PRODUCTION PLANNING FOR WATER SUPPLY NETWORKS

By

Michael Hofmeister, Sean Ahmad Colbert-Kelly

Kirill Levin, Huijuan Li

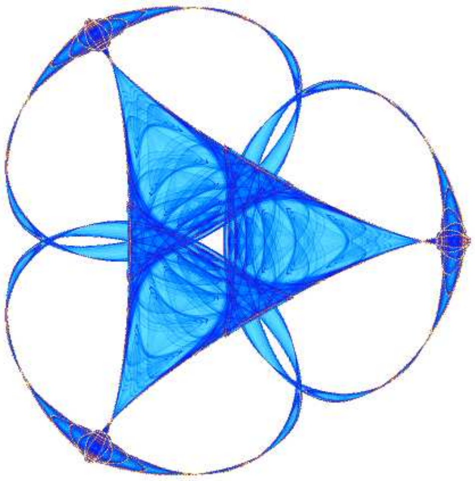
Ignacio Rozada, Rogelio Salinas Gutiérrez

and

Benjamin Sánchez Lengeling

IMA Preprint Series # 2334

(September 2010)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA

400 Lind Hall

207 Church Street S.E.

Minneapolis, Minnesota 55455–0436

Phone: 612-624-6066 Fax: 612-626-7370

URL: http://www.ima.umn.edu

Mathematical Modeling in Industry XIV
CIMAT – IMA – PIMS

Production planning for water supply networks

Mentor

Michael Hofmeister	Siemens Corporate Technology, DE

Team 3

Sean Ahmad Colbert-Kelly	Purdue University, US
Kirill Levin	University of Toronto, CA
Huijuan Li	Rutgers University, US
Ignacio Rozada	University of British Columbia, CA
Rogelio Salinas Gutiérrez	Center for Research in Mathematics, MX
Benjamin Sánchez Lengeling	Universidad de Guanajuato, MX

August 11, 2010

Abstract

In this project, we solved a minimum cost flow problem derived from a simulated water supply network. The objective function was a combination of minimizing both the cost of moving water through the network, and the number of pump switches. The original water flow model was projected into a mathematical graph, and the associated minimum cost flow problem for the simplified continuous case was solved as an LP problem. We then considered discrete pumps and added the minimized discrete pumps via both a heuristic approach and by expanding the original network. The problem was solved to within 3% of the optimal solution, while simultaneously minimizing pump switches.

Contents

Chapter 1

Introduction

1.1 Water Supply Network Planning

Consider the real-life scenario of a water supply network, which routes water from various sources to consumers in a town or city within a particular region. To ensure that the demands of the consumers are exactly met, planning must take place.

The planning objectives are to ensure that the network is operating safely, the consumers receive the correct amount of water, the operation costs are minimized, and that as small a number of pump switches (turning a pump on or off) as possible are performed. The last objective is intended to capture the cost of maintenance and replacement of the pumps, as switches cause the majority of wear and tear damage. Figure 1.1 describes the process for creating and executing a network schedule.

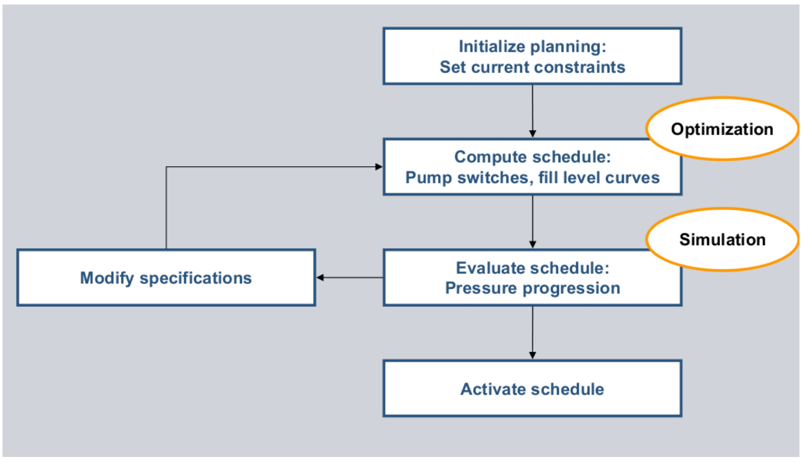

Initialize planning: Set current constraints
Optimization
Compute schedule: Pump switches, fill level curves
Simulation
Modify specifications
Evaluate schedule: Pressure progression
Activate schedule

Figure 1.1: Methodology for developing and implementing a water network schedule.

Inputs for the network include contracts with other water suppliers, demand forecasts for the consumers, energy costs for pumps, and capacities of the various other water sources. As water flows through the network, certain requirements must be maintained, such as filling levels of reservoirs within the network. A plan usually covers a

24 hour period, and consists of a schedule for turning pumps on and off, settings for valves, and amounts of water to retrieve from the water sources.

1.2 The Water Supply Network at the Workshop

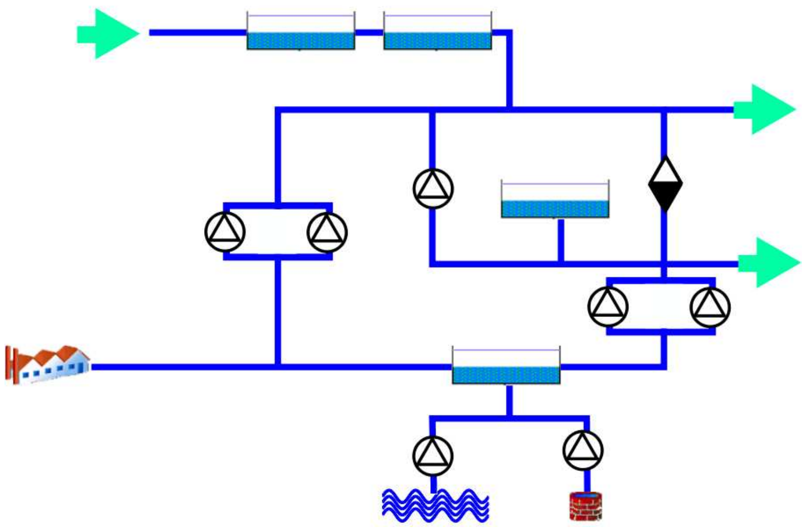


Figure 1.2: Diagram of the team's specific water supply network

The team was given a specific water supply network to schedule and plan (shown in Figure 1.2). In this network, there are four water sources: a well, a spring, a feeder, and an external water supplier. Within the network, there are four reservoirs that store water, six pumps that move water through the network, and one valve for controlling water flow. The network leads into two separate areas of consumers with specified demands.

The spring is connected to a pump that must run constantly, hence forcing a minimal, constant supply of water into the network. The well, feeder, and supplier all have a maximum flow capacity per hour. The customers have hourly demands that are forecasted and provided as input. Both the pumps and the valves have a certain maximal capacity of water that can go through them. Pumps are discrete, in the sense that they are either on at full capacity, or turned off, and they may be switched every minute; they also have an energy requirement. The valve, on the other hand, is continuously controllable, and is free to operate.

The reservoirs begin the cycle with specified amounts of water in them, and must end up with specific amounts for the next planning cycle. Each reservoir has a minimum and maximum filling level, which correspond to "safe" levels of water, and are meant as a buffer. The reservoirs' cross-sectional area, maximum height, and geodesic height are all given.

The energy and water supplier have prices that vary per hour.

1.3 Goal of the Project

Our main objective consisted of developing a tool that outputs a valid plan given the aforementioned inputs. The plan must minimize operational costs while simultaneously reducing the number of pump switches.

Pressure progression in the system was not considered.

Chapter 2

The Network Graph

The water supply network depicted in Figure 1.2 can be described mathematically as a graph, i.e., a collection of nodes and edges. The key to the transformation is the fact that there is only a limited number of places where the network operator can observe water flow. These spots consist of the pumps, the valve, and the pipes that go to the source and sink. On the other hand, the flow on the reservoirs cannot be observed or controlled directly. The graphical representation is then a network where the nodes are the areas we cannot observe, and the edges correspond to the places where we can control or observe water flow.

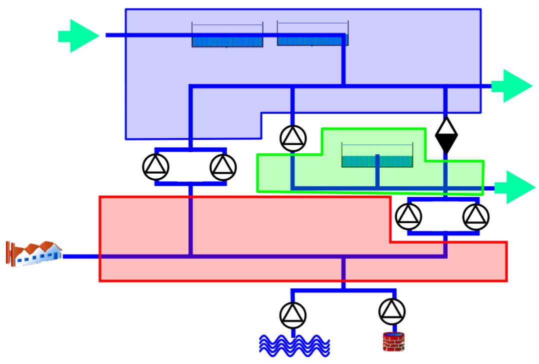


Figure 2.1: The original network subdivided into 3 subnetworks, which will form the basis for the graph representation.

The full network can then be broken into three subgraphs, as shown in Figure 2.1, plus a node representing the sources, and a node representing the sinks. There are 12 edges, one for each of the six pumps, the valve, the external supplier, the spring, the feeder, and the two consumers.

The red, green, and blue edges will become important once we describe the dynamics of the network over 24 hours. Suffice it to say that they represent the water levels in the reservoirs as they change over time.

Each node has a number associated to it, which represents its capacity, i.e. the amount of water that is either produced or consumed by the node. The source node has a positive capacity, whereas the capacity of the sink node is negative. The green, red, and blue nodes, which for the most part contain the water in the reservoirs, have zero capacity, as the water flowing in at each time step will be the same as the water flowing out of them. In the minimization process this will provide us with an equivalent of Kirchhoff's laws: the flow through each node has to be equal to the capacity, and the total flow of water in the network has to be zero.

Likewise, each edge in the network has an upper and lower capacity, representing the minimum and maximum amount of water that can flow through it. Most of the lower bounds are zero. In the case of the pumps and the valve, the upper bounds represent their maximum capacity. There is also a cost per unit-volume associated with each edge. This represents the cost of moving water through each edge and, in the case of the pumps, corresponds to the energy cost multiplied by the energy consumption of each pump. The valve has zero cost, while the water supplier has cost given by the price of a unit of water per hour. The spring and consumer edges have fixed requirements (i.e. lower bound equal to upper bound) which correspond to their supply and demand, respectively.

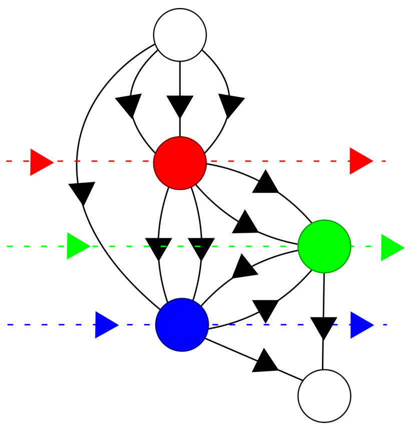


Figure 2.2: A one hour snapshot of the graph.

As previously mentioned, the problem consists of scheduling the supply of water to the network during a 24 hour interval, with the specified demands of the consumers varying hourly. The standard method for solving minimum cost flow problems is strictly for stationary networks, as opposed to our problem where the network changes each hour. A way to overcome this limitation is to expand the network given in Figure 2.2, and turn it into a stationary problem. We need a copy of the network for each hour, and they will all be connected through the reservoirs, the sources and the sinks. The expanded network will have new edges linking the reservoirs across time, essentially stating that the level in each reservoir at the end of each hour becomes the initial level at the start of the next hour. These are the only true edges that have non-zero lower bounds, corresponding to the minimum amount of water that must be stored in the reservoirs. Similarly, the upper bound corresponds to the maximum amount of water that may be stored in them. A two hour instance is given below, in Figure 2.3.

The full graph has 24 instances of Figure 2.2, connected through the reservoirs. The first and last network are slightly different in that the starting and remaining water in the reservoirs is be linked to the source and sink respectively. This can be seen in Figure 2.4, where there are 22 instances of the intermediate network in between the starting and final network, each corresponding to one hour.

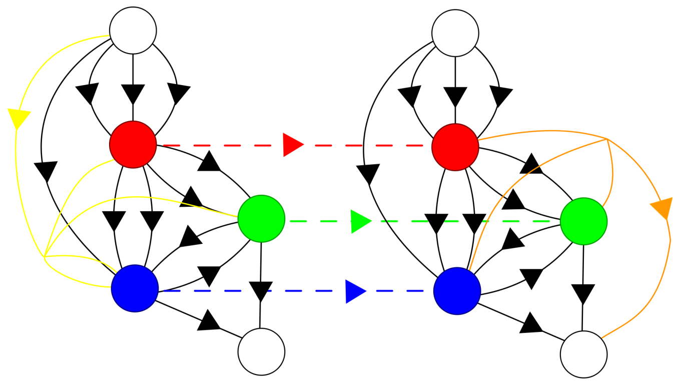


Figure 2.3: A two hour snapshot of the dynamic network. The red, green, and blue edges link the water levels in the reservoirs across time, and the yellow and orange edges are necessary for flow conservation in the first and last time step.

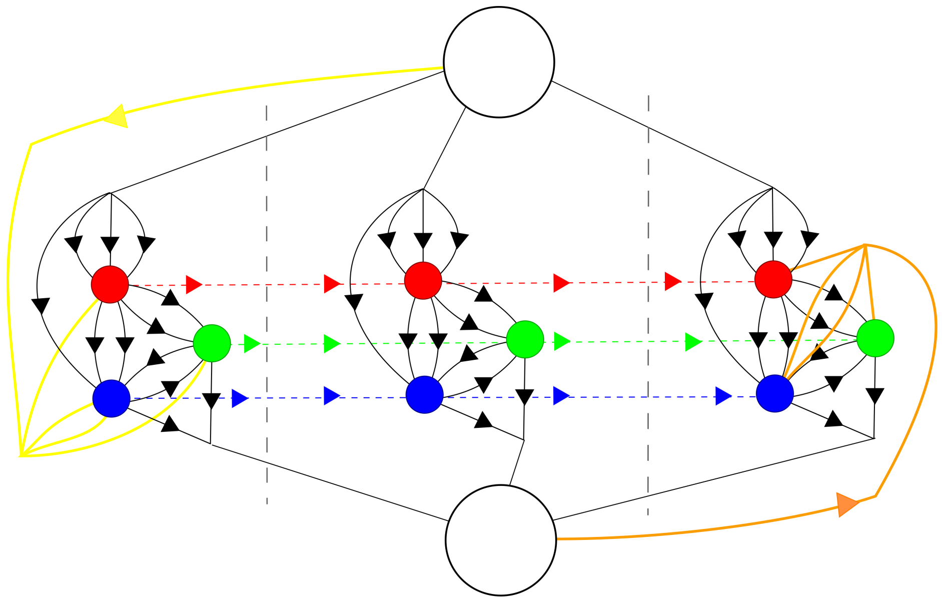


Figure 2.4: A representation of the full network. The full model would have 22 instances of the middle graph, connected through the reservoirs and through the common source and sink.

Chapter 3

The minimum cost flow problem

The real-world problem described in the introduction consists of the optimization of the costs associated with operating a water supply network. In the equivalent mathematical formulation described in Chapter 2, the problem consists of minimizing the flow across the edges of the network. This is a well known problem in network flow theory, known as the minimum cost flow problem (see for example Ahuja et al [2]). It is stated in the following way:

$$\begin{aligned} &\text{Minimize} \sum_{(i,j)\in A} C_{ij}x_{ij} \\ \text{subject to} &\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{for all } i \in N, \\ &l_{ij} \le x_{ij} \le u_{ij} \text{ for all } (i,j) \in A, \\ &\sum_{i=1}^{n} b(i) = 0. \end{aligned} \tag{3.1}$$

The first sum corresponds to the objective function that we want to minimize. The c_{ij} values correspond to the cost associated with each edge (i, j) from the set A of all edges of the network, and x_{ij} is the flow through each edge once the optimization is performed. The second condition and the final condition on the sum of all the $b(i)$ are essentially Kirchhoff's laws, with the added constraints that the flow through each edge has to be between a lower and an upper bound.

If our problem was to minimize the costs of water flowing though the network, this formulation would be enough. In fact, this is the first problem that we solved. However, the full problem requires us to also minimize the number of times that the pumps are turned on and off. In further chapters we will discuss how we tackled this problem, both with a heuristics and by expanding the network.

The list of variables in our problem is given in Table 3.1. In reality we have $12\times24-3$ variables, as the flow though each edge can in principle change at each hour.

Variable	Cost	Description
x_F	0	Flow from the feeder
p_W	c_W	Flow from the well (pump)
x_{WS}	c_{WS}	Flow from the water supplier
p_{1a}	c_{1a}	Flow though pump $P_{1,a}$ (red to green subnet)
p_{1b}	c_{1b}	Flow though pump $P_{1,b}$ (red to green subnet)
p_{2a}	c_{2a}	Flow though pump $P_{2,a}$ (red to blue subnet)
p_{2b}	c_{2b}	Flow though pump $P_{2,b}$ (red to blue subnet)
p_3	c_3	Flow though pump P_3 (green to blue subnet)
v	0	Flow though the valve
x_R	0	Residue from previous hour in the red subnet
x_G	0	Residue from previous hour in the green subnet
x_B	0	Residue from previous hour in the blue subnet

Table 3.1: Variables and their associated costs

As a first approach, we reformulated Equation 3.1 into the following system:

$$\text{Minimize} \sum_{n=1}^{24} \left(\sum_{\substack{\text{pumps} \\ p}} c_p^n \cdot p^n + c_{\text{WS}}^n \cdot p_{\text{WS}}^n \right)$$

while satisfying the network Kirchhoff laws:

$$\sum_{n=1}^{24}(x_F^n + p_W^n + x_{\text{WS}}^n) = \sum_{n=1}^{24}(-S^n + T_1^n + T_2^n)$$

$$\begin{aligned} p_W^n + x_{\text{WS}}^n - p_{1a}^n - p_{1b}^n - p_{2a}^n - p_{2b}^n + x_R^{n-1} - x_R^n &= -S^n \\ p_{1a}^n + p_{1b}^n - p_3^n + v^n + x_G^{n-1} - x_G^n &= T_2^n \qquad (3.2) \\ x_F^n + p_{2a}^n + p_{2b}^n + p_3^n - v^n + x_B^{n-1} - x_B^n &= T_1^n, \qquad (3.3) \end{aligned}$$

and the constraints on the flows.

Here S^n, T_1^n and T_2^n denote the amounts of flow from the spring, and the demands of consumers 1 and 2, at hour n, respectively.

As an initial simplification we assumed that pumps were continuous, as this effectively turned the system in Equation 3.2 into a linear problem, which could be easily solved numerically with any linear programming solver.

There is an additional objective stated in the original problem, which involves minimizing the number of pump switches. If there was a way to express the number of pump switches in linear form involving the flows (x_i), then the problem could also be solved using linear programming tools. The full problem considering true binary pumps cannot be solved only with this approach. In Chapter 5 we discuss two different methods that we used to find the minimum cost while considering binary pumps.

Chapter 4

Solving the Linear Program

We solved the system described at the end of Chapter 3 by writing it as a linear programming problem. We utilized two different programs, `lpsolve` and `ZIBopt` (see Appendix B). Both packages used a version of the simplex method (see Reference [6]), and were both able to deal with integer and mixed-integer linear programs, which came in handy when we solved the system with binary pumps. We initially solved the linear system with `lpsolve`, but moved to `ZIBopt` as it was easier to extend for dealing with binary pumps, and provided better performance.

The linear system with continuous variables hase 285 variables and 73 constraints (without considering the bounds on the edges). The code ran in milliseconds, and the minimized objective function (the cost) was 4137.5. The system did not try to minimize pump switches. The number of pump switches was 18, or 16 if pumps that are turned on at the beginning or end of the day are considered as being on through the previous or next day, respectively.

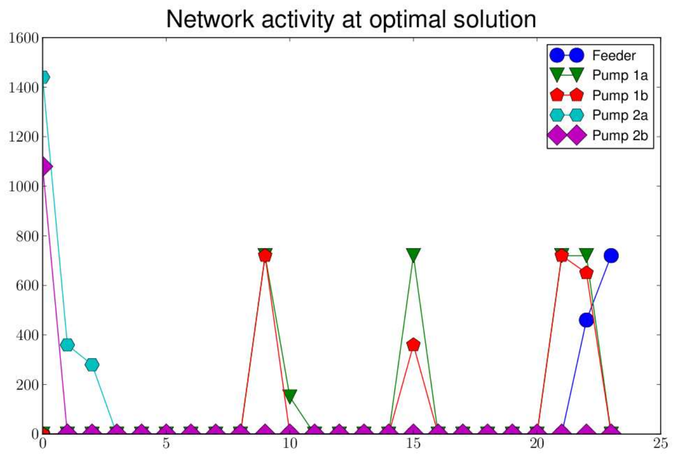


Figure 4.1: Each of the active pumps in the 24 hour interval at the optimal linear solution.

The plot of network activity at the optimal solution is given in Figure 4.1. As evidenced from the graph, several variables remain unused throughout the entire day. For example, the external supplier is never utilized, as the water from this particular source is significantly more expensive than any other source. There are others that have no activity, with the valve perhaps being the most notorious, as there is no cost associated with water flowing through it.

As a numerical experiment, we calculated the maximum amount of water that could be requested by the customers based on the information about the sources, which came out to about 6 times the water in the original data. We then ran the optimization code with the demand at full capacity.

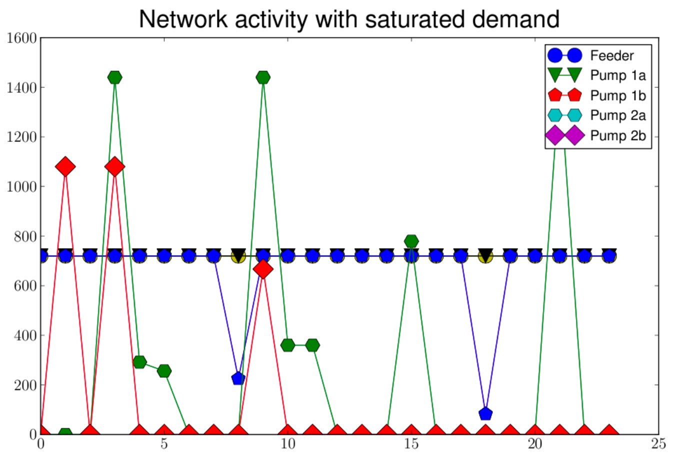


Figure 4.2: Each of the active pumps in the 24 hour interval with fully saturated demand.

Most interestingly, both pump 3 and the valve remained unused in Figure 4.2, even with full utilization of the external supply. The total number of pump switches in this scenario was 22.

4.1 Pump Scheduling

A first approach in considering discrete pumps is to consider the solution from the continuous problem. The flow through the pumps were considered as flowing at a fraction of the full capacity during the hour. However, with the discrete pumps, the flow through the pumps would be considered as flowing at full capacity during a fraction of the hour.

Figure 4.3 depicts a discrete pump scenario. The one potentially big caveat is that by considering pumps in this way, the reservoirs tend to overflow and underflow at certain points in the day. This can happen due to the fact that water is no longer flowing in and out of the reservoir uniformly throughout the hour. The reservoirs have safety tolerances (20% and 80%), and if the over/underflow is small enough, it could be tolerated in the real world.

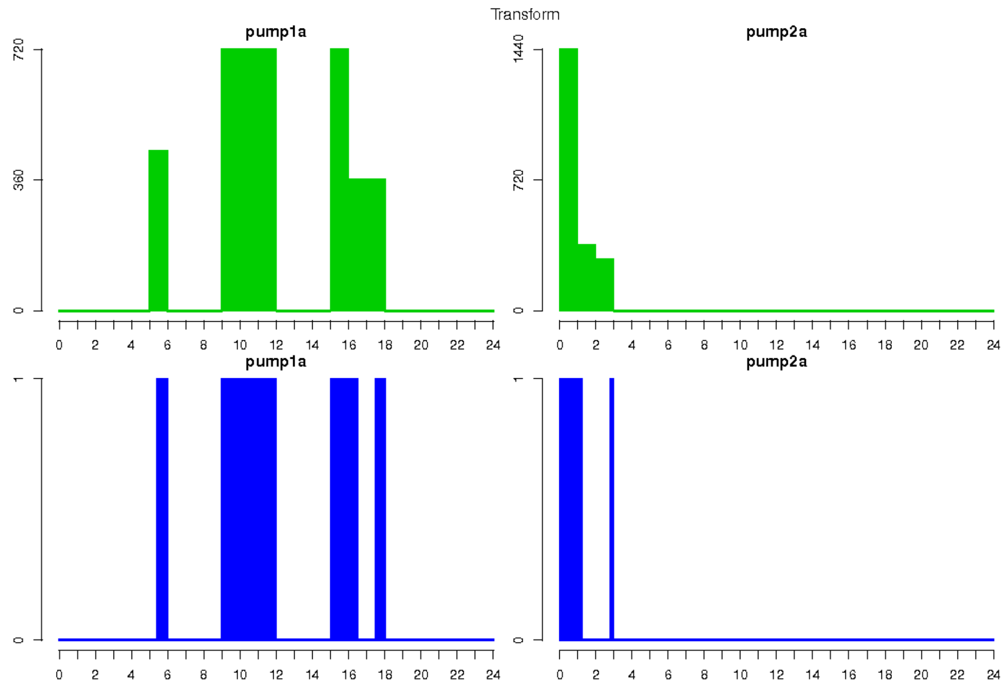


Figure 4.3: Going from continuous pumps, to discrete pumps working for fractions of an hour.

When the pumps are considered to be discrete, we implemented a scheduling algorithm that aimed to minimize the number of pump switches. Two pump utilizations back-to-back would count as two switches (on plus off), and as four (on, off, on, off) if they are separate.

In Figure 4.4, we plotted the reservoir levels for the original data and for the fully saturated demand. The fully saturated demand utilizes the greedy scheduling algorithm. This algorithm states that within any hour block, if there is flow within the pump and the pump did not run for the full hour, then we determine when to run the pump. If there is no water flow in the previous hour, then we start the pump during the hour so that it will run until the start of the next hour. If there was water flow from the previous hour, then we will run the pump until enough flow is required and then turn it off. We can see one underflow in original data, and a few more in the graph for saturated demand. All of them are quite close to the safety tolerances.

A more complete model would take this into account and optimize the pump switches in order to minimize over/underflows. Some suggestions on how to address this problem can be found in Chapter 5.

Figure 4.5 illustrates how the level in reservoir 3 changes throughout the day, and the activity of the pumps after being scheduled to minimize switches.

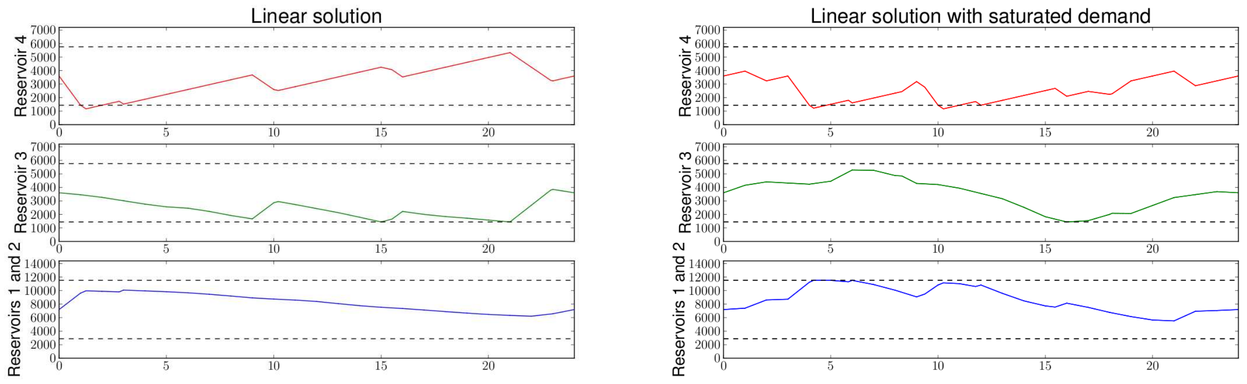


Figure 4.4: Reservoir levels with the pumps scheduled to minimize pump switches, for the original data (left), and for fully saturated demand (right)

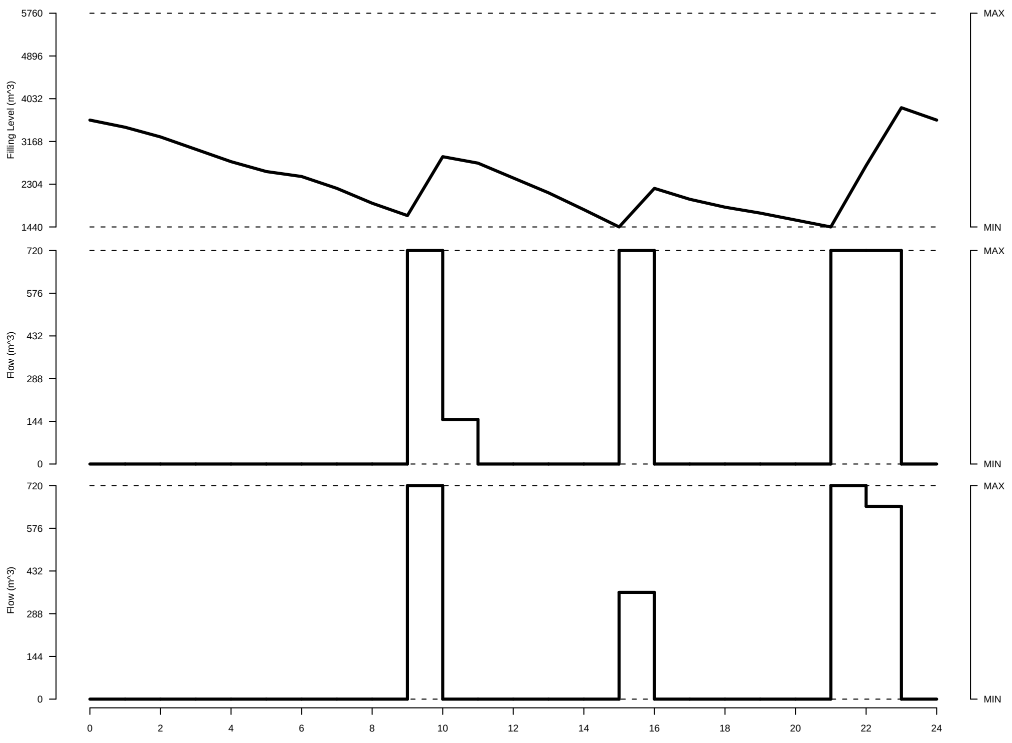


Figure 4.5: Reservoir 3 and pumps 1a and 1b (continuous case).

Chapter 5

Discrete Pumps and Pump Switching

Our first implementation made several simplifications on the problem. The first and foremost was the treatment of pumps as continuous variables, which allowed us to formulate the problem as a linear program. In reality, pumps are discrete: they may be switched on and off at minute intervals and must operate at full capacity while they are on. An even more crucial disadvantage is the lack of a convenient way of counting, or even approximating, the number of pump switches.

These considerations led us to the idea of modelling pumps as integer variables. We took two separate approaches, both with distinct advantages and disadvantages. The first approach is to enlarge our graph model to keep track of each pump for every minute, and record when it is turned on and off. The second approach is a heuristic that forces pumps to be either on or off for the duration of the entire hour. In such solutions, there is a simple way to count pump switches.

In both approaches, the effect of minimizing pump switches was achieved by introducing a cost to pump switches and adding it to the objective function. Since that cost is not part of the provided data, as it would be difficult to compute it in real-world scenarios, we introduced a variable λ and tested our models with λ set to various values in order to ascertain the solution space.

5.1 Enlarging the Graph

To inject a tiny amount of simplicity, we make the assumption that all pumps are off at the beginning of the day and should be off at the end of the day.

Let p be a pump from subnet A to subnet B. In the original graph, the amount of flow through p at hour n is modelled by an arc from A^n to B^n. We replace the arc by a new subgraph defined as follows:

For each minute i, introduce a new node $v_p^{n,i}$. Roughly speaking, this node represents the action that is performed with respect to a pump that is on at the i^{th} minute of the n^{th} hour. In other words, there is a decision to either turn the pump off or leave it on. The pump can be turned on at any minute of the hour, so we add an arc from A^n to $v_p^{n,i}$ with cost λ, since it will correspond to a pump switch. Similarly, a pump that is on can be turned off at any minute of the hour, so we add an arc from $v_p^{n,i}$ to B^n with cost λ, since it too corresponds to a pump switch. The other option is to leave the pump on until the next minute, in which case it will incur the cost of running the pump for one minute in hour n, which we denote by c_p^n. Hence, assuming $i \neq 60$ (the last minute of the hour), we add an arc from $v_p^{n,i}$ to $v_p^{n,i+1}$, with cost c_p^n. In the case that $i = 60$, and $n \neq 24$, we may still leave the pump on, in which case it will be on at the first minute of the next hour. In other words, we add an arc from $v_p^{n,60}$ to $v_p^{n+1,1}$ with cost c_p^n. All the arcs have boolean capacities, and the nodes must retain none of the flow.

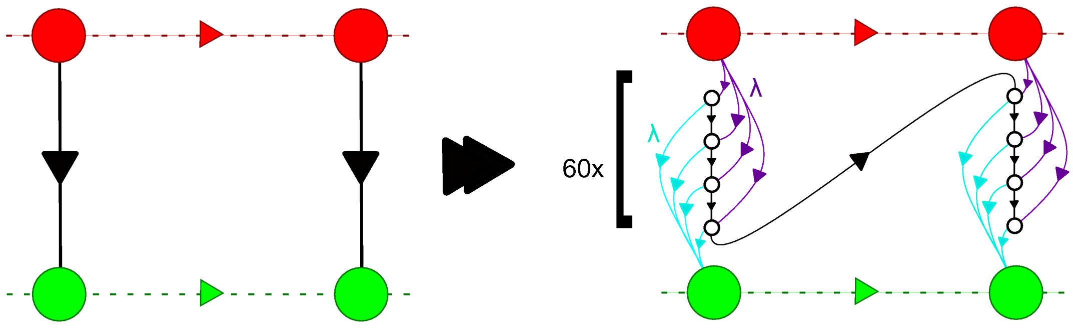


Figure 5.1: An example of the graph enlargement procedure. Purple arcs correspond to turning the pump on, teal arcs correspond to turning it off, and black arcs correspond to continuing to pump.

Any feasible flow of this graph will encode a full schedule for all the pumps, including times that each pump is to be turned on and off, modulo a possible situation where a pump is on from the previous minute and continues to operate, but also the on and off arcs are saturated (i.e. all the arcs in and out of $v_p^{n,i}$ are saturated). Of course, this will never occur in an optimal solution, which will provide an honest schedule.

The (mixed integer) linear program we solve is the standard one associated to the minimum cost flow through this graph (see Ahuja et al. [2]). We may also add extra constraints to ensure that even non-optimal feasible solutions provide actual schedules. Denote by $p_{\text{in}}^{n,i}$ and $p_{\text{out}}^{n,i}$ the (boolean) variables corresponding to turning the pump p on or off, respectively, at minute i of hour n. We add the constraints

$$p_{\text{in}}^{n,i} + p_{\text{out}}^{n,i} \leq 1,$$

which ensure that a pump is not turned on and off in the same minute.

An optimal solution to this graph fully solves the original problem (given a value for λ). Unfortunately, there is a downside in the form of complexity: the MILP contains 26061 variables, the vast majority of which are boolean. As a result, solving it takes quite a long time, and in fact, we had not successfully computed an optimal solution for any instance, and any value of λ that we tried. We were, however, able to compute feasible solutions that were within 3% of the optimal solution for almost all the instances we tried with reasonable values of λ, in relatively short periods of time (under 10 minutes). Our results are discussed in Chapter 6.

5.2 Boolean Pumps

The second approach is heuristic, with an attempt to keep the complexity of the problem low. This is done by adding the restriction that pumps cannot be turned on or off during the hour, they can only be switched at the beginning of the hour. In other words, the pump variables are forced to be boolean.

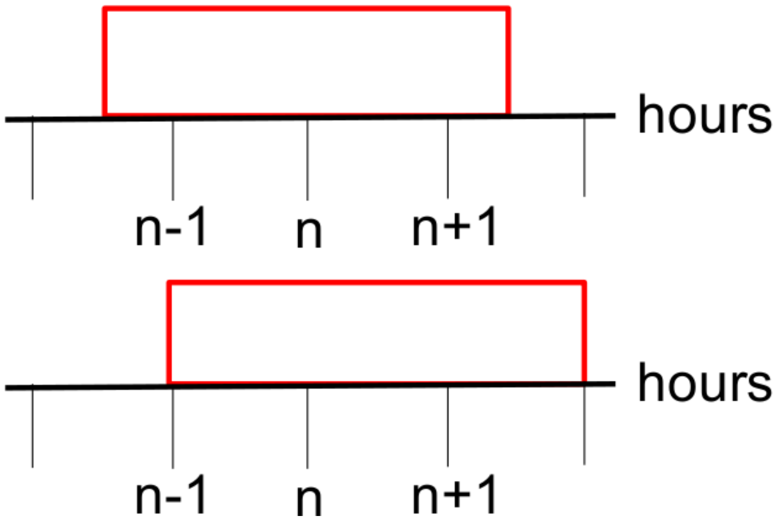


Figure 5.2: Here we can observe the effect on scheduling using normal pumps (top) and boolean pumps (bottom).

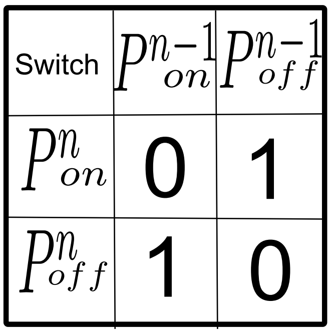
This restriction avoids the averaging of reservoir water at each hour. Instead, the water being pumped will always be either 0 or the maximum capacity of the pump.

An immediate consequence is that over/underflows are significantly less likely to occur. Most commonly, overflow occurs inside the hour due to the switching of pumps and valves, and contributions from the water supplier. By forcing pumps to run for the full hour, we are limiting the number of forces which are able to create sudden surges in water.

Another consequence of boolean pumps is that it gives an easy way to incorporate directly in to our optimization a way of counting and penalizing pump switches. We wish to use a linear function for detecting when a switch is turned on/off, so that we may continue to use linear programming to solve the problem. Ideally, we would like a function that behaves as in Figure 5.3. It is easy to check that the function

$$|p^n - p^{n-1}|, \; 2 \leq n \leq 24$$

satisfies the conditions, where p^n is the flow through pump p at hour n, but is only piecewise linear.



Switch	P^{n-1}_{on}	P^{n-1}_{off}
P^{n}_{on}	0	1
P^{n}_{off}	1	0

Figure 5.3: The ideal function for detecting a boolean switch.

To get around the linear restriction, we use a standard trick. For each pump p and hour n, introduce a variable $\hat{p}^n$ with the following linear restrictions:

$$\begin{aligned} &\hat{p}^n \geq p^n - p^{n-1}, \\ &\hat{p}^n \geq p^{n-1} - p^n, \\ &\hat{p}^n \in \{0, 1\}. \end{aligned}$$

The variable $\hat{p}^n$ implicitly defines the function we want. The restrictions, which depend on the difference of flow, will force the binary variable to take on values as in Figure 5.3.

Suppose the original objective function is F. We now augment it with pump switch penalties as follows:

$$F + \lambda \sum_{n=1}^{24} \sum_{\substack{\text{pumps} \\ p}} c_p^n \cdot \hat{p}^n,$$

where c_p^n is the cost of running pump p for the full hour n. Observe that the cost of a pump switch depends on the cost of pumping water through that pump. Heuristically this represent the fact that a machine that cost more to pump water probably will also be a more expensive machine, and hence we want to switch it less frequently .

This heuristic approach was faster than the graph enlargement, and was generally able to produce a solution that was within 1% of the optimal solution in less than 1 minute. The results are discussed further in Chapter 6.

Chapter 6

Comparison of Results

Following the formulation of the problem in Chapter 3, we computed an optimal solution for the relaxed, continuous problem, with an objective value of 4137.5. Our particular solution had 18 pump switches.

Using the enlarged graph from Chapter 5 and varying λ, we obtained the near optimal results for cost and number of pump switches as follows:

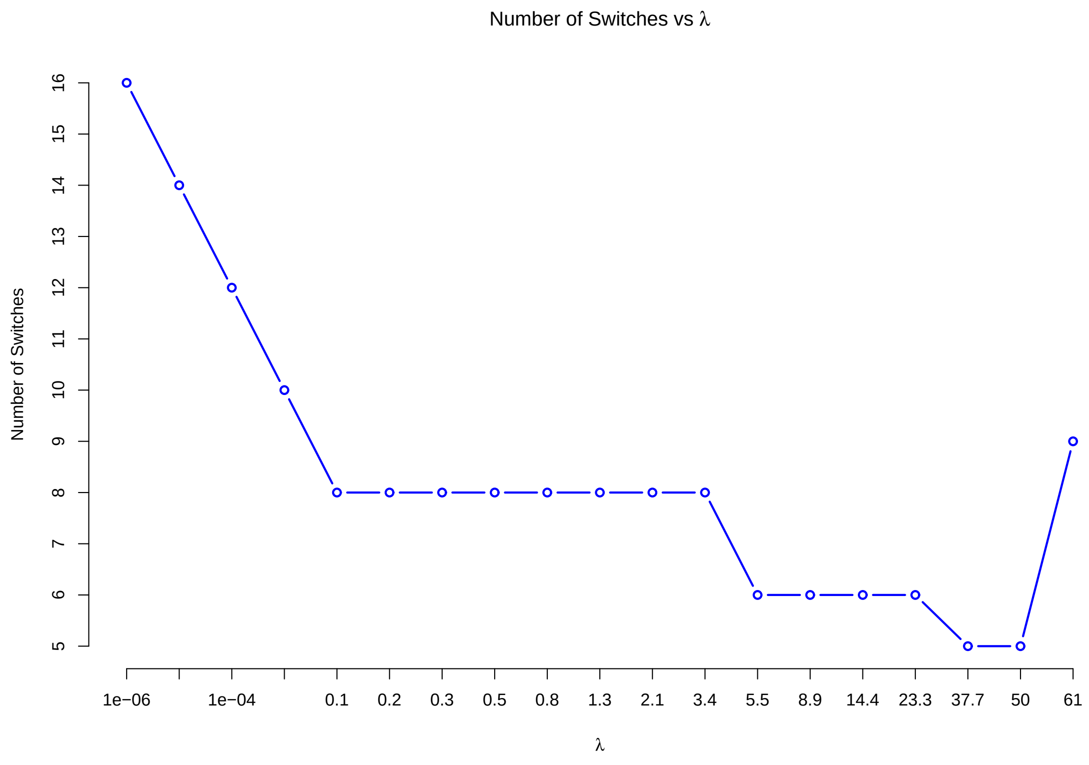


Figure 6.1: Number of Pump Switches vs. λ

Though the results are not verifiably optimal, we may still infer some general be-

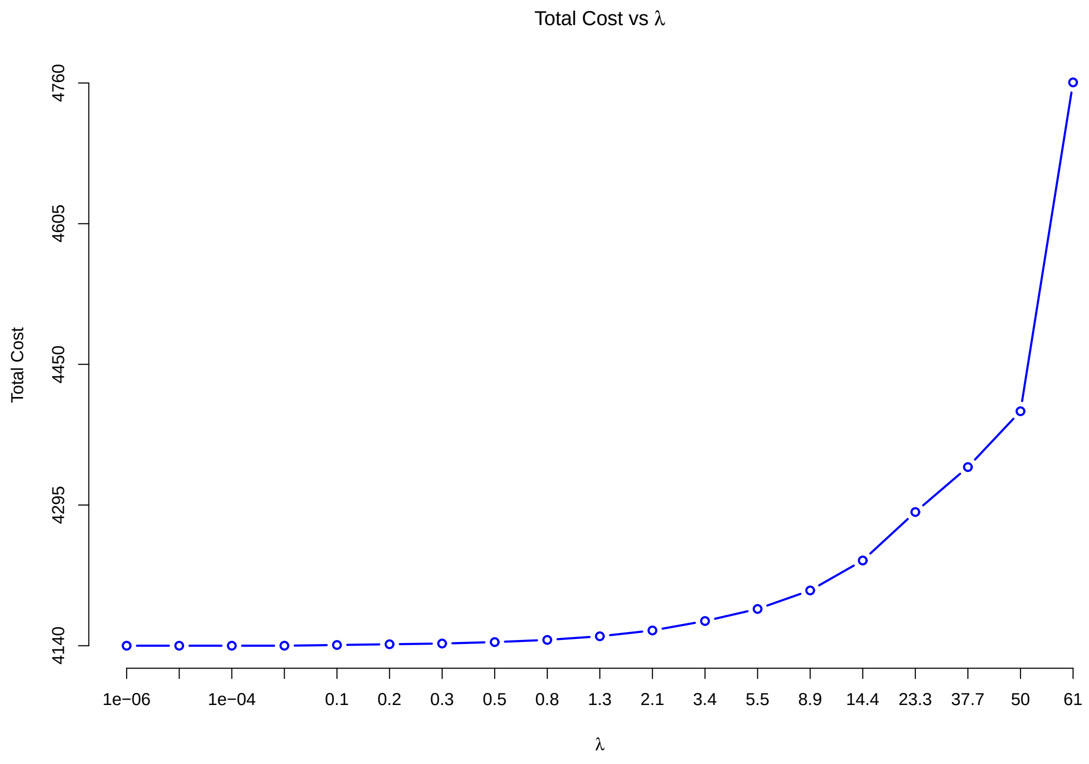


Figure 6.2: Total Cost vs. λ

haviour from them. Figure 6.1 shows that the optimal number of pump switches decreases with λ. The last data point is almost certainly erroneous, and is a result of a suboptimal solution that was accepted because of termination rules imposed on the solver.

Unsurprisingly, total cost is strictly increasing with λ, but more interestingly, there are regions where the increase in cost does not change the solution. In a real-world scenario, this would be sufficient information to provide to a client company, and allow it to decide the appropriate trade-off between pump switches and total cost.

By enlarging the graph, we also get the optimal solutions for filling levels in each reservoirs:

A close inspection reveals that for $\lambda = 0.001, 0.1, 5.5, 37.7$, there is no overflow or underflow in any reservoir.

Using the boolean pumps heuristic, we computed the following near optimal solutions:

Similarly to the results from enlarging the graph, Figures 6.8 and 6.9 show that the number of switches generally decreases while total cost increases with λ. In Figure 6.10, we see that for the optimal number of switches, 4, we have the optimal cost 4425, and for 5 switches, the optimal cost is 4275. Comparing to the same number of switches obtained through enlarging the graph, the boolean pumps have a worse performance. There was also no overflow or underflow in any reservoirs with this data.

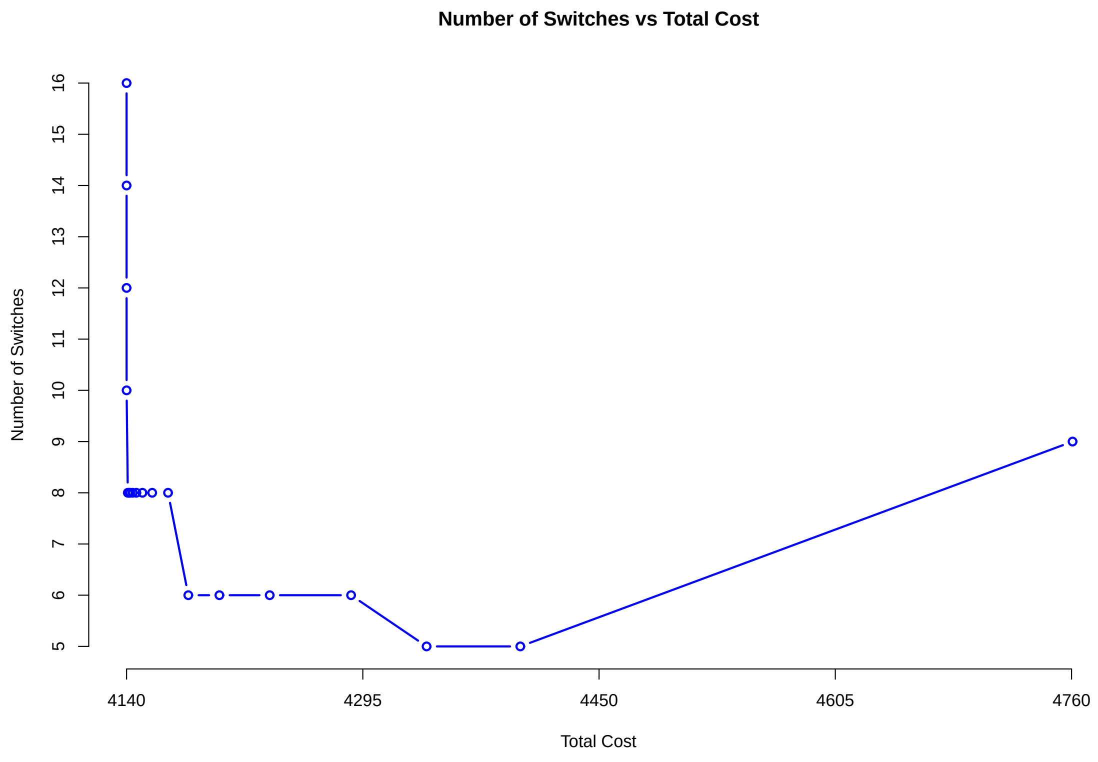


Figure 6.3: Number of Pump Switches vs. Total Cost

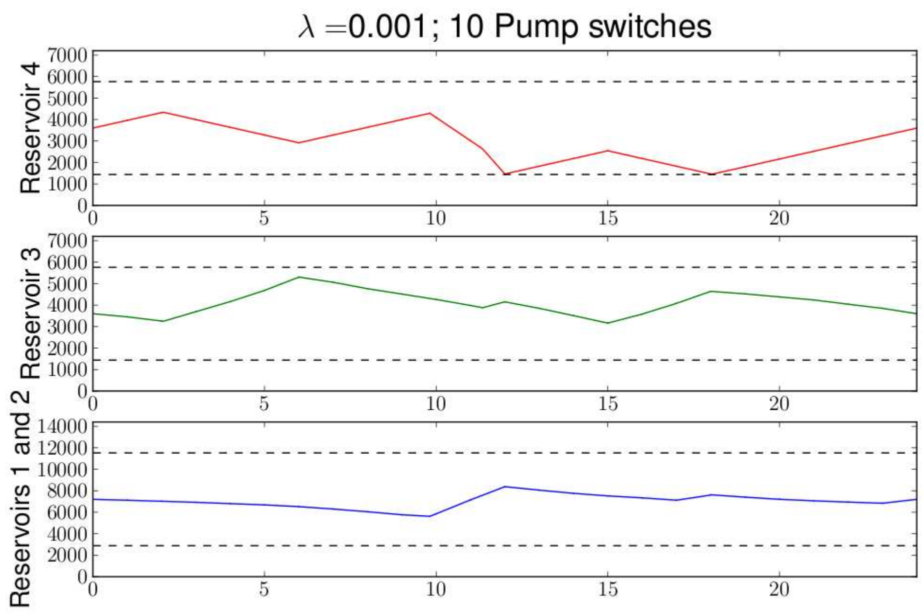


Figure 6.4: $\lambda = 0.001$

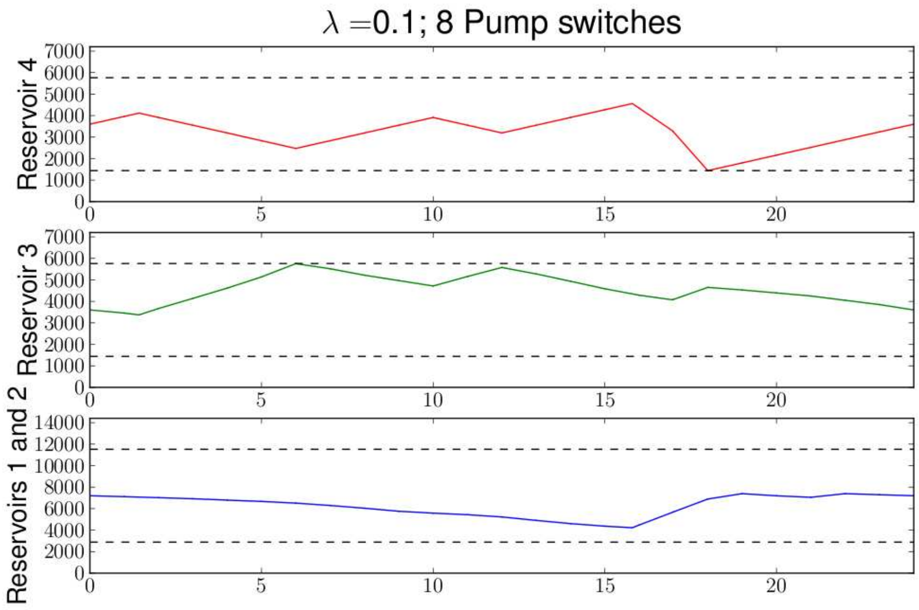


Figure 6.5: $\lambda = 0.1$

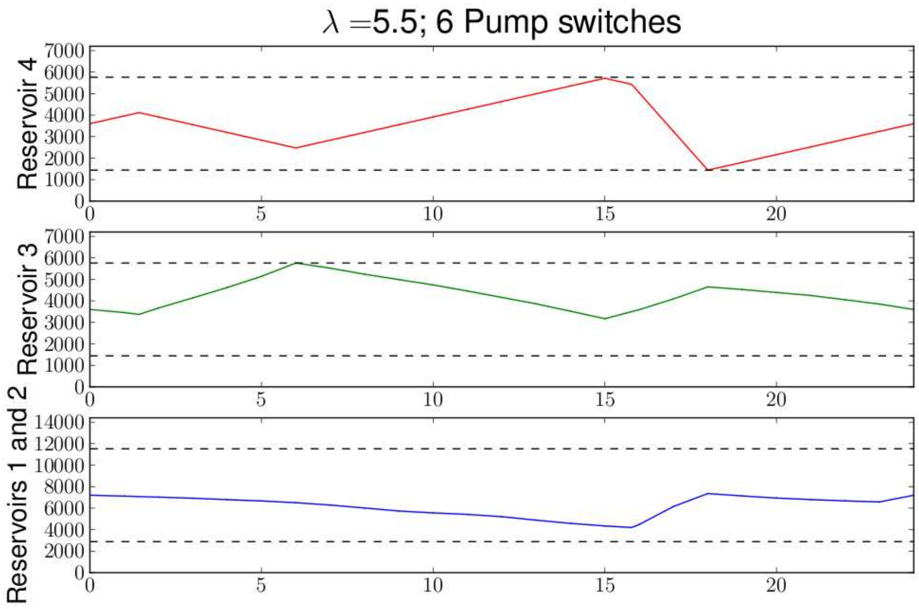


Figure 6.6: $\lambda = 5.5$

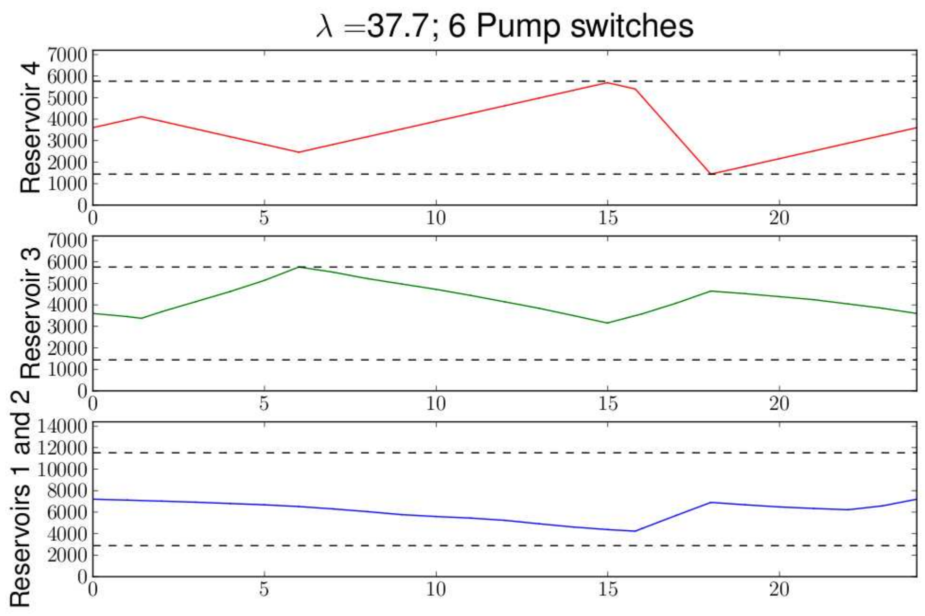


Figure 6.7: $\lambda = 37.7$

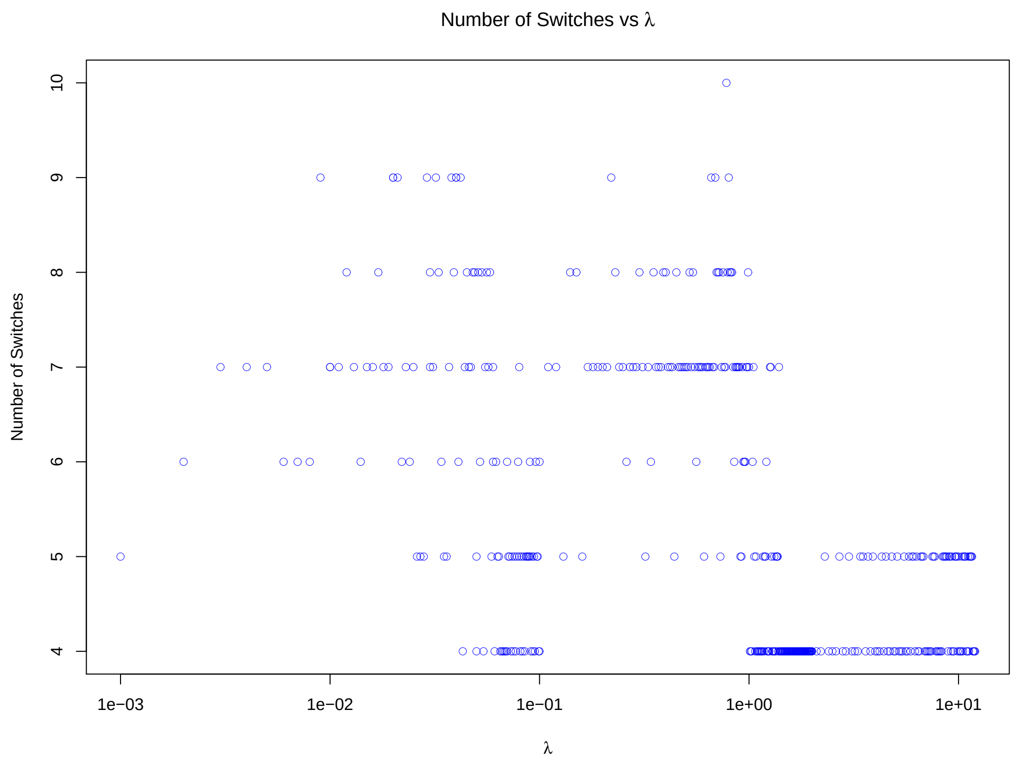


Figure 6.8: Number of Pump Switches vs. different λ

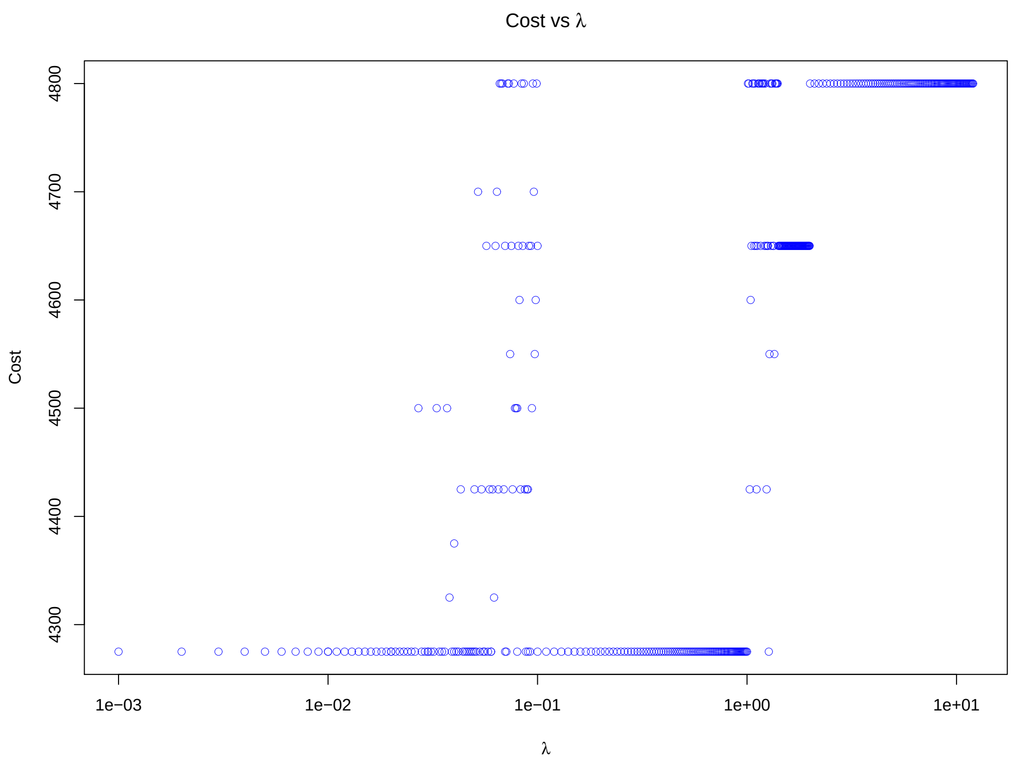


Figure 6.9: Total Cost vs. different λ

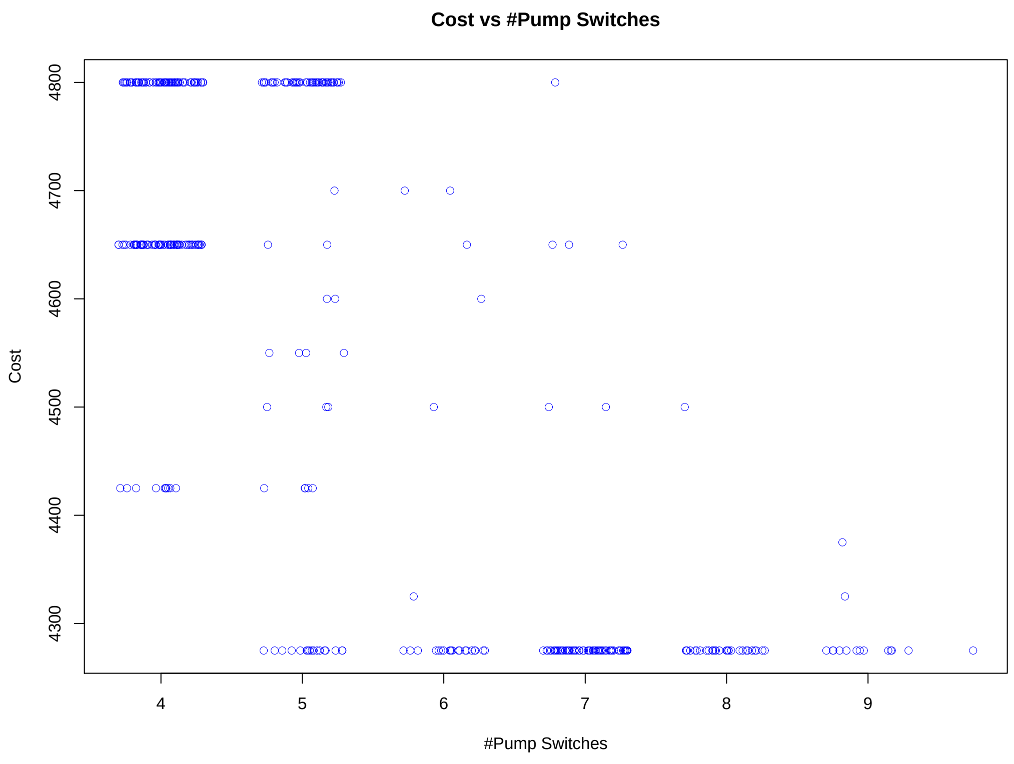


Figure 6.10: Total Cost vs. Number of Pump Switches

Chapter 7

Wishlist

The things which we wished we had more time to try out are the following:

- Pump switches that occur at the beginning or at the end could actually be counted as only one switching, by assuming that the pump has been on since the day before, or that it will stay on the next day. Prioritizing for this pump switches at these intervals could potentially lead to solutions with less pump switches, but we didn't have enough time to modify the objective function.
- Most of the solutions didn't have any over(under)flows, however there could be other scenarios where this become a more important issue. We didn't specifically optimize against over(under)flows, but considering that safety should be the first priority in the simulation, this is definitely an important issue to consider.
- We mostly only considered solutions from the provided data, but while playing around with a few values, particularly the demand, we noticed that more complicated scenarios could arise.
- It would be interested to consider the possibility of network failure, and model optimizing for a degree of robustness.

Chapter 8

Acknowledgements

Our team would like to thank the Institute for Mathematics and Its Applications (IMA), the Pacific Institute for the Mathematical Sciences (PIMS), and the Center for Mathematics Research (CIMAT) for providing us the opportunity to participate in the workshop. We would also like to thank our mentor, Dr. Michael Hofmeister, for guiding our progress and providing insight throughout our project.

Appendix A

The Data

Source	Maximum capacity (m^3/h)	Cost (ct/m^3)
Special Feeder	720	0.00
Well	720	Depends on the time
External Supplier	720	Depends on the time
Spring	360	0.00

Table A.1: Maximum flow capacities and costs for the sources.

Reservoirs	Minimum filling level (m^3)	Maximum filling level (m^3)
1	1440	5760
2	1440	5760
3	1440	5760
4	1440	5760

Table A.2: Maximum and minimum filling levels allowed for the reservoirs. The allowed values are the 20% and 80% for each reservoir.

Pump – Valve	Maximum capacity (m^3/h)	Energy consumption (kWh)
Pump 1a	720	5.00
Pump 1b	720	5.00
Pump 2a	1440	40.00
Pump 2b	1080	30.00
Pump 3	1080	30.00
Pump Well	720	5.00
Pump Spring	360	5.00
Valve	3960	0.00

Table A.3: Maximum flow capacities and electrical comsumption for the pumps and the valve.

Time interval	Region 1 (m^3)	Region 2 (m^3)
1	85	145
2	95	195
3	100	250
4	120	250
5	120	200
6	160	100
7	220	240
8	255	300
9	280	250
10	175	250
11	145	280
12	210	300
13	320	300
14	300	340
15	240	350
16	180	300
17	220	220
18	240	160
19	220	120
20	200	140
21	140	140
22	120	200
23	100	200
24	95	250

Table A.4: Forecasting for water demand.

Time interval	External Supplier	Pump 1a	Pump 1b	Pump 2a	Pump 2b	Pump 3	Well	Spring
1	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
2	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
3	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
4	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
5	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
6	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
7	32	0.17	0.17	0.69	0.69	0.69	0.17	0.35
8	32	0.17	0.17	0.69	0.69	0.69	0.17	0.35
9	15	0.17	0.17	0.69	0.69	0.69	0.17	0.35
10	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
11	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
12	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
13	15	0.17	0.17	0.69	0.69	0.69	0.17	0.35
14	15	0.17	0.17	0.69	0.69	0.69	0.17	0.35
15	15	0.17	0.17	0.69	0.69	0.69	0.17	0.35
16	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
17	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
18	35	0.10	0.10	0.42	0.42	0.42	0.10	0.21
19	35	0.17	0.17	0.69	0.69	0.69	0.17	0.35
20	15	0.17	0.17	0.69	0.69	0.69	0.17	0.35
21	15	0.17	0.17	0.69	0.69	0.69	0.17	0.35
22	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
23	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21
24	15	0.10	0.10	0.42	0.42	0.42	0.10	0.21

Table A.5: Costs of flowing a cubic meter.

Appendix B

The Software

The selected software for programming was `Python`. This is an interpreted, interactive, and object-oriented programming language, which has interfaces to many system calls and libraries; as well as to various window systems, and is extensible in C and C++. Moreover, `Python` is portable: it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.

`lpsolve` is a library that solves Mixed Integer Linear Programming (MILP) problems. The `lpsolve` library can be called from many programming languages, one of them being `Python`. `lpsolve` is written in ANSI C and can be compiled on many different platforms like Linux and WINDOWS. This library was used to solve the relaxed problem.

`ZIBopt` was used to solve the non-continuous pump switches. `ZIBopt` is a library for generating and solving Mixed Integer Programs, using SCIP and SoPlex from the ZIB Optimization Suite.

The graphs for reservoirs and pumps were made with the statistical software `R` and the `matplotlib` library in `Python`. This document was written with LaTeX.

`Python`, `lp_solve`, `ZIBopt`, `R`, and LaTeX are free software and their licenses are related to the GNU General Public License.

Appendix C

The simplex algorithm

Each constraint in the LP represents a half space in the given space for the set. The intersection of all the half hyperplanes representing the constraints forms a polyhedron, which is the set of all feasible solutions to the LP. The simplex algorithm starts by finding a vertex on the polyhedron, and then traverses the hyperplanes of the figure until it finds a neighbouring vertex that improves the solution. The path to the optimal solution is not unique, but since the LP is convex, the optimal solution itself will be unique. A computationally important idea of the simplex method is that given a matrix A of dimension (m, n) with full row rank m, and a vector b of length m and a vector c of length n, $m < n$.

$$\begin{aligned}
&\max \quad c^T x \\
&Ax = b \\
&x > 0
\end{aligned}$$

Then for every vertex y of the polyhedron of feasible solutions of the LP, there is a non-singular sub-matrix B of dimension $m \times m$, which is called a basis of A, representing the vertex y, the basic solution, such that $y_B = B^{-1}b, y_N = 0$. Numerical issues arise often due to geometric reasons, such as that the subspace of feasible solutions having no points of intersection.

Bibliography

[1] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[3] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lp_solve 5.5, open source (mixed-integer) linear programming system. Software, May 1 2004. Available at ¡http://lpsolve.sourceforge.net/5.5/¿. Last accessed Dec, 18 2009.

[4] R. Diestel. Graph theory, electronic edition. http://diestel-graph-theory.com/index.html, 2010.

[5] W. Domschke and A. Drexl. *Einführung in Operations Research.* Springer, 1988.

[6] G. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.