85	$\Delta\tau = 1$	351.2	10.6	334.8	365.6	344.6	12.4	329.3	363.1
80	ib	0.85	$\Delta\tau = 1$	352.8	12.7	321.6	362.7	345.8	12.7	326.1	361.3
80	bf	0.90	$\Delta\tau = 1$	355.1	11.7	329.6	363.7	354.1	12.4	326.7	363.9
80	ib	0.90	$\Delta\tau = 1$	346.4	12.0	324.0	361.0	349.2	10.1	327.6	359.8
80	bf	0.95	$\Delta\tau = 1$	346.2	10.6	330.1	363.3	342.6	11.2	326.5	362.2
80	ib	0.95	$\Delta\tau = 1$	349.1	7.1	334.1	360.2	354.4	6.5	333.6	360.8
80	bf	0.98	$\Delta\tau = 1$	341.3	9.7	326.5	364.5	350.2	8.8	334.9	359.8
80	ib	0.98	$\Delta\tau = 1$	355.1	5.4	340.6	363.7	352.8	6.6	338.9	361.3

Table B.2: Results of ACO for the Richmond network.

ACO parameters				C_{E} ($N_p^{\mathrm{sw}} = 3$)				C_{E} ($N_p^{\mathrm{sw}} \leq 3$)			
A	$Sel.$	ρ	Update	med.	sd.	best	worst	med.	sd.	best	worst
10	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	110.6	6.0	99.0	120.8	117.2	10.0	103.3	136.1
10	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	99.0	3.7	95.4	107.4	95.6	5.4	88.1	105.8
10	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	112.2	6.6	101.5	124.0	111.9	3.5	107.5	118.5
10	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	93.8	2.4	91.4	100.8	90.5	3.1	88.7	99.1
10	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	102.7	4.5	98.5	113.6	103.9	3.9	95.0	110.6
10	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	102.2	3.6	97.1	108.5	102.3	3.4	94.4	108.1
10	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	99.4	3.7	93.2	106.4	98.1	6.5	91.4	115.8
10	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	108.9	3.1	105.9	115.8	110.2	3.2	104.6	117.1
20	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	107.9	7.6	99.5	129.2	107.3	6.8	96.8	121.3
20	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	95.7	2.4	92.2	101.4	94.3	3.5	90.2	102.4
20	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	103.0	7.1	92.0	118.2	101.5	5.9	97.3	114.1
20	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	94.1	3.6	90.6	102.3	91.0	2.7	89.0	97.6
20	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	98.9	3.8	93.3	108.7	98.6	3.4	94.5	105.2
20	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	102.9	3.1	96.1	106.1	100.9	3.2	98.3	111.0
20	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	96.5	3.0	93.1	103.9	95.6	6.0	89.1	109.2
20	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	109.2	2.3	104.7	111.8	108.8	3.5	106.0	117.5
40	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	101.2	4.5	92.5	108.9	103.5	7.1	94.5	116.7
40	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	94.7	1.8	91.2	97.1	93.9	3.2	89.2	98.5
40	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	98.2	3.5	94.1	107.6	102.9	5.2	93.7	108.6
40	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	92.9	1.7	89.6	96.5	90.8	2.1	88.2	95.6
40	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	96.1	1.9	91.6	98.0	96.0	4.0	90.5	104.8
40	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	102.3	2.8	98.9	109.3	105.2	3.3	97.7	111.7
40	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	99.1	3.5	94.0	103.7	96.6	3.2	93.0	101.4
40	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	110.0	2.9	103.4	115.4	111.5	2.5	107.2	115.1
80	bf	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	96.2	3.0	92.1	104.2	98.2	5.3	90.3	106.0
80	ib	0.85	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	94.0	2.6	89.0	99.3	90.5	2.7	89.0	99.0
80	bf	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	97.0	2.2	93.7	101.9	96.7	4.4	90.8	106.5
80	ib	0.90	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	95.5	3.7	91.6	103.5	95.0	2.6	91.3	100.2
80	bf	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	95.8	2.2	91.8	100.1	95.6	4.7	90.6	105.1
80	ib	0.95	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	107.3	3.2	100.6	111.5	107.8	2.3	105.3	112.2
80	bf	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	101.2	2.8	95.9	106.3	101.0	5.4	90.6	111.7
80	ib	0.98	$\Delta\tau(S) = C_{\mathrm{E}}^{\max}/C_{\mathrm{E}}(S)$	110.5	2.3	106.3	114.8	113.8	3.3	108.4	120.7
10	bf	0.85	$\Delta\tau = 1$	109.1	6.7	95.9	120.6	112.5	5.5	101.2	119.3
10	ib	0.85	$\Delta\tau = 1$	100.5	5.1	89.9	109.5	95.4	3.3	90.3	102.7
10	bf	0.90	$\Delta\tau = 1$	104.2	6.7	96.7	118.1	113.4	10.8	99.8	138.8
10	ib	0.90	$\Delta\tau = 1$	94.0	2.3	90.2	97.9	94.7	3.0	90.1	100.5
10	bf	0.95	$\Delta\tau = 1$	101.6	4.3	96.7	111.7	103.5	6.6	95.3	116.5
10	ib	0.95	$\Delta\tau = 1$	100.5	4.2	96.4	111.0	104.1	3.4	100.4	111.0
10	bf	0.98	$\Delta\tau = 1$	97.5	4.2	90.8	107.0	98.7	4.4	93.9	108.9
10	ib	0.98	$\Delta\tau = 1$	110.2	3.1	103.3	114.0	110.4	2.8	105.3	114.9
20	bf	0.85	$\Delta\tau = 1$	106.0	5.7	97.3	119.2	105.6	7.7	93.1	124.3
20	ib	0.85	$\Delta\tau = 1$	96.0	4.0	92.6	106.0	92.2	3.1	88.4	99.2
20	bf	0.90	$\Delta\tau = 1$	101.4	5.2	93.7	109.8	102.6	7.3	96.6	120.8
20	ib	0.90	$\Delta\tau = 1$	93.1	1.8	90.2	95.4	91.1	2.7	88.5	98.4
20	bf	0.95	$\Delta\tau = 1$	97.6	2.7	94.2	103.4	99.1	3.3	93.7	103.6
20	ib	0.95	$\Delta\tau = 1$	103.0	3.6	98.3	110.0	104.7	3.9	96.5	110.2
20	bf	0.98	$\Delta\tau = 1$	97.4	3.2	91.5	102.1	97.8	4.9	90.6	109.3
20	ib	0.98	$\Delta\tau = 1$	111.2	2.0	109.5	117.2	111.9	3.1	106.9	118.3
40	bf	0.85	$\Delta\tau = 1$	101.2	4.5	95.4	110.4	101.6	3.1	97.2	107.6
40	ib	0.85	$\Delta\tau = 1$	95.0	1.7	91.4	97.6	91.1	3.3	88.5	99.3
40	bf	0.90	$\Delta\tau = 1$	99.9	4.8	90.4	106.8	102.6	4.7	93.3	109.2
40	ib	0.90	$\Delta\tau = 1$	94.5	1.7	91.8	97.3	92.2	2.1	89.3	96.9
40	bf	0.95	$\Delta\tau = 1$	95.8	2.2	91.4	99.4	96.2	3.9	89.0	102.7
40	ib	0.95	$\Delta\tau = 1$	106.0	3.8	98.9	111.5	105.9	3.4	100.4	113.3
40	bf	0.98	$\Delta\tau = 1$	99.2	2.4	95.4	104.0	97.4	4.3	92.5	106.8
40	ib	0.98	$\Delta\tau = 1$	108.7	3.1	105.6	116.4	113.7	4.4	104.5	120.7
80	bf	0.85	$\Delta\tau = 1$	98.9	3.8	94.3	107.9	97.5	4.4	91.4	106.0
80	ib	0.85	$\Delta\tau = 1$	92.9	2.1	91.1	97.5	94.4	2.9	88.1	99.1
80	bf	0.90	$\Delta\tau = 1$	96.8	2.3	93.0	99.9	98.9	3.5	91.2	103.2
80	ib	0.90	$\Delta\tau = 1$	97.1	2.2	92.2	100.8	99.9	2.7	95.1	105.0
80	bf	0.95	$\Delta\tau = 1$	96.3	2.4	92.7	100.6	96.2	2.8	90.4	100.7
80	ib	0.95	$\Delta\tau = 1$	108.8	3.0	104.6	114.0	110.7	3.9	103.1	116.5
80	bf	0.98	$\Delta\tau = 1$	103.3	3.3	95.9	108.6	106.9	3.8	97.3	111.5
80	ib	0.98	$\Delta\tau = 1$	111.4	3.6	105.0	118.7	115.4	4.2	108.0	120.5

Table B.3: Results of MMAS for the Vanzyl network for constraint $N_p^{\mathrm{sw}} \leq 3$.

MMAS parameters				C_{E}				N^{sw}			
A	$Sel.$	ρ	p_{best}	med.	sd.	best	worst	med.	sd.	best	worst
10	bf	0.85	0.05	343.8	10.9	327.7	360.7	4	1.2	3	7
10	ib	0.85	0.05	354.9	6.8	339.3	362.7	4	1.0	3	6
10	bf	0.85	0.5	355.7	10.1	334.1	363.1	4	1.1	3	7
10	ib	0.85	0.5	353.0	6.7	331.7	360.6	4	1.2	3	7
10	bf	0.85	0.7	358.5	9.8	336.3	369.5	5	1.0	3	7
10	ib	0.85	0.7	355.0	4.9	345.6	362.7	4	1.1	3	6
10	bf	0.85	0.9	359.3	8.7	334.2	364.8	4	1.0	3	7
10	ib	0.85	0.9	358.4	6.0	332.8	363.6	4	1.1	2	7
10	bf	0.85	0.9999	363.4	7.9	339.8	379.4	4	1.0	3	6
10	ib	0.85	0.9999	358.5	4.5	342.2	363.6	5	1.0	3	7
10	bf	0.90	0.05	340.7	9.4	327.7	358.6	5	0.9	3	6
10	ib	0.90	0.05	351.2	7.2	334.9	365.4	4	1.0	2	6
10	bf	0.90	0.5	357.3	11.5	324.7	363.6	4	1.1	3	7
10	ib	0.90	0.5	350.6	5.4	340.0	359.2	5	0.8	3	6
10	bf	0.90	0.7	359.0	10.3	330.8	366.5	4	1.1	2	6
10	ib	0.90	0.7	356.7	7.4	335.8	360.4	4	0.9	3	6
10	bf	0.90	0.9	361.2	10.4	328.2	365.5	4	1.1	2	6
10	ib	0.90	0.9	356.7	9.5	328.1	359.8	4	1.0	3	7
10	bf	0.90	0.9999	362.4	7.7	340.4	370.9	4	1.1	2	7
10	ib	0.90	0.9999	355.6	4.6	341.3	365.7	5	0.9	3	6
10	bf	0.95	0.05	341.7	9.1	327.0	358.5	5	1.3	3	9
10	ib	0.95	0.05	356.0	5.8	342.3	363.2	4	1.0	3	6
10	bf	0.95	0.5	353.9	10.9	331.4	363.1	4	0.9	2	6
10	ib	0.95	0.5	350.8	5.9	339.7	360.7	4	1.2	2	8
10	bf	0.95	0.7	358.0	12.7	327.1	365.4	4	1.4	2	7
10	ib	0.95	0.7	350.8	5.1	341.8	359.2	4	1.1	3	6
10	bf	0.95	0.9	356.0	9.7	332.3	361.6	4	0.9	3	6
10	ib	0.95	0.9	354.1	8.0	334.2	359.5	5	1.0	3	7
10	bf	0.95	0.9999	359.4	5.4	344.3	365.0	4	1.0	3	6
10	ib	0.95	0.9999	350.6	6.5	337.3	358.8	5	1.0	3	7
10	bf	0.98	0.05	344.7	11.2	327.6	359.3	5	1.1	3	7
10	ib	0.98	0.05	357.9	5.8	346.7	365.1	4	1.0	2	6
10	bf	0.98	0.5	349.8	10.0	328.2	363.1	4	0.8	3	6
10	ib	0.98	0.5	356.3	8.1	326.3	362.6	4	1.0	3	6
10	bf	0.98	0.7	346.7	12.1	329.8	363.5	5	0.8	3	6
10	ib	0.98	0.7	353.3	5.3	340.9	360.0	4	1.0	3	6
10	bf	0.98	0.9	354.9	11.4	327.7	362.7	4	0.9	3	6
10	ib	0.98	0.9	354.5	4.7	341.4	360.7	4	1.0	3	7
10	bf	0.98	0.9999	358.1	10.1	327.7	363.6	4	1.1	2	6
10	ib	0.98	0.9999	353.1	5.9	340.3	359.3	4	1.0	3	7
20	bf	0.85	0.05	345.3	11.3	328.2	361.0	5	0.9	4	7
20	ib	0.85	0.05	354.2	5.9	341.6	360.9	4	0.8	3	6
20	bf	0.85	0.5	356.7	10.3	327.4	363.1	4	0.9	3	6
20	ib	0.85	0.5	356.4	6.8	327.4	359.6	5	1.0	3	7
20	bf	0.85	0.7	358.3	9.1	327.7	363.1	4	1.0	3	6
20	ib	0.85	0.7	356.1	10.6	324.7	361.7	4	0.9	3	6
20	bf	0.85	0.9	358.6	10.9	327.1	365.5	4	1.0	2	7
20	ib	0.85	0.9	358.0	6.0	333.6	363.9	5	1.1	2	6
20	bf	0.85	0.9999	359.1	9.1	338.2	365.9	4	1.1	2	7
20	ib	0.85	0.9999	356.9	8.3	329.5	362.7	4	1.0	3	6
20	bf	0.90	0.05	342.9	11.6	327.7	359.9	5	0.9	3	7
20	ib	0.90	0.05	353.3	6.7	331.6	360.4	4	0.9	3	6
20	bf	0.90	0.5	356.0	11.3	327.7	363.1	4	0.8	3	5
20	ib	0.90	0.5	355.6	6.1	336.5	359.9	4	0.6	3	5
20	bf	0.90	0.7	353.4	10.8	333.3	363.6	5	0.7	3	6
20	ib	0.90	0.7	354.2	7.7	332.1	359.7	5	0.9	3	7
20	bf	0.90	0.9	359.3	10.9	330.2	368.2	4	0.9	3	6
20	ib	0.90	0.9	356.9	8.2	331.5	361.3	4	1.0	3	6
20	bf	0.90	0.9999	359.9	10.9	332.0	366.6	4	1.0	2	6
20	ib	0.90	0.9999	356.0	5.2	343.7	362.1	5	0.9	3	6
20	bf	0.95	0.05	350.3	10.3	329.8	360.7	4	1.2	3	7
20	ib	0.95	0.05	355.4	6.4	334.5	362.4	4	1.0	3	7

(continues in next page . . .)

Table B.3: Results of MMAS for the Vanzyl network for constraint $N_p^{\mathrm{sw}} \leq 3$ (continued from previous page).

MMAS parameters				C_{E}				N^{sw}			
A	$Sel.$	ρ	p_{best}	med.	sd.	best	worst	med.	sd.	best	worst
20	bf	0.95	0.5	347.4	12.8	327.0	368.7	4	0.7	3	5
20	ib	0.95	0.5	354.4	5.5	341.2	360.5	4	1.0	3	7
20	bf	0.95	0.7	347.4	11.5	326.5	363.6	5	1.0	3	7
20	ib	0.95	0.7	354.1	7.2	331.7	359.5	5	0.9	3	7
20	bf	0.95	0.9	357.1	11.5	322.7	362.1	4	1.4	2	7
20	ib	0.95	0.9	352.2	5.4	340.5	359.5	4	0.9	3	6
20	bf	0.95	0.9999	356.9	8.6	335.7	365.3	4	1.2	2	7
20	ib	0.95	0.9999	352.0	5.3	341.8	360.3	5	1.1	3	7
20	bf	0.98	0.05	349.3	9.9	330.6	360.1	5	1.1	3	7
20	ib	0.98	0.05	359.3	4.8	345.3	363.8	4	1.0	3	6
20	bf	0.98	0.5	348.7	10.6	328.7	362.1	5	0.8	3	6
20	ib	0.98	0.5	357.2	9.5	327.4	363.9	4	1.0	2	6
20	bf	0.98	0.7	344.5	9.7	331.9	361.7	4	0.7	3	6
20	ib	0.98	0.7	356.7	7.4	332.2	364.5	4	0.9	3	6
20	bf	0.98	0.9	341.0	12.0	325.8	361.2	4	1.2	3	8
20	ib	0.98	0.9	354.9	5.0	343.8	362.7	5	1.1	3	7
20	bf	0.98	0.9999	341.6	11.0	323.1	359.4	5	0.9	3	6
20	ib	0.98	0.9999	358.4	6.0	341.8	362.7	4	1.0	3	7
40	bf	0.85	0.05	340.8	10.1	322.4	359.8	5	1.0	3	7
40	ib	0.85	0.05	353.6	6.5	337.7	361.6	4	0.9	3	6
40	bf	0.85	0.5	350.2	12.1	325.7	363.1	4	0.9	2	6
40	ib	0.85	0.5	354.3	8.8	332.4	363.1	5	1.0	3	6
40	bf	0.85	0.7	342.5	11.6	326.5	361.4	5	1.0	3	7
40	ib	0.85	0.7	355.7	11.6	325.8	359.8	4	1.1	3	6
40	bf	0.85	0.9	358.0	9.9	326.3	363.1	4	1.3	2	8
40	ib	0.85	0.9	356.0	8.1	329.6	359.4	4	1.0	3	6
40	bf	0.85	0.9999	358.9	11.1	327.4	365.3	4	0.8	3	6
40	ib	0.85	0.9999	356.3	9.4	330.0	363.2	4	0.9	3	6
40	bf	0.90	0.05	345.9	12.2	327.6	359.8	4	1.0	3	6
40	ib	0.90	0.05	352.3	6.5	336.1	362.7	4	0.7	3	6
40	bf	0.90	0.5	354.6	12.7	323.2	362.5	4	0.8	3	6
40	ib	0.90	0.5	355.1	5.1	339.3	361.0	5	0.8	3	6
40	bf	0.90	0.7	355.6	12.8	324.2	362.1	5	1.1	3	8
40	ib	0.90	0.7	346.9	9.7	327.2	360.7	5	0.8	3	6
40	bf	0.90	0.9	352.6	9.2	334.6	361.7	4	0.8	3	6
40	ib	0.90	0.9	350.3	10.6	325.0	360.4	4	1.0	3	6
40	bf	0.90	0.9999	358.4	11.2	326.5	363.9	4	1.1	2	6
40	ib	0.90	0.9999	352.4	9.4	327.7	361.2	4	0.7	3	6
40	bf	0.95	0.05	346.2	8.8	331.2	359.4	4	1.0	3	6
40	ib	0.95	0.05	357.1	4.2	345.0	363.3	4	0.9	3	6
40	bf	0.95	0.5	344.9	12.7	326.0	363.1	5	1.0	3	6
40	ib	0.95	0.5	353.8	7.7	333.8	361.0	5	1.1	3	6
40	bf	0.95	0.7	345.3	12.2	327.4	361.2	5	0.7	3	6
40	ib	0.95	0.7	353.9	5.7	339.1	362.3	4	0.9	3	6
40	bf	0.95	0.9	340.3	9.6	328.7	362.1	4	1.2	2	7
40	ib	0.95	0.9	355.4	7.1	336.8	360.8	5	1.0	3	6
40	bf	0.95	0.9999	339.7	13.8	325.0	362.1	4	0.7	3	5
40	ib	0.95	0.9999	356.1	6.9	339.0	360.1	5	0.8	3	6
40	bf	0.98	0.05	358.5	6.5	343.0	363.7	4	0.9	3	6
40	ib	0.98	0.05	358.1	4.2	348.3	363.6	5	0.8	3	6
40	bf	0.98	0.5	359.1	6.8	333.4	364.2	5	1.1	3	7
40	ib	0.98	0.5	358.4	4.5	349.2	365.3	5	1.0	3	7
40	bf	0.98	0.7	359.4	7.0	338.5	364.1	4	1.2	2	7
40	ib	0.98	0.7	359.1	10.4	323.4	365.5	4	1.1	3	7
40	bf	0.98	0.9	358.6	5.1	345.3	363.5	4	1.1	3	7
40	ib	0.98	0.9	361.0	6.9	336.7	366.8	5	0.8	3	6
40	bf	0.98	0.9999	359.9	6.7	344.3	365.8	5	1.2	3	7
40	ib	0.98	0.9999	357.0	8.8	335.0	368.1	4	1.1	2	6
80	bf	0.85	0.05	345.3	9.6	332.8	359.0	4	1.0	3	7
80	ib	0.85	0.05	352.3	8.6	335.7	361.5	4	1.1	3	7
80	bf	0.85	0.5	356.4	13.2	325.0	363.1	5	1.2	3	8

(continues in next page . . .)

Table B.3: Results of MMAS for the Vanzyl network for constraint $N_p^{\text{sw}} \leq 3$ (continued from previous page).

MMAS parameters				C_{E}				N^{sw}			
A	$Sel.$	ρ	p_{best}	med.	sd.	best	worst	med.	sd.	best	worst
80	ib	0.85	0.5	351.2	9.7	330.2	361.9	4	1.0	3	7
80	bf	0.85	0.7	344.6	11.1	327.2	360.8	4	1.0	3	6
80	ib	0.85	0.7	352.9	10.4	329.6	359.8	4	1.0	3	6
80	bf	0.85	0.9	357.9	12.3	324.7	363.6	4	1.1	3	6
80	ib	0.85	0.9	350.7	10.1	325.3	361.2	4	1.3	3	8
80	bf	0.85	0.9999	356.3	9.6	334.4	363.9	4	0.9	2	6
80	ib	0.85	0.9999	352.9	8.8	331.9	359.3	4	1.0	3	6
80	bf	0.90	0.05	344.5	10.4	329.6	362.0	4	1.2	3	9
80	ib	0.90	0.05	355.4	7.0	338.1	361.1	4	0.8	3	6
80	bf	0.90	0.5	340.1	12.1	323.3	361.8	5	0.9	3	6
80	ib	0.90	0.5	355.4	7.5	335.2	360.4	4	1.1	3	7
80	bf	0.90	0.7	343.8	12.1	324.6	364.3	5	0.9	3	6
80	ib	0.90	0.7	353.9	7.4	333.4	360.4	4	0.8	3	6
80	bf	0.90	0.9	344.9	10.4	329.4	362.2	4	1.1	3	7
80	ib	0.90	0.9	345.5	8.5	328.7	359.8	4	0.8	3	6
80	bf	0.90	0.9999	344.1	12.1	325.7	362.1	5	0.9	2	6
80	ib	0.90	0.9999	353.5	7.5	329.4	359.4	4	1.0	3	7
80	bf	0.95	0.05	353.4	10.6	325.9	366.0	4	1.0	3	7
80	ib	0.95	0.05	357.3	7.8	339.6	364.7	4	1.0	3	6
80	bf	0.95	0.5	359.3	7.6	337.9	364.4	4	1.0	3	6
80	ib	0.95	0.5	358.9	6.8	340.2	365.0	4	1.0	3	6
80	bf	0.95	0.7	356.5	7.4	337.0	364.4	4	1.0	3	6
80	ib	0.95	0.7	357.4	7.5	338.6	365.9	5	1.1	3	7
80	bf	0.95	0.9	354.9	9.8	326.2	363.8	4	1.0	3	6
80	ib	0.95	0.9	359.4	5.8	345.0	364.5	4	1.0	3	6
80	bf	0.95	0.9999	353.3	7.9	333.0	361.9	5	1.2	3	6
80	ib	0.95	0.9999	356.3	6.4	341.1	363.7	4	1.3	3	8
80	bf	0.98	0.05	359.7	8.7	330.4	365.5	4	1.2	2	7
80	ib	0.98	0.05	360.2	5.8	344.2	366.9	4	1.2	2	7
80	bf	0.98	0.5	358.4	6.9	342.3	366.2	4	1.0	3	7
80	ib	0.98	0.5	359.1	3.9	351.7	367.5	5	1.1	3	7
80	bf	0.98	0.7	359.6	4.3	347.0	365.2	4	1.0	3	6
80	ib	0.98	0.7	360.8	4.8	350.2	365.9	4	1.2	2	7
80	bf	0.98	0.9	359.4	6.5	343.1	364.5	5	1.1	3	8
80	ib	0.98	0.9	360.1	6.4	339.4	366.6	5	1.2	3	7
80	bf	0.98	0.9999	359.3	6.3	341.4	365.0	5	1.3	3	7
80	ib	0.98	0.9999	361.2	4.8	349.0	366.9	5	1.2	3	8

Table B.4: Results of MMAS for the Richmond network for constraint $N_p^{\text{sw}} \leq 3$.

MMAS parameters				C_{E}				N^{sw}			
A	$Sel.$	ρ	p_{best}	med.	sd.	best	worst	med.	sd.	best	worst
10	bf	0.85	0.05	91.7	1.6	90.3	95.0	12	1.4	10	14
10	ib	0.85	0.05	115.8	4.1	110.5	125.4	10	1.8	7	12
10	bf	0.85	0.5	93.5	3.6	89.1	101.8	11	2.0	8	15
10	ib	0.85	0.5	100.2	5.2	90.0	110.3	12	1.8	9	15
10	bf	0.85	0.7	93.8	5.1	89.2	107.2	11	2.0	6	13
10	ib	0.85	0.7	90.9	2.1	88.7	95.3	12	1.2	10	14
10	bf	0.85	0.9	98.1	4.3	91.1	109.7	12	1.6	8	14
10	ib	0.85	0.9	91.7	3.2	89.2	97.2	11	1.6	9	14
10	bf	0.85	0.9999	108.4	13.1	94.1	148.2	12	2.8	6	17
10	ib	0.85	0.9999	95.4	3.3	90.3	99.7	12	1.3	9	14
10	bf	0.90	0.05	92.7	3.7	88.9	103.1	12	1.1	10	14
10	ib	0.90	0.05	117.8	3.6	111.8	123.5	10	1.9	7	13
10	bf	0.90	0.5	92.0	3.4	88.3	100.7	11	1.7	9	15
10	ib	0.90	0.5	106.9	3.4	96.8	110.9	12	2.0	9	15
10	bf	0.90	0.7	92.1	4.4	88.5	106.1	12	1.8	10	16
10	ib	0.90	0.7	102.9	3.5	96.3	108.2	12	2.1	8	15
10	bf	0.90	0.9	95.3	5.2	91.6	108.5	11	1.5	9	14
10	ib	0.90	0.9	96.9	2.9	90.8	99.5	12	2.0	9	16
10	bf	0.90	0.9999	108.2	7.2	101.0	124.6	10	1.2	7	12
10	ib	0.90	0.9999	92.5	2.6	90.2	99.8	12	2.1	10	17
10	bf	0.95	0.05	92.9	2.3	90.0	97.3	11	1.5	9	14
10	ib	0.95	0.05	114.9	3.9	106.0	120.6	10	1.3	7	12
10	bf	0.95	0.5	92.2	3.6	89.6	102.1	11	1.2	8	13
10	ib	0.95	0.5	110.7	3.6	104.9	117.3	11	1.9	9	15
10	bf	0.95	0.7	92.5	4.8	90.1	109.1	11	1.8	9	16
10	ib	0.95	0.7	106.5	3.0	103.3	114.2	13	1.8	11	17
10	bf	0.95	0.9	94.7	2.9	90.4	100.8	11	1.9	7	15
10	ib	0.95	0.9	108.0	3.0	101.0	111.6	12	1.1	10	14
10	bf	0.95	0.9999	103.5	5.5	92.4	113.8	10	1.8	6	13
10	ib	0.95	0.9999	104.7	4.1	97.4	110.4	13	1.9	8	15
10	bf	0.98	0.05	93.9	2.7	90.0	99.5	12	1.8	9	14
10	ib	0.98	0.05	122.1	3.5	115.6	130.5	10	1.9	7	13
10	bf	0.98	0.5	95.0	4.0	89.2	105.7	12	1.5	8	13
10	ib	0.98	0.5	116.9	4.7	104.5	125.0	11	1.5	8	13
10	bf	0.98	0.7	94.8	3.8	89.3	103.4	12	1.9	9	16
10	ib	0.98	0.7	115.8	3.0	109.0	118.5	10	1.2	8	12
10	bf	0.98	0.9	94.7	4.0	88.5	103.8	11	1.2	9	13
10	ib	0.98	0.9	112.8	3.6	108.1	120.0	10	1.8	8	14
10	bf	0.98	0.9999	100.0	4.8	91.1	107.7	10	1.0	9	12
10	ib	0.98	0.9999	115.5	3.5	108.1	120.3	11	2.1	7	16
20	bf	0.85	0.05	91.6	2.9	90.5	100.8	12	1.9	10	16
20	ib	0.85	0.05	111.9	3.3	105.8	118.4	12	1.1	10	13
20	bf	0.85	0.5	93.8	2.7	88.8	98.9	11	1.2	9	14
20	ib	0.85	0.5	91.3	2.2	88.0	96.4	12	1.5	9	14
20	bf	0.85	0.7	94.4	2.4	91.2	99.7	11	1.9	9	16
20	ib	0.85	0.7	90.3	1.6	88.3	93.7	12	2.0	9	15
20	bf	0.85	0.9	94.3	3.8	90.3	102.6	10	1.6	9	14
20	ib	0.85	0.9	91.8	4.8	88.8	109.5	11	0.9	10	13
20	bf	0.85	0.9999	102.7	7.5	94.5	121.3	11	1.8	8	13
20	ib	0.85	0.9999	95.9	3.7	89.1	100.0	10	1.4	9	13
20	bf	0.90	0.05	92.4	2.5	88.3	97.8	11	1.5	10	14
20	ib	0.90	0.05	112.7	3.8	106.2	118.8	10	1.6	8	13
20	bf	0.90	0.5	93.7	3.1	88.9	97.6	11	1.2	9	13
20	ib	0.90	0.5	98.7	2.6	94.6	105.8	11	1.8	9	15
20	bf	0.90	0.7	92.4	3.1	90.8	102.1	12	1.7	9	15
20	ib	0.90	0.7	96.9	3.5	90.8	102.0	11	2.4	9	18
20	bf	0.90	0.9	97.0	4.6	91.5	105.6	11	1.6	7	13
20	ib	0.90	0.9	91.6	2.2	89.0	96.9	11	1.2	9	13
20	bf	0.90	0.9999	105.2	6.7	91.3	117.9	11	1.7	8	14
20	ib	0.90	0.9999	91.9	1.7	88.8	96.0	13	1.4	9	15
20	bf	0.95	0.05	96.0	3.5	90.6	102.1	11	1.0	10	13
20	ib	0.95	0.05	114.1	4.3	108.1	124.5	11	1.5	8	14

(continues in next page . . .)

Table B.4: Results of MMAS for the Richmond network for constraint $N_p^{\text{sw}} \leq 3$ (continued from previous page).

MMAS parameters				C_{E}				N^{sw}			
A	$Sel.$	ρ	p_{best}	med.	sd.	best	worst	med.	sd.	best	worst
20	bf	0.95	0.5	91.6	4.6	88.4	104.8	11	1.5	9	14
20	ib	0.95	0.5	107.9	3.0	104.0	113.8	12	1.1	10	13
20	bf	0.95	0.7	93.0	2.9	88.5	98.6	11	1.6	8	13
20	ib	0.95	0.7	106.7	3.3	102.2	113.0	11	1.7	8	14
20	bf	0.95	0.9	94.4	5.0	90.8	106.3	11	1.6	7	13
20	ib	0.95	0.9	105.5	2.9	101.2	109.8	11	1.9	9	15
20	bf	0.95	0.9999	95.2	3.2	90.7	101.5	11	1.6	8	14
20	ib	0.95	0.9999	104.9	3.7	98.4	111.1	13	1.2	11	15
20	bf	0.98	0.05	100.7	3.7	94.3	107.3	11	1.0	9	13
20	ib	0.98	0.05	122.5	5.1	111.2	128.7	10	1.4	7	12
20	bf	0.98	0.5	102.1	3.4	93.8	103.2	11	2.0	8	15
20	ib	0.98	0.5	117.6	5.4	106.9	128.9	11	1.6	8	13
20	bf	0.98	0.7	98.6	3.3	94.4	105.5	10	1.6	8	13
20	ib	0.98	0.7	118.4	4.1	111.0	127.9	11	1.6	8	13
20	bf	0.98	0.9	98.7	3.9	90.8	104.5	11	1.5	9	15
20	ib	0.98	0.9	119.5	4.0	112.4	127.5	10	1.8	7	13
20	bf	0.98	0.9999	97.6	3.8	92.3	105.7	11	1.5	8	14
20	ib	0.98	0.9999	119.7	3.9	106.8	122.5	10	1.4	9	14
40	bf	0.85	0.05	92.1	2.4	88.9	98.2	12	1.7	10	15
40	ib	0.85	0.05	107.7	3.5	102.9	116.0	12	1.5	9	14
40	bf	0.85	0.5	92.5	3.1	88.6	100.0	11	1.7	8	14
40	ib	0.85	0.5	90.9	1.7	88.7	96.1	11	0.8	10	13
40	bf	0.85	0.7	92.1	2.6	90.6	99.8	11	1.7	8	15
40	ib	0.85	0.7	90.7	1.7	87.7	93.2	11	1.4	9	14
40	bf	0.85	0.9	92.9	4.1	90.1	104.4	12	1.8	9	15
40	ib	0.85	0.9	91.7	2.5	90.0	98.2	11	1.5	9	15
40	bf	0.85	0.9999	104.2	6.2	94.2	113.4	11	1.5	8	13
40	ib	0.85	0.9999	92.5	3.4	88.7	100.7	11	1.4	9	13
40	bf	0.90	0.05	93.6	2.3	89.6	97.2	12	1.5	10	15
40	ib	0.90	0.05	110.5	3.8	103.1	118.9	11	1.7	7	14
40	bf	0.90	0.5	92.5	2.8	89.0	100.0	11	1.5	8	14
40	ib	0.90	0.5	101.1	3.8	94.9	107.7	11	0.8	10	12
40	bf	0.90	0.7	93.3	2.7	87.9	99.8	11	1.5	9	15
40	ib	0.90	0.7	98.2	4.2	92.1	108.7	13	1.6	9	13
40	bf	0.90	0.9	94.4	3.4	91.2	103.5	11	1.5	10	15
40	ib	0.90	0.9	94.6	2.8	91.2	100.1	11	1.3	9	14
40	bf	0.90	0.9999	96.5	4.0	93.1	105.2	11	1.9	7	14
40	ib	0.90	0.9999	94.6	2.9	89.5	101.5	11	1.2	10	14
40	bf	0.95	0.05	98.3	4.1	90.8	105.5	11	1.5	7	13
40	ib	0.95	0.05	113.2	3.9	108.5	124.1	10	1.6	8	14
40	bf	0.95	0.5	96.2	3.2	91.4	103.9	11	1.6	8	14
40	ib	0.95	0.5	111.1	3.9	103.3	116.9	11	2.1	8	15
40	bf	0.95	0.7	96.6	5.1	90.7	106.7	11	1.3	9	13
40	ib	0.95	0.7	111.3	3.5	106.6	118.7	10	1.7	8	14
40	bf	0.95	0.9	95.7	3.8	89.6	103.0	10	1.4	9	14
40	ib	0.95	0.9	112.1	3.6	104.0	115.7	10	1.1	9	12
40	bf	0.95	0.9999	95.1	3.6	91.8	103.0	11	1.4	9	13
40	ib	0.95	0.9999	110.4	4.6	96.2	115.5	11	2.3	8	16
40	bf	0.98	0.05	118.5	2.7	113.3	124.6	10	1.9	7	13
40	ib	0.98	0.05	125.1	7.1	113.4	141.7	11	1.8	7	13
40	bf	0.98	0.5	121.7	6.3	110.5	137.0	10	1.7	8	13
40	ib	0.98	0.5	124.4	8.1	106.3	140.8	10	1.6	9	13
40	bf	0.98	0.7	117.6	4.7	107.3	126.7	10	2.0	8	15
40	ib	0.98	0.7	125.2	7.1	111.0	137.5	11	1.8	7	13
40	bf	0.98	0.9	117.0	4.8	107.0	126.8	10	2.3	7	16
40	ib	0.98	0.9	130.1	8.3	116.9	142.5	11	2.0	7	14
40	bf	0.98	0.9999	114.8	7.2	107.5	129.0	10	2.1	7	15
40	ib	0.98	0.9999	123.6	8.0	114.2	138.2	10	2.0	8	14
80	bf	0.85	0.05	94.8	3.6	89.4	102.4	11	1.1	9	13
80	ib	0.85	0.05	105.6	2.9	101.9	111.6	12	1.7	9	15
80	bf	0.85	0.5	93.2	2.4	89.7	97.1	11	1.7	9	15

(continues in next page . . .)

Table B.4: Results of MMAS for the Richmond network for constraint $N_p^{\text{sw}} \leq 3$ (continued from previous page).

MMAS parameters				C_E				N^{sw}			
A	$Sel.$	ρ	p_best	med.	sd.	best	worst	med.	sd.	best	worst
80	ib	0.85	0.5	94.4	4.3	91.2	103.4	11	1.5	9	14
80	bf	0.85	0.7	94.3	3.8	89.9	102.4	11	1.7	8	13
80	ib	0.85	0.7	93.2	3.2	89.0	100.6	11	1.2	9	13
80	bf	0.85	0.9	94.3	3.6	90.1	103.3	10	1.3	8	13
80	ib	0.85	0.9	93.1	3.3	90.3	99.8	11	1.6	8	14
80	bf	0.85	0.9999	97.7	4.4	92.3	105.6	11	1.5	8	13
80	ib	0.85	0.9999	93.1	3.0	90.2	100.1	11	1.3	9	14
80	bf	0.90	0.05	96.5	2.9	91.6	102.8	11	1.1	9	13
80	ib	0.90	0.05	111.6	3.0	107.0	117.5	12	1.8	7	14
80	bf	0.90	0.5	94.9	2.9	90.9	101.0	11	2.1	9	16
80	ib	0.90	0.5	105.3	3.2	100.9	113.8	12	2.0	9	15
80	bf	0.90	0.7	94.8	3.4	89.8	100.3	11	1.3	9	13
80	ib	0.90	0.7	107.4	3.9	99.7	111.7	12	1.6	8	14
80	bf	0.90	0.9	94.9	4.7	90.8	108.3	11	0.8	9	11
80	ib	0.90	0.9	104.8	2.4	100.3	108.9	10	1.7	8	14
80	bf	0.90	0.9999	94.8	4.1	92.2	103.0	11	1.6	9	14
80	ib	0.90	0.9999	103.6	3.1	98.8	108.7	11	1.5	9	14
80	bf	0.95	0.05	113.1	3.7	104.5	117.6	11	2.1	9	16
80	ib	0.95	0.05	121.3	5.3	111.3	133.9	10	1.8	8	14
80	bf	0.95	0.5	108.6	3.1	102.7	113.4	11	1.2	9	13
80	ib	0.95	0.5	120.2	5.9	110.7	130.7	9	1.9	6	13
80	bf	0.95	0.7	110.3	5.4	105.5	123.3	10	1.2	9	13
80	ib	0.95	0.7	122.6	4.5	112.2	128.0	10	1.9	7	13
80	bf	0.95	0.9	113.1	5.2	101.5	117.6	11	2.0	8	15
80	ib	0.95	0.9	118.7	4.4	113.6	129.0	10	1.6	8	13
80	bf	0.95	0.9999	111.0	4.1	101.9	117.3	10	1.5	8	14
80	ib	0.95	0.9999	118.9	5.4	109.2	130.1	10	2.2	6	13
80	bf	0.98	0.05	129.8	8.2	117.0	142.0	11	1.7	7	12
80	ib	0.98	0.05	128.2	6.3	118.6	140.7	10	1.8	7	13
80	bf	0.98	0.5	129.8	5.4	122.4	142.5	10	1.8	8	13
80	ib	0.98	0.5	132.7	9.4	117.6	150.6	10	1.8	7	13
80	bf	0.98	0.7	125.6	10.7	118.8	159.1	11	1.7	8	14
80	ib	0.98	0.7	132.6	6.6	119.6	143.1	10	2.1	7	14
80	bf	0.98	0.9	127.8	7.9	113.8	144.2	10	2.8	6	19
80	ib	0.98	0.9	131.9	8.2	116.7	143.0	10	2.1	7	14
80	bf	0.98	0.9999	130.6	8.3	112.1	143.7	9	2.0	6	14
80	ib	0.98	0.9999	126.6	9.7	107.2	142.8	10	1.8	7	14

Appendix C

Description of Solutions

Results of our experiments are publicly available at `http://sbe.napier.ac.uk/~manuel/`. All algorithms presented in this work use a common format for their output. We provide here a description of the format to help understand those results.

The output format of our optimisation algorithms is plain-text. Each solution is contained in a single line with fields separated by white space. Tab characters are used to separate main fields while simple spaces separate elements within a variable-length field. This format is both understandable for human readers and easier to read by computer programs. Each solution contains the objective values, additional information about the solution (such as constraint violations), the representation of the schedule of pumps used by the optimisation algorithm, and a canonical representation of the schedule.

The first elements of each line are the objective values. In a single objective algorithm, such as SEA (Chapter 4) or ACO (Chapter 6), this is a single value that indicates the total energy cost per day (C_{E} in Eq. 2.1). In the multi-objective variants proposed in Chapter 5, one additional field is given after the total energy cost. This additional field is one of: (*i*) total number of pump switches (N^{sw} in Eq. 2.2); (*ii*) shortest idle time (I^{T} in Eq. 5.1); or (*iii*) total volume deficit (ΔV in Eq. 2.4).

Independently of the problem formulation, additional fields contain information that may be helpful to assess the quality of a solution. The values of these fields represent in this order:

- Total volume deficit (ΔV in Eq. 2.4), which is the sum of volume deficits for all tanks. The volume deficit of a tank is the percentage of the initial volume minus the final volume with respect to the initial volume. Only percentages that are higher than a given volume deficit tolerance are accumulated to avoid surplus in one tanks compensating loss of volume in another. Unless explicitly stated otherwise, volume tolerance was zero in our experiments. The total volume deficit should be zero in any feasible solution, otherwise, there is a deficient balance between the supply and demand of water.

- Total number of pump switches (N^{sw} in Eq. 2.2). A *pump switch* is defined as turning on a pump that was not operating in the previous period Lansey & Awumah (1994). This value represents the number of switches performed by all pumps. High values are undesirable, since switching pumps on/off causes wear and tear of pumps that will increase future maintenance costs.

- Shortest idle time (I^{T} in Eq. 5.1) is the minimum idle time among all pumps. The idle time is the time interval in seconds between two operating periods of a pump. Very short time intervals between two operating periods may damage pumps and pipes due to sudden pressure fluctuations. Therefore, longer values are assumed to limit future maintenance costs.

- Pressure deficit (ΔH in Eq 2.6) is the total amount of pressure violations produced in all demand nodes during the scheduling period. This value should be zero in any feasible schedule.

- Number of warnings from the hydraulic simulator (see Section 2.4.1). Warnings indicate that EPANET encountered problems while evaluating the schedule. These warnings do not signal a problem in EPANET itself. A warning points out a potential issue with the schedule. For example, EPANET will issue a warning whenever a pump is forced to shut down or operate beyond its maximum rated flow. Schedules that generate warnings are considered infeasible, since it is very likely that the calculated objective values and constraints are incorrect.

Further variable-length fields indicate two different representations of the schedule of the pumps. The first group of fields is the schedule of pumps in the representation utilised within the optimisation algorithm. In this group, there are as many variable-length fields as pumps in the network instance. In the solutions generated in this work, this representation can be either:

- The binary representation (Section 3.1), where each element of the variable-length field is the pump status at each operating period. Therefore, in each field there are as many elements as the number of time intervals (N^{T}). An example of such field is:

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1
```

- Level-controlled triggers (Section 3.2), wherein each field contains four elements that represent two pairs of trigger levels, for off-peak and peak electrical tariff periods respectively. Each pair contains a lower trigger, whenever water in the associated tank is below this level the pump is activated, and an upper trigger, whenever water in the associated tank is above this level the pump is shut down. An example of this field is:

```
2.72 3.37 0.25 2.21
```

- Absolute time-controlled triggers (Section 3.3.1), wherein each field contains $2 \cdot SW$ elements, where SW is a representation parameter that defines the maximum number of switches per pump. Each element is the absolute time since the start of the scheduling period at which a pump changed its status. The time unit depends on the minimum time interval t_{min} used by the representation, typically one hour. Odd-numbered elements correspond to a switch from *off* to *on*, while even-numbered elements indicate a switch from *on* to *off*. Values higher than the scheduling period are ignored and allow to represent a pump schedule with less switches than SW. An example of this representation would be:

```
0 1 5 24 25 25
```

- Relative time-controlled triggers (Section 3.3.2), wherein each field contains $2 \cdot SW$ elements, as well. However, in this case, each element is the duration of an operating or idle interval. Odd-numbered elements correspond to an idle status of the pump while even-numbered elements indicate an active status. Zero values represent empty intervals and allow to represent a pump schedule with less switches than SW. An example of this representation is:

```
0 7 0 7 3 7
```

Finally, the last group of variable-length fields of each solution is a canonical representation of the schedule. This canonical representation is independent of the particular representation used by the optimisation algorithm. This representation contains as many fields as pumps. Each variable-length field contains as many elements as times the pump changed its status. These elements are non-negative integers that represent the time in seconds since the start of the scheduling period when a change on the status of a pump occurred. The following example shows the binary representation and its corresponding canonical representation of the schedule of a single pump:

```
0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1
10800 14400 28800 36000 39600 43200 61200 64800
```

A complete example is given below of the output corresponding to a single solution obtained by a single-objective optimiser using the binary representation. In this example, tabs are replaced by new lines for the sake of readability:

```
321.527
0.00
10
10800
0.0
0
1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1
18000 21600 25200 28800 32400 36000 39600 57600
7200 10800 21600 25200 32400 36000 39600 61200
0 32400 36000 39600
```

Appendix D

Changelog of EPANET Toolkit

During the development of our algorithms, it was necessary to add some features to the EPANET Toolkit and, in some cases, to fix some defects in the original code. Since one of our goals is the reproducibility of the results to allow future comparison, we made an effort to maintain backwards compatibility with the original code and to document any changes that may affect the computation of the results. In addition, the modified code is publicly available at `http://sbe.napier.ac.uk/~manuel/epanetlinux`. For reference, we provided here a list of all the modifications to the EPANET Toolkit.

- Added a file `Makefile` to simplify building on GNU/Linux. As a result, the library can be build by simply issuing the command `make` in the source directory.
- GNU/Linux specific code is conditional on a the macro `LINUX` being defined. It is automatically defined by `Makefile` when building the library in GNU/Linux.
- Added three example applications:

  `evalleveltriggers`: takes as input a network instance and a schedule of the pumps in level-controlled triggers representation. The schedule is evaluated and results are printed.

  `evalpattern`: takes as input a network instance and a schedule of the pumps described by utilisation patterns. The schedule is evaluated and results are printed.

  `getcoords`: takes as input a network instance. Prints the coordinates of the elements of the network.
- A vector (pump schedule vector) saves the time of day (in seconds) when a pump changes its status. No more than 24 events will be recorded in the vector of pump schedule and events occurred after 24 hours will not be recorded. Calculation of

pump switches and update of the pump schedule vector are performed by function `pumpswitch()`, which must be called from every point where the status of a pump is changed.

- Controls may be dynamically added by calling `ENsetcontrol()` with a value of parameter `cindex` such that $\texttt{cindex} = \texttt{count} + 1$, where `count` was obtained by calling `ENgetcount(EN_CONTROLCOUNT, &count).`
- When function `ENaddleveltrigger()` (`epanet.c`) is called with parameters $\texttt{start\_time} > \texttt{stop\_time}$, it adds one rule for the interval $[\texttt{start\_time}, \texttt{SECperDAY} - 1]$ and another for $[0, \texttt{stop\_time}]$. In case that `start_time` is equal to `stop_time`, do nothing.
- Function `addlevelbasedtrigger()` (`rules.c`) returns error 202 if called with parameters $\texttt{start\_time} \geq \texttt{stop\_time}$.
- New parameter codes were added to existing toolkit functions:
  - `EN_UPATTERN` of `ENsetlinkvalue()` allows to assign an utilisation pattern to a pump.
  - `EN_SCHEDULE` of `ENgetlinkvalue()` returns the pump schedule vector, which describes the time of the day (in seconds) when a pump status changed.
  - `EN_JUNCS`, `EN_PUMPCOUNT` and `EN_RESERVCOUNT` of `ENgetcount()` return the number of junctions, pumps and reservoirs in the network, respectively.
  - `EN_INITVOL` and `EN_VOLUME` of `ENgetnodevalue()` return initial and current volume of a tank.
  - `EN_CLOCKSTART` of `ENgettimeparam()` returns time of the day at which simulation begins (seconds).
  - `EN_MAXLEVEL` and `EN_MINLEVEL` of `ENgetnodevalue()` return maximum and minimum levels of a tank.
- New functions were added to the application program interface of the EPANET library:
  - `ENgettotalleakage()` gives total volume of water leaked by emitters.
  - `ENgetnode_xcoord()` gives x coordinate of a node.
  - `ENgetnode_ycoord()` gives y coordinate of a node.
  - `ENgettankindex()` and `ENgetpumpindex()` return the node index corresponding to each tank or pump without knowing its id.

- `ENgetpumpswitches()` returns the number of switches of a pump.
- `ENgettotalenergycost()` calculates total energy cost per pump plus demand cost.
- `ENaddpattern()` adds a new pattern to the database.
- `ENaddleveltrigger()` adds rule with following format:

  ```
  IF SYSTEM CLOCKTIME >= start_time (in seconds)
  AND SYSTEM CLOCKTIME <  stop_time  (in seconds)
  AND TANK id(tank_index) LEVEL [BELOW|ABOVE] level
  THEN PUMP id(pump_index) STATUS IS status
  ```
- `ENgetminstoptime()` returns the shortest time interval (in seconds) that a certain pump was not active.
- `ENrulesclear(void)` removes all rules.
- `ENgetnumwarnings(void)` returns number of warnings generated.

- Several modifications of the code were investigated in order to reduce the computation time without altering simulation results:
  - `pipecoeff()` marked as static inline.
  - Help arrays in `linsolve()` moved from `vars.h` to `smatrix.c`.
  - Enabled GCC option `-fmerge-constants` which attempts to merge identical constants (string constants and floating point constants) across compilation units.
  - Saved one division in `pipecoeff()`.
  - Removed unneccessary variables and operations in `linsolve()`. For example, `calloc()` should set memory to zero, and thus, `memset()` is redundant.
  - Replaced `ABS()` by `fabs()`, which is faster. However, not all `ABS()` calls could be replaced because of implicit casting between float and double resulting on rounding errors.
  - Added the option to compile for a particular architecture using a version of GNU Compiler Collection (GCC) higher than 3. For example, the command would be '`make march=pentium4`' for an Intel Pentium 4 architecture. The GCC documentation specifies the names of the architectures supported by each version of GCC.

The above modifications reduced the computation time by 13.7% on average when compiling with GCC 3.2.2 and running on an Intel Pentium 4 CPU 2.66GHz (cache size: 512 KB).

- Compiling with optimisation level 3 (`-O3`) reduces execution time by approximately 35%. However, this optimisation level affects floating-point calculations producing slightly different results. Therefore, optimisation is turned off by default. In order to turn it on, build the library using '`make all OPTIMISE=-O3`'.
- Section `[COORDINATES]` of input file is now parsed by `ENopen()`.
- The following constants are defined in `toolkit.h`, and hence, they may be used by application programs:

  `EN_MAX_ID_LEN`: Maximum number of characters in ID name.

  `EN_MAX_MSG_LEN`: Maximum number of characters in message text.

  `EN_MAX_FILENAME_LEN`: Maximum number of characters in file name.

  `EPANET_VERSION`: Version number corresponding to our variant of EPANET library.
- Fixed bugs in EPANET 2.00.10:
  - Missing parameter code `EN_SOURCEPAT` in `ENgetnodevalue()` caused error 251.
  - Avoid segmentation fault when `ENgetcontrol()` in `epanet.c` is called with `*nindex` > `Nnodes`.
  - Complex rules are checked since the start of the simulation. This may break compatibility with the original EPANET version 2.00.10, where the complex rules are not enforced until the first time step ($1/10$ of time interval) and, therefore, the results are slightly different if the initial conditions are not consistent with the rules)
  - Do not return error if the name of the report file and the name of the binary output file are both the empty string `""`.
  - Setting `lindex` to 0 using `ENsetcontrol()` should remove the control. However, `controltimestep()` does not check that `lindex` is equal to 0 before using other parameters, and thus, it does not ignore this control. The result is a wrong time step or a segmentation fault depending whether the parameters were valid or uninitialized values. This bug fix may break backward compatibility for programs which remove controls by setting `lindex` to 0 using `ENsetcontrol()` since a wrong time step may affect the results without the program reporting any error or crashing.

Appendix E

Thread-Safe Variant of EPANET Toolkit

From high-performance supercomputers accessible to researchers, to the new generation of multi-core personal computers and laptops, parallel computers are becoming increasingly prevalent nowadays. Parallel computation may reduce the time required to solve a problem. However, our tools need to be adapted in order to take advantage of it. In the context of the problem of finding an optimal schedule of pumps in a water distribution network, the tool may be a combination of an optimisation algorithm and a hydraulic simulator. The optimisation algorithm generates potential schedules of pumps, while the hydraulic simulator evaluates those schedules to calculate its cost and identify violations of system and performance constraints. Although a hydraulic simulator may require just a few seconds to perform an extended period simulation of a particular pump schedule, finding a near-optimal schedule typically requires the evaluation of thousands of different schedules. Some optimisation algorithms, such as Evolutionary Algorithms and Ant Colony Optimisation (ACO), are particularly well-suited for parallel execution, since, at each iteration, they generate a population of candidate solutions that can be independently evaluated. However, one of the most popular research simulators, the EPANET Toolkit (Rossman, 1999), was not designed with parallelism in mind.

In this appendix, we propose the implementation of a thread-safe variant of the EPANET Toolkit, which involves a more object-oriented design, allowing concurrent multiple simulations within the same application to be performed. This thread-safe variant allows a program to execute multiple simulations in concurrent *threads*, which are lightweight processes within the same computer program. The performance benefits of this thread-safe variant of EPANET Toolkit have been tested by combining it with an Ant Colony Optimisation (ACO) algorithm (see Section 6.5.2). This appendix explains the motivations behind the development of the thread-safe EPANET library. In addition, the thread-safe library is tested in a simple random search algorithm to assess its feasibility and its overhead compared to the original EPANET.

E.1 Limitations of EPANET Toolkit For Parallel Algorithms

The EPANET Toolkit (Rossman, 1999) is an open-source C library that provides an application programming interface (API) for hydraulic and water quality simulations. An optimisation algorithm would call certain functions of EPANET to load a network description, modify the schedule of the pumps, run an extended period simulation and collect information such as the energy consumption of the pumps, tank levels and pressure values. Figure E.1 shows the algorithmic schema that a hypothetical sequential optimisation algorithm would follow when interacting with EPANET. Among other functions, EPANET allows an application to load a network instance (`ENopen`), obtain information about the network (`ENgetcount`), assign schedules to pumps (`ENsetpattern`), and run extended period simulations (`ENopenH`, `ENinitH`, `ENrunH`, `ENnextH`, `ENcloseH`). The simple and straightforward interface is probably one of the reasons why EPANET is widely used for research.

However, some aspects of the design of EPANET make difficult its use in parallel applications. First, the complete status of a particular simulation cannot be easily retrieved and saved. In other words, we cannot simply make a copy of a running simulation, then start a new one and, once the new one is finished, restart the first one. In fact, the implementation of the library keeps most of its internal information on global variables that are dynamically allocated with no encapsulation at all. Moreover, data structures related to a particular simulation are often lumped together with data concerning the network description, which typically never changes during the simulation. This lack of encapsulation means that the status of a particular simulation cannot be isolated from another different simulation. The second issue that precludes the use of EPANET in parallel algorithms is that most API functions are not reentrant. A *reentrant* function only depends on its arguments and it does not hold any internal state. It neither calls non-reentrant functions. Therefore, it can be re-entered while it is running. This does not imply thread-safety by itself. If a reentrant function modifies its arguments and multiple threads call the function with the same arguments, there will be a problem of data synchronisation. However, a non-reentrant function called by multiple threads will not execute correctly even if the arguments are not shared among threads. In order to take advantage of parallel execution of multiple simulations, the functions called during simulation must be reentrant.

Two main refactoring efforts were undertaken to enable parallelism in EPANET. First, data structures related to a hydraulic simulation were encapsulated within a simulation object, thus multiple simulations can be created and modified concurrently. There are already types in EPANET for pumps, tanks and other elements. However, these objects contain both data that corresponds to the description of the network, such as pump shut-off head and tank maximum volume, and data that is dynamically calculated during simulation, such as pump energy usage and tank head. Thus an important step in our refactoring effort was the separation of these two kinds of data. Simulation data is encapsulated into a new type of object (`ENsimulation_t`). This was not the only new type that was created during refactoring. Other internal structures were also encapsulated within objects in order to make crucial EPANET functions reentrant.

The second refactoring task was the review of all EPANET functions, identifying those which are required in order to perform concurrent hydraulic simulations and converting them into reentrant functions. Since a reentrant function should only call other reentrant functions, the conversion proceeded from the API functions down to the internal functions used only within EPANET. In the original EPANET code, a function would depend on some internal state, reading and writing global variables, thus the same function cannot be executed by multiple threads. A reentrant variant works on a simulation object which is passed as an argument to the function. Hence, two threads can execute the same function concurrently as long as they use different simulation objects.

Figure E.2 shows an example of an optimisation algorithm using the new thread-safe variant of EPANET. For our purposes, it is not necessary that two threads are able to concurrently execute all EPANET functions on the same data. In fact, we do not even need all functions to be reentrant, since there is no need in a parallel optimisation algorithm to execute those functions in parallel. An example would be the function `ENopen()`, which is responsible for reading the network description from an input file. This function only needs to be called once. By restricting ourselves to our objective of enabling parallel hydraulic simulation, we obviate the incorporation to the EPANET library of synchronisation mechanisms, such as locking and mutual exclusion, that would be required otherwise. As well, this avoids new dependencies in order to build and use the EPANET library. We do not negate that future developments of EPANET may incorporate these characteristics. Nevertheless, the current approach is sufficient to implement state-of-the-art optimisation algorithms that make use of parallelism.

E.2 Parallel Random Search

We use a parallel random search algorithm to assess the potential of the new thread-safe version of EPANET. This random search algorithm simply generates a number of random pump schedules and evaluates them. After reaching a maximum of evaluations, it returns the schedule that generated the lowest electrical cost without violating any constraint.

The parallel random search evaluates schedules in parallel by using several threads. A candidate schedule of the pumps is assigned to each thread, which evaluates the schedule

```
Initialize ()
{
      ...
      ENopen (network_file);
      ENgetcount (EN_PUMPCOUNT, &num_pumps);
      ...
}

solution_t GenerateSolution ()
{
      ...
      schedule = GenerateSchedules ();
      ...
}

EvaluateSolution (solution_t schedule)
{
      ...
      ENopenH ();
      for (p = 0; p < num_pumps; p++)
            ENsetpattern (pump[p], schedule[p], 24);
      ENinitH (0);
      do {
            ENrunH ();
            ENnextH (&tstep);
      } while (tstep > 0);
      ENgettotalenergycost (&cost);
      EncloseH ();
      ...
      return cost;
}

main ()
{
      ...
      Initialize ();
      while (not stopping_criteria) {
            for (i = 0; i < population_size; i++) {
                  solution[i] = GenerateSolution ()
                  cost[i] = EvaluateSolution (solution[i])
            ...
      }
      ...
}
```

Figure E.1: Example of optimisation algorithm using EPANET.

```
Initialize ()
{
      ...
      ENopen (network_file);
      ENgetcount (EN_PUMPCOUNT, &num_pumps);
      ...
}

solution_t GenerateSolution ()
{
      ...
      schedule = GenerateSchedules ();
      ...
}

double EvaluateSolution (solution_t schedule)
{
      ...
      // A new simulation object is created every time
      // this function is executed.
      ENsimulation_t simulation;
      ENopenH (&simulation);
      for (p = 0; p < num_pumps; p++)
            ENsetpattern (simulation->pump[p], schedule[p], 24);
      ENinitH (simulation, 0);
      do {
            ENrunH (simulation);
            ENnextH (simulation, &tstep);
      } while (tstep > 0);
      ENgettotalenergycost (simulation, &cost);
      EncloseH (simulation);
      ...
      return cost;
}

main ()
{
      ...
      Initialize ();
      while (not stopping_criteria) {
            for (i = 0; i < population_size; i++) {
            // This ``for'' can be executed concurrently
            // using i parallel threads.
                  solution[i] = GenerateSolution ();
                  cost[i] = EvaluateSolution (solution[i])
            ...
      }
      ...
}
```

Figure E.2: Example of algorithm using the thread-safe version of EPANET.

by performing a hydraulic simulation. As soon as one thread finishes the evaluation of a solution, a new random solution is generated and assigned to it. Therefore, a thread does not wait for other threads to finish.

We apply the random search algorithm to the Richmond network instance described in Section 2.5.2. The efficacy of the algorithm in terms of solution quality is of no interest here. Our only goal is to study how much the execution time is reduced by increasing the number of threads in a multi-core computer. We are also interested in the algorithm speedup, which is defined as:

$$S_n = \frac{T_1}{T_n} \tag{E.1}$$

where S_n is the speedup, n is the number of processors, T_1 is the time required by the sequential algorithm and T_n is the time of the parallel algorithm with p processors. The concept of speedup indicates how well a parallel algorithm performs in comparison to the sequential code when using an increasing number of processors. Ideal speedup occurs when $S_n = n$. Conversely, an ideal parallel execution time can be calculated, which corresponds to the execution time of the parallel algorithm that result in ideal speedup.

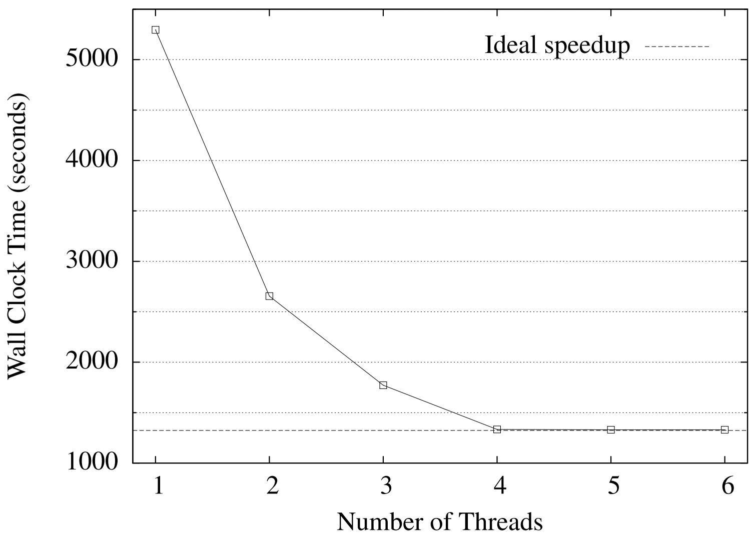


Figure E.3: Runtime in seconds for Random Search algorithm.

Figure E.3 shows the wall-clock time required for 8000 evaluations of the random search algorithm in a 4-CPU machine (2 dual-core AMD64 Opteron 275, 2.2 GHz and 64KB/1MB of cache memory per core) running GNU/Linux. The algorithm is implemented in C and uses POSIX threads (Kerrisk, 2005). With one thread the algorithm is sequential and only makes use of one CPU. In this case, Fig. E.3 shows that the runtime for 8000 evaluations is close to one hour. By using two threads, the load is shared between two CPUs and therefore, the computation time is halved. For a number of threads equal or higher than four, the speedup obtained is practically ideal, that is, close to four. Therefore, there is little to no overhead in the parallel implementation of the EPANET library with respect to the sequential version.

E.3 Summary

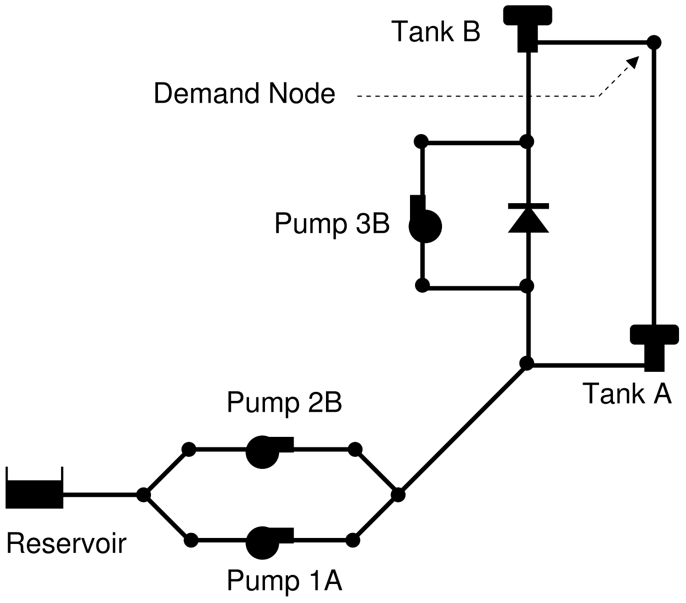
In this appendix we have discussed the limitations of the original EPANET Toolkit for parallel computing. Thereby, we present a thread-safe variant of the EPANET Toolkit that can be used by parallel applications. By introducing encapsulation to the data structures of the original EPANET and modifying functions to be reentrant, the proposed thread-safe library can simulate multiple schedules in parallel on the same network instance. This enables optimisation algorithms to take advantage of multiple processors to evaluate multiple candidate solutions concurrently, hence reducing the total runtime of the algorithm. Source code of the thread-safe version of EPANET proposed here is available at `http://sbe.napier.ac.uk/~manuel/epanet-thread-safe`.

Experiments were performed by linking the thread-safe library with a trivial random search algorithm. Experimental results showed that the speedup obtained by the algorithm was practically ideal. Admittedly, the random search algorithm discussed here is of little practical interest. Nonetheless, it suffices to show that there is no noticeable overhead of the thread-safe variant with respect to the original EPANET. As shown in Section 6.5.2, the computation time of state-of-the-art optimisers can be greatly reduced by using the thread-safe library proposed here.

Appendix F

Vanzyl Network EPANET Input File

This test network was proposed by van Zyl, Savic & Walters (2004). We reproduce here a more complete description of the network, which can be directly used by EPANET without further modifications.



```
[JUNCTIONS]
;ID     Elev    Demand  Pattern
 n1     10      0       ;
 n10    100     0       ;
 n12    100     0       ;
 n11    100     0       ;
 n13    100     0       ;
 n2     10      0       ;
 n3     75      0       ;
 n361   100     0       ;
 n362   100     0       ;
 n364   100     0       ;
 n365   100     0       ;
 n5     30      50      pattern24       ;
 n6     30      100     pattern24       ;

[RESERVOIRS]
;ID     Head    Pattern
 r1     20      ;

[TANKS]
;ID  Elevation InitLevel MinLevel MaxLevel Diameter MinVol VolCurve
 t6  85         9.5       0        10       20       0       ;
 t5  80         4.5       0        5        25       0       ;

[PIPES]
;ID     Node1   Node2   Length  Diameter Roughness MinorLoss Status
 p1     r1      n1      1       1000    100     0       Open    ;
 p10    n1      n10     1       1000    100     0       Open    ;
 p12    n1      n12     1       1000    100     0       Open    ;
 p11    n11     n2      1       1000    100     0       Open    ;
 p13    n13     n2      1       1000    100     0       Open    ;
 p2     n2      n3      2600    450     100     0       Open    ;
 p18    n3      n361    1       1000    100     0       Open    ;
 p361   n361    n362    1       1000    100     0       Open    ;
 p364   n364    n365    1       1000    100     0       Open    ;
 p4     n365    t6      2000    350     100     0       Open    ;
 p6     t6      n6      1100    300     100     0       Open    ;
 p5     t5      n5      500     300     100     0       Open    ;
 p3     n3      t5      1000    350     100     0       Open    ;
 p7     n6      n5      1       200     100     0       Open    ;
 p19    n361    n365    1       1000    100     0       CV      ;

[PUMPS]
;ID     Node1   Node2   Parameters
 pmp1   n10     n11     HEAD 1  ;
 pmp2   n12     n13     HEAD 1  ;
 pmp3   n362    n364    HEAD 6  ;


[PATTERNS]
;ID     Multipliers
;
 pattern24      0.62    0.62    0.67    0.76    0.91    1.1
 pattern24      1.48    1.71    1.48    1.02    0.73    0.55
 pattern24       0.49    0.55    0.73    1.02    1.36    1.53
 pattern24       1.53    1.36    1.1     0.91    0.76    0.67
;
 pumptariff      0.0244  0.0244  0.0244  0.0244  0.0244  0.0244
 pumptariff      0.0244  0.1194  0.1194  0.1194  0.1194  0.1194
 pumptariff      0.1194  0.1194  0.1194  0.1194  0.1194  0.1194
 pumptariff      0.1194  0.1194  0.1194  0.1194  0.1194  0.1194


[CURVES]
;ID     X-Value Y-Value
;PUMP:
 1      0       100
 1      120     90
 1      150     83
;PUMP:
 6      0       120
 6      90      75
 6      150     0
;EFFICIENCY:
 leff   50      78
 leff   107     80
 leff   151     68
 leff   200     60

[ENERGY]
 Global Efficiency      85
 Global Price   0
 Demand Charge  0
 Pump   pmp1    Efficiency      leff
 Pump   pmp1    Price   1
 Pump   pmp1    Pattern pumptariff
 Pump   pmp2    Efficiency      leff
 Pump   pmp2    Price   1
 Pump   pmp2    Pattern pumptariff
 Pump   pmp3    Price   1
 Pump   pmp3    Pattern pumptariff

[REACTIONS]
 Order Bulk     1
 Order Tank     1
 Order Wall     1
 Global Bulk    0
 Global Wall    0
 Limiting Potential     0
 Roughness Correlation  0

[TIMES]
 Duration               24
 Hydraulic Timestep     1:00
 Quality Timestep       0:05
 Pattern Timestep       1:00
 Pattern Start          7:00
 Report Timestep        1:00
 Report Start           0:00
 Start ClockTime        7 am
 Statistic              None

[REPORT]
 Status   Full
 Summary  No
 Page     0

[OPTIONS]
 Units  LPS
 Headloss       H-W
 Specific Gravity       1
 Viscosity      1
 Trials 40
 Accuracy       0.00001
 Unbalanced     Continue 10
 Pattern1
 Demand Multiplier      1.0
 Emitter Exponent       0.5
 QualityNone mg/L
 Diffusivity    1
 Tolerance      0.01

[COORDINATES]
;Node   X-Coord Y-Coord
 n1     2100.00 3900.00
 n10    2300.00 4100.00
 n12    2300.00 3700.00
 n11    2700.00 4100.00
 n13    2700.00 3700.00
 n2     2900.00 3900.00
 n3     3800.00 4800.00
 n361   3800.00 5300.00
 n362   3500.00 5300.00
 n364   3500.00 5800.00
 n365   3800.00 5800.00
 n5     4500.00 6000.00
 n6     4500.00 6500.00
 r1     1800.00 3900.00
 t6     3800.00 6500.00
 t5     4500.00 4800.00

[LABELS]
;X-Coord  Y-Coord Label & Anchor Node
 2315.11 3553.23 "Pump1A"
 2344.09 4408.22 "Pump2B"
 3074.43 5558.25 "Pump3B"
 4270.52 4707.55 "TankA" t5
 3451.77 6482.73 "TankB" t6

[BACKDROP]
 DIMENSIONS     46.32   2697.51 5335.63 7152.33
 UNITS  None
 FILE
 OFFSET 0.00    0.00

[END]
```

Bibliography

Andersen, J.H. & Powell, R.S., 1999. The use of continuous decision variables in an optimising fixed speed pump scheduling algorithm. **In**: Powell, R.S. & Hindi, K.S., eds., *Computing and Control for the Water Industry*, pp. 119–128. Research Studies Press Ltd.

Atkinson, R., van Zyl, J.E., Walters, G.A. & Savic, D.A., 2000. Genetic algorithm optimisation of level-controlled pumping station operation. **In**: *Water network modelling for optimal design and management*, pp. 79–90. Centre for Water Systems, Exeter, U.K.

Beume, N. & Rudolph, G., 2006. Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. **In**: Kovalerchuk, B., ed., *Proceedings of the Second IASTED Conference on Computational Intelligence*, pp. 231–236. ACTA Press, Anaheim.

Bleuler, S., Laumanns, M., Thiele, L. & Zitzler, E., 2003. PISA – a platform and programming language independent interface for search algorithms. **In**: Fonseca, C.M., Fleming, P., Zitzler, E., Deb, K. & Thiele, L., eds., *Evolutionary Multi-Criterion Optimization (EMO 2004)*, *Lecture Notes in Computer Science*, vol. 2632, pp. 494–508. Springer.

Boulos, P.F., Orr, C.H., de Schaetzen, W., Chatila, J.G., Moore, M., Hsiung, P. & Thomas, D., 2001. Optimal pump operation of water distribution systems using genetic algorithms. **In**: *AWWA Distribution System Symp.* Denver, USA: American Water Works Association.

Cohen, G., 1982. Optimal control of water supply networks. **In**: Tzafestas, S.G., ed., *Optimization and Control of Dynamic Operational Research Models*, vol. 4, chap. 8, pp. 251–276. Amsterdam: North-Holland Publishing Company.

Dandy, G.C. & Gibbs, M.S., 2003. Optimizing system operations and water quality. **In**: Bizier, P. & DeBarry, P., eds., *Proceedings of World Water and Environmental Resources Congress*. Philadelphia, USA: ASCE. On CD-ROM.

Dean, A. & Voss, D., 1999. *Design and Analysis of Experiments*. Springer, London, UK.

Deb, K., 2000. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, **186** (2/4) pp. 311–338.

Deb, K., 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK: Wiley.

Dorigo, M., 1992. *Optimization, Learning and Natural Algorithms*. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy. In Italian.

Dorigo, M. & Gambardella, L., 1997. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, **1** (1) pp. 53–66.

Dorigo, M., Maniezzo, V. & Colorni, A., 1996. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, **26** (1) pp. 29–41.

Dorigo, M. & Stützle, T., 2004. *Ant Colony Optimization*. Cambridge, MA: MIT Press.

Ehrgott, M., 2000. *Multicriteria optimization*, *Lecture Notes in Economics and Mathematical Systems*, vol. 491. Springer-Verlag, Berlin.

Ertin, E., Dean, A.N., Moore, M.L. & Priddy, K.L., 2001. Dynamic optimization for optimal control of water distribution systems. **In**: Priddy, K.L., Keller, P.E. & Angeline, P.J., eds., *Applications and Science of Computational Intelligence IV, Proceedings of SPIE*, vol. 4390, pp. 142–149.

Eshelman, L.J. & Schaffer, J.D., 1992. Real-coded genetic algorithms and interval-schemata. **In**: Whitley, L.D., ed., *FOGA*, pp. 187–202. Morgan Kaufmann. ISBN 1-55860-263-1.

Fogel, D.B., 1995. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. IEEE Press.

Fonseca, C.M., Grunert da Fonseca, V. & Paquete, L., 2005. Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. **In**: Coello, C.C., Aguirre, A.H. & Zitzler, E., eds., *Evolutionary Multi-criterion Optimization (EMO 2005)*, *Lecture Notes in Computer Science*, vol. 3410, pp. 250–264. Springer.

Fonseca, C.M., Paquete, L. & López-Ibáñez, M., 2006. An improved dimension-sweep algorithm for the hypervolume indicator. **In**: *IEEE Congress on Evolutionary Computation*, pp. 1157–1163. IEEE Press.

Furlong, N.E., Lovelace, E.A. & Lovelace, K.L., 2000. *Research Methods and Statistics: An Integrated Approach*. Harcourt College Publishers.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goldman, F.E. & Mays, L.W., 2000. The application of simulated annealing to the optimal operation of water systems. **In**: *Proceedings of 26th Annual Water Resources Planning and Management Conference*. Tempe, USA: ASCE.

Grunert da Fonseca, V., Fonseca, C.M. & Hall, A.O., 2001. Inferential performance assessment of stochastic optimisers and the attainment function. **In**: Zitzler, E., Deb, K., Thiele, L., Coello, C.A. & Corne, D., eds., *Evolutionary Multi-criterion Optimization (EMO 2001)*, *Lecture Notes in Computer Science*, vol. 1993, pp. 213–225. Springer.

Herrera, F., Lozano, M. & Sánchez, A.M., 2003. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, **18** (3) pp. 309–338.

Kazantzis, M.D., Simpson, A.R., Kwong, D. & Tan, S.M., 2002. A new methodology for optimizing the daily operations of a pumping plant. **In**: *Proceedings of 2002 Conference on Water Resources Planning*. Roanoke, USA: ASCE.

Kerrisk, M., 2005. pthreads - POSIX threads. **In**: *Linux Programmer's Manual*, Section 7. `http://www.linux-man-pages.org/man7/pthreads/`. (Last accessed May 15 2008).

Knowles, J.D., Thiele, L. & Zitzler, E., 2006. A tutorial on the performance assessment of stochastive multiobjective optimizers. TIK-Report 214, Computer Engineering and Networks Laboratory, ETH Zürich. Revised version.

Lansey, K.E. & Awumah, K., 1994. Optimal pump operations considering pump switches. *Journal of Water Resources Planning and Management, ASCE*, **120** (1) pp. 17–35.

López-Ibáñez, M., 2007. High performance ant colony optimisation of the pump scheduling problem. **In**: Alberigo, P., Erbacci, G., Garofalo, F. & Monfardini, S., eds., *Science and Sumpercomputing in Europe*, pp. 371–375. CINECA. ISBN 978-88-86037-21-1.

López-Ibáñez, M., Paquete, L. & Stützle, T., 2006. Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, **5** (1) pp. 111–137.

López-Ibáñez, M., Paquete, L. & Stützle, T., 2009a. Exploratory analysis of stochastic local search algorithms for biobjective problems through empirical attainment functions. **In**: Bartz-Beielstein, T., Chiarandini, M., Paquete, L. & Preuss, M., eds., *Empirical Methods for the Analysis of Optimization Algorithms*. Natural Computing Series, Springer Verlag, Berlin, Germany. To appear.

López-Ibáñez, M., Paquete, L. & Stützle, T., 2009b. Exploratory analysis of stochastic local search algorithms in biobjective optimization. Tech. Rep. TR/IRIDIA/2009-015, IRIDIA, Université Libre de Bruxelles, Belgium. This has been published as a book chapter (López-Ibáñez, Paquete & Stützle, 2009a).

López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2005a. Multi-objective optimisation of the pump scheduling problem using SPEA2. **In**: *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 435–442. Edinburgh, UK.

López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2005b. Optimal pump scheduling: Representation and multiple objectives. **In**: Savic, D.A., Walters, G.A., King, R. & Thiam-Khu, S., eds., *Proceedings of the Eighth International Conference on Computing and*

Control for the Water Industry (CCWI2005), vol. 1, pp. 117–122. University of Exeter, UK.

López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2008a. Ant colony optimisation for the optimal control of pumps in water distribution networks. *Journal of Water Resources Planning and Management, ASCE*, **134** (4) pp. 337–346.

López-Ibáñez, M., Prasad, T.D. & Paechter, B., 2008b. Parallel optimisation of pump schedules with a thread-safe variant of EPANET toolkit. **In**: van Zyl, J.E., Ilemobade, A.A. & Jacobs, H.E., eds., *Proceedings of the 10th Annual Water Distribution Systems Analysis Conference WDSA2008*. ASCE.

Mäckle, G., Savic, D.A. & Walters, G.A., 1995. Application of genetic algorithms to pump scheduling for water supply. **In**: *Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA'95*, vol. 414, pp. 400–405. Sheffield, UK: IEE Conference Publication.

Maier, H.R., Simpson, A.R., Zecchin, A.C., Foong, W.K., Phang, K.Y., Seah, H.Y. & Tan, C.L., 2003. Ant colony optimization for design of water distribution systems. *Journal of Water Resources Planning and Management, ASCE*, **129** (3) pp. 200–209.

McCormick, G. & Powell, R.S., 2003a. Optimal pump scheduling in water supply systems with maximum demand charges. *Journal of Water Resources Planning and Management, ASCE*, **129** (5) pp. 372–379.

McCormick, G. & Powell, R.S., 2003b. A progressive mixed integer-programming method for pump scheduling. **In**: Maksimović, C., Butler, D. & Memon, F.A., eds., *Advances in Water Supply Management*, pp. 307–313. London, UK.

McCormick, G. & Powell, R.S., 2004. Derivation of near-optimal pump schedules for water distribution by simulated annealing. *Journal of the Operational Research Society*, **55** (7) pp. 728–736.

Michalewicz, Z., 1996. *Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edition*. Berlin, Germany: Springer-Verlag.

Mühlenbein, H. & Schlierkamp-Voosen, D., 1993. Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, **1** (1) pp. 25–49.

Nitivattananon, V., Sadowski, E.C. & Quimpo, R.G., 1996. Optimization of water supply system operation. *Journal of Water Resources Planning and Management, ASCE*, **122** (5) pp. 374–384.

Ormsbee, L.E. & Lansey, K.E., 1994. Optimal control of water supply pumping systems. *Journal of Water Resources Planning and Management, ASCE*, **120** (2) pp. 237–252.

Ormsbee, L.E. & Reddy, S.L., 1995. Nonlinear heuristic for pump operations. *Journal of Water Resources Planning and Management, ASCE*, **121** (4) pp. 302–309.

Paquete, L., Stützle, T. & López-Ibáñez, M., 2005. Towards the empirical analysis of SLS algorithms for multiobjective combinatorial optimization problems through experimental design. **In**: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F. & Reimann, M., eds., *6th Metaheuristics International Conference (MIC 2005)*, pp. 739–746. Vienna, Austria.

Paquete, L., Stützle, T. & López-Ibáñez, M., 2007. Using experimental design to analyze stochastic local search algorithms for multiobjective problems. **In**: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F. & Reimann, M., eds., *Metaheuristics: Progress in Complex Systems Optimization*, *Operations Research / Computer Science Interfaces*, vol. 39, pp. 325–344. Springer-Verlag, New York.

Pezeshk, S. & Helweg, O.J., 1996. Adaptative search optimisation in reducing pump operation costs. *Journal of Water Resources Planning and Management, ASCE*, **122** (1) pp. 57–63.

Prasad, T.D. & Walters, G.A., 2003. Optimal rerouting to minimise residence times in water distribution networks. **In**: Maksimović, C., Butler, D. & Memon, F.A., eds., *Advances in Water Supply Management*, pp. 299–306. London, UK.

Rossman, L.A., 1994. *EPANET User's Guide*. Risk Reduction Engineering Laboratory, Office of Research and Development, U.S. Environmental Protection Agency, Cincinnati, USA.

Rossman, L.A., 1999. The EPANET Programmer's Toolkit for analysis of water distribution systems. **In**: *Proceedings of the Annual Water Resources Planning and Management Conference*. Reston, USA: ASCE.

Rossman, L.A., 2000. *EPANET 2 Users Manual*. U.S. Environmental Protection Agency, Cincinnati, USA.

Sakarya, A.B.A. & Mays, L.W., 2000. Optimal operation of water distribution pumps considering water quality. *Journal of Water Resources Planning and Management, ASCE*, **126** (4) pp. 210–220.

Savic, D.A., Walters, G.A. & Schwab, M., 1997. Multiobjective genetic algorithms for pump scheduling in water supply. **In**: Corne, D. & Shapiro, J.L., eds., *Evolutionary Computing Workshop, AISB'97*, *Lecture Notes in Computer Science*, vol. 1305, pp. 227–236. Springer-Verlag Berlin.

Sheskin, D.J., 2000. *Hanbook of Parametric and Nonparametric Statistical Procedures*. Second ed. Chapman & Hall/CRC.

Simpson, A.R., Sutton, D.C., Keane, D.S. & Sherriff, S.J., 1999. Optimal control of pumping at a water filtration plant using genetic algorithms. **In**: Savic, D.A. & Walters, G.A., eds., *Water Industry Systems: Modelling and Optimization Applications*, vol. 2. Baldock, United Kingdom: Research Studies Press Ltd.

Sotelo, A., von Lücken, C. & Barán, B., 2002. Multiobjective evolutionary algorithms in pump scheduling optimisation. **In**: Topping, B.H.V. & Bittnar, Z., eds., *Proceedings of the Third International Conference on Engineering Computational Technology*. Civil-Comp Press, Stirling, Scotland.

Stützle, T. & Hoos, H.H., 2000. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, **16** (8) pp. 889–914.

van Zyl, J.E., 2001. *A Methodology for Improved Operational Optimization of Water Distribution Systems*. Ph.D. thesis, School of Engineering and Computer Science, University of Exeter, UK.

van Zyl, J.E., Savic, D.A. & Walters, G.A., 2004. Operational optimization of water distribution systems using a hybrid genetic algorithm. *Journal of Water Resources Planning and Management, ASCE*, **130** (2) pp. 160–170.

Walski, T.M., Chase, D.V., Savic, D.A., Grayman, W., Beckwith, S. & Koelle, E., 2003. *Advanced Water Distribution Modeling and Management*. First ed. Haestad Methods, Inc., Haestad Press.

Wegley, C., Eusuff, M. & Lansey, K.E., 2000. Determining pump operations using particle swarm optimization. **In**: Hotchkiss, R.H. & Glade, M., eds., *Building Partnerships, Proceedings of the Joint Conference on Water Resources Engineering and Water Resources Planning and Management*. Minneapolis, USA.

Zhang, Q. & Li, H., 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, **11** (6) pp. 712–731.

Zhang, Q. & Suganthan, P.N., 2009. Special session on performance assessment of multiobjective optimization algorithms/CEC'09 MOEA competition. `http://dces.essex.ac.uk/staff/qzhang/moeacompetition09.htm`.

Zitzler, E., Laumanns, M. & Thiele, L., 2002. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. **In**: Giannakoglou, K., Tsahalis, D., Periaux, J., Papaliliou, K. & Fogarty, T., eds., *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems. Proceedings of the EUROGEN2001 Conference*, pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE).

Zitzler, E. & Thiele, L., 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, **3** (4) pp. 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M. & Grunert da Fonseca, V., 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, **7** (2) pp. 117–132.