Solving the Salinity Control Problem in a Potable Water System*

Chiu Wo Choi and Jimmy H.M. Lee

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{cwchoi,jlee}@cse.cuhk.edu.hk

Abstract. Salinity is the relative concentration of salts in water. In a city of southern China, the local water supply company pumps water from a nearby river for potable use. During the winter dry season, the intrusion of sea water raises the salinity of the river to a high level and affects approximately the daily life of 450,000 residents of the city. This paper reports the application of constraint programming (CP) to optimize the logistical operations of the raw water system so as to satisfy the daily water consumption requirement of the city and to keep the potable salinity below a desirable level for as many days as possible. CP is the key to the success of the project for its separation of concerns and powerful constraint language that allows for rapid construction of a functional prototype and production system. Flexibility and adaptiveness allow us to deal with our clients' many changes in the requirements. Deriving good variable and value ordering heuristics, and generating useful implied constraints, we demonstrate that branch-and-bound search with constraint propagation can cope with an optimization problem of large size and great difficulty.

1 Introduction

Salinity is the relative concentration of salts in water measured in parts per million (ppm). All types of water, except distilled water, contain different concentration of salts. The salinity of very clean water is about 50 ppm, while sea-water is about 35,000 ppm.

In a city of southern China, the local water supply company pumps water into a raw water system from a nearby river for supplying water to the city. The pumped water is to be stored and mixed with water in a number of reservoirs in the raw water system. The water is also treated before supplying to the general public for daily consumption.

The geographic location of the pumping station is close to the river estuary. During the winter dry season, the water level of the river is low due to lack of rainfall. Tidal flows and other weather conditions lead to the intrusion of sea-water into the river. As a result, the salinity of the water pumped from the river could drastically rise to such levels as 2,500 ppm while the desirable salinity level of potable water is below 250 ppm. During the *salinity period*, the daily life of some 450,000 residents is affected.

* We thank the anonymous referees for their insightful comments. The work described in this paper was substantially supported by a grant (Project no. CUHK4219/04E) from the Research Grants Council of the Hong Kong SAR.

There are a number of ways to better prepare for the crisis. On the engineering side, the water company can improve the monitoring of the salinity levels and the pumping system. Reservoirs can be topped up with fresh water before the dry season begins. Better leak detection at the water pipes and the reservoirs can reduce water loss.

Before attempting on larger scale engineering work, such as seawater desalination, the water company decided to tackle the salinity issue as an optimization problem. The idea is to carefully plan when and how much water to pump from the river using supplied prediction information of the salinity profile at the water source, and how much water should be transferred among the reservoirs in the raw water system. The aim is to satisfy the daily water consumption while optimizing the number of days in which the salinity of the potable water is below a given desirable level.

In the beginning of this project, the water supply company required us to handle at most 90 days for the duration of the salinity period. Later, upon receiving satisfactory preliminary results, the water supply company requested us to extend the duration to at most 180 days, the problem model of which consists of about 4,500 variables and 9,000 constraints. The search space of such model is about $(3,612,000)^{180}$. In addition to the shear size, the problem consists of physical conditions expressible as a mixture of linear and non-linear constraints, as well as ad hoc conditions which can only be modeled as a table constraint. In view of the stringent requirements and tight production schedule, we adopt constraint programming (CP) as the key technology of the project, following the success of the CLOCWiSe project [1].

The rest of this paper is organized as follows. Section 2 discusses the current practice and why constraint programming (CP) is used in this project. Section 3 details the application domain. Operations of the raw water system, as well as the objective of the problem, are described in length. Section 4 describes how CP is applied to model the problem. Section 5 describes the improvements to increase search efficiency, followed by a discussion of some testing results in Section 6. In Section 7, we discuss the added values of CP and other possible approaches that have been tried to solve the problem. We conclude the paper in Section 8.

2 Current Practice Versus Constraint Programming

The water supply company has developed a spreadsheet to optimize the operations of water pumping and transfer during the salinity period. The spreadsheet approach is primitive and uses manual trial-and-error method to perform optimization. The spreadsheet consists of macros that encode equations on the law of conservation of matters. Users of the spreadsheet have to input the given data and guess some values for the number of pumping hours and amount of water to be transferred between reservoirs. The macro will then compute automatically the potable salinity using the given inputs. Users have to check whether the resulting potable salinity is satisfactory; if not, the guessed values must be manually tuned repeatedly until a satisfactory result is obtained.

The major weakness of such manual method is that it is tedious and time consuming. The problem on hand is usually too large and too complex for humans to perform such manual optimization process. Users of the spreadsheet often obtain solutions that violate some constraints of the problem, since some constraints stated above cannot be enforced automatically using a spreadsheet. Field workers in the pumping stations and reservoirs often lack the knowledge of operating a spreadsheet.

The water supply company would like to have an automated system with a more realistic model and a simple interface so as to generate solutions which satisfy all the constraints of the problem. Moreover, the system should be flexible enough to cater for changes in the topology of the water system and additional constraints.

We propose the application of CP to develop an automated optimization engine for solving the salinity problem. A key advantage of CP is the separation of modeling and solving. By modeling, we mean the process of determining the variables, the associated domains of the variable, the constraints and the objective function. The availability of a rich constraint language allows for a constraint model relatively close to the original problem statement, making the model easy to verify and adaptable to changes. Indeed, during the development of the system, our client changed the constraints and requirements a good many times. CP allowed us to change the model quickly and meet the tight development schedule.

Although efficient commercial constraint solvers are available, out-of-the-box execution strategies usually fail to handle even small testing instances of the problem. We make two improvements to speed up solution search and quality of solutions. First, by studying the problem structure and insights of human experts in depth, we devise good search heuristics for both variables and values that allow us to find solutions faster. We also program an opportunistic iterative improvement strategy. Second, we give a general theorem that allows us to derive useful implied constraints from a set of linear equalities. Adding these implied constraints into the model can increase the amount of constraint propagation, which in turns reduces the search space substantially.

Our application exemplifies the advantage of separation of concerns offered by CP. After the problem model was constructed, we never had to touch the model again except when users requested changes in the requirements. The focus of the development is thus on improving search and looking for better heuristics.

3 Application Domain Description

The entire water supply system consists of the raw water system, water treatment plant and potable water distribution network. In the raw water system, water is pumped from the river and carried to the treatment plant. Surplus water are stored in reservoirs for emergency and salinity control during dry season. In order to ensure that the water supplied to the city is safe for drinking, raw water is treated in the treatment plant before being carried by the distribution network to general households and commercial establishments. It is important to note that the water treatment plant is incapable of removing salt from the raw water since salt is highly soluble and tends to stay dissolved. In this project, we focus on optimizing the logistical operations of the raw water system to control the salinity of potable water.

3.1 The Raw Water System

Figure 1 shows the topology of the water supply system. The raw water system consists of 3 pumping stations (X, Y and Z) denoted by black dots, 4 reservoirs (A, B, C and

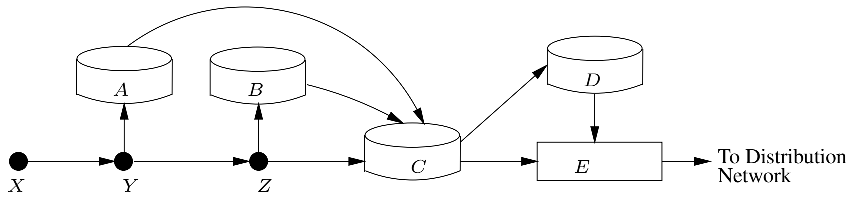

Fig. 1. The Raw Water System Model

D) denoted by cylinders and a water treatment plant (E) denoted by a rectangle. Arrows denote connecting pipes and the direction of water flow. Raw water (from the river) is pumped at pumping station X and carried all the way to reservoir C for storage. Surplus water is delivered, via pumping stations Y and Z, to and stored in reservoirs A and B respectively for future use. Water in reservoirs A and B can be transferred and mixed with water in reservoir C for regulation of salinity of water during the dry seasons. Water is carried from reservoir C to reservoir D for storage and to the water treatment plant E which is connected directly to the distribution network. In the water treatment plant E, water from reservoir C can be mixed with water from reservoir D. Water is treated in the water treatment plant E before being supplied to the general public for consumption.

There are several (reasonable) assumptions made by the water supply company for the raw water system to simplify the computational model. The unit of measurement for volume is cubic meter (m^3) and the unit of measurement for operation of pumps is in hours. Salinity concentration in each reservoir is homogeneous and instantaneous mixing occurs when water is poured in each reservoir. There is little rainfall during the dry seasons and the river is the only source of raw water. The computational model operates on a day-by-day basis, so that the predicted salinity data represent daily averages. Since salinity of the raw water varies during a single day, operators at the pumping station would use their experience to decide the best time during the day to pump water with lower salinity.

3.2 Physical and Human Constraints

There are three types of constraints concerning the raw water system. The first type of constraints is about the law of conservation of matters (i.e. water and salts). The general form for the law of conservation of water of a reservoir is

$$\text{volume today} = \text{volume yesterday} - \text{volume flow-out} + \text{volume flow-in}. \quad (1)$$

Analogously, the general form for the law of conservation of salts of a reservoir is

$$\begin{aligned}(\text{salinity today} \times \text{volume today}) &= (\text{salinity yesterday} \times \text{volume yesterday}) - \\ (\text{salinity flow-out} \times \text{volume flow-out}) &+ (\text{salinity flow-in} \times \text{volume flow-in}) \quad (2)\end{aligned}$$

The second type of constraints is about physical limitation on the capacity of pumps, reservoirs and pipes. Each pumping station has a maximum number of usable pumps and each pump has a given capacity measured in cubic meters per hour. Each reservoir has a

Table 1. Constraint on Water Flowing Out of Reservoir C

Volume of Reservoir C (m^3)	Maximum Flow-out Capacity of Reservoir C (m^3/day)
2,345,650 – 2,454,590	211,395
⋮	⋮
1,200,000 – 1,256,250	160,445

minimum and maximum capacity. It is impossible to pump water out of a reservoir when it is at the minimum capacity, and overflowing a reservoir at its maximum capacity for dilution is forbidden. Each reservoir also has a volume threshold which reserves certain amount of water above the minimum capacity for emergency use. The volume threshold for each reservoir is different from one day to another, and the volume threshold overrides the minimum capacity. The pipes, which connect reservoirs A and B to reservoir C and the pipe which connects reservoir D to the treatment plant E, have a maximum capacity measured in cubic meters per day.

Flowing from reservoir C located at a high topographical level, water is carried by gravity to reservoir D and the water treatment plant E. Therefore, the maximum amount of water that can flow out of reservoir C depends on the water pressure which decreases as the water level of reservoir C goes down. Due to the complex nature of the physics behind the water transfer mechanism, the constraint is given in the form of a table constructed empirically using measurement and experimentation. The water supply company provides a table (see Table 1) to specify such constraint.

The third type of constraints is about the requirements of the general public on water consumption. It is mandatory to have enough water supply to the general public everyday. There is a maximum level of potable salinity to ensure that water is safe for drinking. Between any two consecutive days, the salinity level of potable water should not increase too drastically; otherwise, the general public will feel a sudden increase in saltiness of drinking water and that will raise public discontent. There are *no corresponding constraints to restrict sudden decreases*, since drop in salinity is generally welcome by the public.

3.3 Problem Statement

To control the salinity of potable water, the water supply company needs to control carefully when and how much water is pumped from the river and how much water is transferred among the reservoirs. The aim is to satisfy all the constraints stated in Section 3.2 and to keep the salinity of potable water below a desirable level for as many days as possible during the salinity period. The given data include the initial volume and salinity level of reservoirs and the prediction[1] of salinity level of the river during the salinity period.

[1] The prediction of salinity level of the river is supplied to us by the water supply company. The prediction model is beyond the scope of this project.

4 Problem Modeling

Let n denote the duration of the given salinity period, (i.e. $n \leq 180$ days). Since values in our model are defined on a day-by-day basis, we have a set of variables for each day $i \in \{1, \ldots, n\}$, and each set contains seven variables. The first three variables are P_i^X, P_i^Y and P_i^Z which denote the number of pumping hours to operate at pumping stations X, Y and Z respectively. The other four variables are O_i^A, O_i^B, O_i^C and O_i^D which denote the amount of water flowing out of reservoirs A, B, C and D respectively.

4.1 Domains Discretization

The associated domains of the above variables are all continuous in nature, i.e. time for pumping hours and volume for water transfers. After consulting the water supply company, we learn that it does not make sense to operate the pumps for a very short time (e.g. 3 minutes) or to transfer a very small amount of water (e.g. 10 m^3). Therefore, we discretize the domains to reflect this reality and to reduce the search space. Assuming the pumps are operated in unit of ϕ pumping hours (e.g. $\phi = 6$ hours), the domains D of the 3 pump variables are

$$D(P_i^X) = \{0, \ldots, \lfloor N^X \cdot 24/\phi \rfloor\} \qquad D(P_i^Y) = \{0, \ldots, \lfloor N^Y \cdot 24/\phi \rfloor\}$$
$$D(P_i^Z) = \{0, \ldots, \lfloor N^Z \cdot 24/\phi \rfloor\}$$

where N^X, N^Y and N^Z denote the maximum number of usable pumps in pumping stations X, Y and Z respectively. Assuming water is transferred in unit of τ m^3 (e.g. $\tau = 5,000$m^3), the domains D of the 4 flow-out variables are

$$D(O_i^A) = \{0, \ldots, \lfloor F^A/\tau \rfloor\} \quad D(O_i^B) = \{0, \ldots, \lfloor F^B/\tau \rfloor\}$$
$$D(O_i^C) = \{0, \ldots, \lfloor F^C/\tau \rfloor\} \quad D(O_i^D) = \{0, \ldots, \lfloor F^D/\tau \rfloor\}$$

where F^A, F^B, F^C, and F^D denote the maximum amount of water that can flow out of reservoirs A, B, C, and D respectively. We also have a number of other variables but they are auxiliary in the sense that the values of the auxiliary variables are fixed once the values of the decision variables are known.

4.2 Constraints and Objective Function

To express the constraints on the law of conservation of water for the reservoirs, we derive the following constraints from Equation 1,

$$V_i^A = V_{i-1}^A - (O_i^A \cdot \tau) + I_i^A \tag{3}$$
$$V_i^B = V_{i-1}^B - (O_i^B \cdot \tau) + I_i^B \tag{4}$$
$$V_i^C = V_{i-1}^C - (O_i^C \cdot \tau) + (O_i^A \cdot \tau) + (O_i^B \cdot \tau) + I_i^X - I_i^A - I_i^B \tag{5}$$
$$V_i^D = V_{i-1}^D - (O_i^D \cdot \tau) + I_i^D \tag{6}$$

where V_i^A, V_i^B, V_i^C and V_i^D are auxiliary variables denoting the volume of the four reservoirs on day $i \in \{1, \ldots, n\}$; I_i^A, I_i^B, I_i^X and I_i^D are auxiliary variables denoting the amount of water to flow into the four reservoirs on day $i \in \{1, \ldots, n\}$. We express the amount of water pumps from the pumping stations using the constraints

$$I_i^A = P_i^Y \cdot \phi \cdot K^Y \quad I_i^B = P_i^Z \cdot \phi \cdot K^Z \quad I_i^X = P_i^X \cdot \phi \cdot K^X$$

where K^Y, K^Z and K^X denote the capacity of the pumps in pumping stations Y, Z and X respectively. We use the following constraints to express that there is only a single source of water flowing into reservoir D,

$$I_i^D = (O_i^C \cdot \tau) - U_i \qquad\qquad V_i^E = U_i + (O_i^D \cdot \tau)$$

where V_i^E denotes the amount of water consumption on day $i \in \{1, \ldots, n\}$, and U_i denotes the surplus water flowing out of reservoir C after some water is supplied for consumption .

To express the constraints on the law of conservation of salts for reservoirs A, B, C, D, we derive the following constraints from Equation 2,

$$\begin{aligned}
(S_i^A \cdot V_i^A) &= (S_{i-1}^A \cdot V_{i-1}^A) - (S_{i-1}^A \cdot O_i^A \cdot \tau) + (S_i^X \cdot I_i^A) \\
(S_i^B \cdot V_i^B) &= (S_{i-1}^B \cdot V_{i-1}^B) - (S_{i-1}^B \cdot O_i^B \cdot \tau) + (S_i^X \cdot I_i^B) \\
(S_i^C \cdot V_i^C) &= (S_{i-1}^C \cdot V_{i-1}^C) - (S_{i-1}^C \cdot O_i^C \cdot \tau) + (S_{i-1}^A \cdot O_i^A \cdot \tau) + \\
&\quad (S_{i-1}^B \cdot O_i^B \cdot \tau) + (S_i^X \cdot I_i^X) - (S_i^X \cdot I_i^A) - (S_i^X \cdot I_i^B) \\
(S_i^D \cdot V_i^D) &= (S_{i-1}^D \cdot V_{i-1}^D) - (S_{i-1}^D \cdot O_i^D \cdot \tau) + (S_{i-1}^C \cdot I_i^D)
\end{aligned}$$

where S_i^A, S_i^B, S_i^C, S_i^D are auxiliary variables denoting the salinity level of the four reservoirs on day $i \in \{1, \ldots, n\}$, and S_i^X is the (given) predicted value of salinity level of the river. We also need a constraint to specify the law of conservation of salts for potable water

$$(S_i^E \cdot V_i^E) = (S_{i-1}^C \cdot U_i) + (S_{i-1}^D \cdot O_i^D \cdot \tau)$$

where V_i^E and S_i^E denote the amount of water consumption and the potable salinity on day $i \in \{1, \ldots, n\}$. Note that the variables denoting salinity level are continuous, and the constraints associated to these variables involve both finite domain and continuous variables.

We can express the physical limitation on the volume of the reservoirs using the following constraints,

$$\begin{array}{ll}
V_{\min}^A + H_i^A \le V_i^A \le V_{\max}^A & \quad V_{\min}^B + H_i^B \le V_i^B \le V_{\max}^B \\
V_{\min}^C + H_i^C \le V_i^C \le V_{\max}^C & \quad V_{\min}^D + H_i^D \le V_i^D \le V_{\max}^D
\end{array}$$

where $V_{\min}^A$, $V_{\min}^B$, $V_{\min}^C$ and $V_{\min}^D$ denote the minimum capacity of the four reservoirs, $V_{\max}^A$, $V_{\max}^B$, $V_{\max}^C$ and $V_{\max}^D$ denote the maximum capacity of the four reservoirs, and H_i^A, H_i^B, H_i^C and H_i^D denote the volume threshold of the four reservoirs on day $i \in \{1, \ldots, n\}$.

The following set of constraints expresses the requirements given in Table 1,

$$O_i^C \le \begin{cases} 211,395 \;\; \textit{if} \;\; 2,345,650 < V_i^C \le 2,454,590 \\ \quad \vdots \qquad\qquad\qquad \vdots \\ 160,445 \;\; \textit{if} \;\; 1,200,000 < V_i^C \le 1,256,250 \end{cases}$$

We have intentionally used $<$ and $\leq$ to specify the bounds on each level to avoid potential conflict with domain discretization.

Last but not least, we have the following constraints to express the requirements of the general public on potable salinity,

$$S_i^E \leq S_{\max}^E \qquad\qquad S_i^E \leq S_{i-1}^E + \delta$$

where $S_{\max}^E$ denotes the maximum level of potable salinity and δ denotes the maximum allowable daily increase in potable salinity. Clearly, the objective of the problem is to *maximize* the sum

$$\sum_{i=1}^{n} (S_i^E \leq S_{\text{desire}}^E)$$

which represents the total number of days that potable salinity is below the desirable level S_{desire}^E.

5 Improving Search

We implement the above model using ILOG Solver 6.0 [5]. Out-of-the-box execution strategies used in our initial implementation fails to handle even small testing instances of the problem. There are two important issues in applying CP to solve problems. The first issue is to use an appropriate search strategy so that (good) solutions appear earlier in the search. There is no definite rule for discovering what is a good search strategy. By studying the problem structure and insights of human experts in depth, we are able to come up with a good search strategy. The second issue is that the model should also have *strong* propagation: that is, it should be able to quickly reduce the domains of the variables of the problem. We give a theorem for deriving useful implied constraints from a set of linear equalities to increase the amount of constraint propagation.

5.1 Variable and Value Ordering Heuristics

Since values in our model are defined on a day-by-day basis, it does make sense to label the variables chronologically by the days. We propose to pick first the seven decision variables for day 1, then day 2, and so on until day n. Such variable ordering has the advantages of turning many of the non-linear constraints into linear constraints, since constraint propagation on linear constraints is usually stronger than that on non-linear constraints.

Within day $i \in \{1, \ldots, n\}$, we propose to pick the variables based on the following order: (P_i^X, P_i^Y, P_i^Z, O_i^C, O_i^D, O_i^A, O_i^B). This ordering is the best we have so far after extensive experiments. The rationale is that the river is the only source of water, the pumps dictate the amount of salts to take into the reservoirs and are very important in controlling the salinity of potable water. In the raw water system, reservoirs A and B serve only as storage for surplus water which can be used to dilute the water pumps from the river, and hence are less important than reservoirs C and D.

Different variables represent different control parameters of the raw water system. Rather than using a single value ordering heuristic for all variables, we have different heuristics for different variables depending on their strategic roles in the raw water system.

- For variable P_i^X, the value ordering heuristic depends on the salinity of river S_i^X on day $i \in \{1, \ldots, n\}$. In order to control the salinity, it is common sense to pick lower value for P_i^X (i.e. pump less water) if S_i^X is high (i.e. salty river water); and pick higher value for P_i^X otherwise. We make use of a user-supplied salinity level *avoidPump* to indicate when the salinity should be considered high. If S_i^X is less than *avoidPump*, then larger values in the domain of S_i^X can be tried first; and vice versa, otherwise. In comparing S_i^X and *avoidPump*, the magnitude of their difference is taken into account too.
- For variables P_i^Y and P_i^Z, the value ordering heuristic picks the middle value first. Pumping stations Y and Z pump the water coming from pumping station X, and we prefer to pump more water from the river when it is less salty. We lean on pumping more water into Reservoirs A and B for dilution, but at the same time do not want to overdo it (since it is dangerous when the salinity of the water from pumping station X is high).
- For variable O_i^C, rather than choosing the values one-by-one from the domains, we use bisection to perform domain splitting. Bisection divides the values in a variable domain into two equal halves, and this process is repeated recursively forming a binary tree with leave nodes containing only a single value. The water supply company prefers to use more water in reservoir C for consumption. Therefore, our heuristic prefers to visit the branch with larger domain values first each time the domains are bisected.
- For variable O_i^D, the water supply company wants to avoid using too much water from reservoir D. If $S_{i-1}^C \leq S_{\text{desire}}^E$, we use bisection and visit first the branch with smaller domain values. Otherwise, our heuristic picks the value which gives the minimum amount of water required to satisfy $S_i^E \leq S_{\text{desire}}^E$.
- For variable O_i^A and O_i^B, we use bisection and visit first the branch with smaller values. The rationale is to keep more fresh water in reservoirs A and B for dilution.

For most of the test cases given by the water supply company, the above search strategy performs well. We called this strategy the NORMAL strategy. However, there are some stringent (unrealistic) test cases where the amount of daily water consumption is usually higher than the maximum amount of water that can flow out of reservoir C. If we are too frugal in supplying water from reservoirs A and B to C, reservoirs C and D alone would not be able to handle the high daily water consumption. To deal with such situation, we propose another set of value ordering heuristic, called the HIGH strategy, especially for test cases with such stringent daily water consumption pattern. The only modification is to visit first the branch with larger domain values when bisecting domains of variables O_i^A and O_i^B. The rationale is to keep reservoir C as full as possible.

5.2 Greedy Search Strategy

The basic solution search technology is branch-and-bound with constraint propagation. The *avoidPump* user input parameter turns out to have great impact on the quality of the solutions generated. Since our value ordering heuristics are designed to generate good quality solutions earlier in the search, prolonging the search effort could be fruitless. We adopt an opportunistic iterative improvement approach.

Our search strategy encompasses trying different *avoidPump* values in succession with a *timeout* (300 seconds) period for each value. After consultation with human operators and extensive experimentations, we adopt to try the following *avoidPump* values in sequence: $600, 700, \ldots, 1500$. A smaller (larger) *avoidPump* value implies a more conservative (aggressive) approach to pumping water. In other words, we progress from a more conservative to a more aggressive approach.

For every *avoidPump* value, we start execution with the best solution from the last execution as guidance. After a *timeout* period expires, the system examines if a better solution is found. If yes, execution continues for another *timeout* period; otherwise, the next *avoidPump* value is tried. The rationale is that if a better solution is found within the *timeout* period for a particular *avoidPump* value, the value is good and should be given more chance to search for even better solution. On the other hand, a *avoidPump* value failing to find any good solutions within the *timeout* period is probably no good and there is probably no point to search further.

5.3 Adding Implied Constraints

Most of the constraints in our model are linear equalities denoting the law of conservation of water and salts. Given a set of linear equalities sharing common terms, we can introduce a new variable to denote the common terms and reformulate the linear equalities in terms of the new variables. The resulting set of linear equalities can be added as implied constraints to increase the amount of constraint propagation. Modern constraint solvers use bounds propagation [6] for linear arithmetic constraints. We state without proof the following theorem based on the work of Harvey and Stuckey [4] and Choi *et al.* [2,3].

Theorem 1. *Let* $c_1 \equiv \sum_i (a_i)(x_i) + \sum_j (b_j)(y_j) = d_1$ *and* $c_2 \equiv \sum_j (b_j)(y_j) + \sum_k (c_k)(z_k) = d_2$, *we can reformulate* c_1 *and* c_2 *as* $c_3 \equiv \sum_j (b_j)(y_j) - v = 0$, $c_4 \equiv \sum_i (a_i)(x_i) + v = d_1$, *and* $c_5 \equiv v + \sum_k (c_k)(z_k) = d_2$. *Bounds propagation on* $\{c_1, c_2, c_3, c_4, c_5\}$ *is stronger than bounds propagation on* $\{c_1, c_2\}$.

For instance, observe that there is a common term $-(O_i^A \cdot \tau) + I_i^A$ between Equations 3 and 5, similarly a common term $-(O_i^B \cdot \tau) + I_i^B$ between Equations 4 and 5. We can reformulate Equations 3, 4 and 5 as follow

$$\begin{array}{ll} W_i^A = -(O_i^A \cdot \tau) + I_i^A & W_i^B = -(O_i^B \cdot \tau) + I_i^B \\ V_i^A = V_{i-1}^A + W_i^A & V_i^B = V_{i-1}^B + W_i^B \\ V_i^C = V_{i-1}^C - (O_i^C \cdot \tau) + I_i^X - W_i^A - W_i^B & \end{array}$$

where W_i^A and W_i^B are auxiliary variables representing the common term. We can add the above equalities as implied constraints to our model. Suppose $\tau = 5000$ and the domain D is such that: $D(V_0^A) = \{1300000\}$, $D(V_0^B) = \{1237350\}$, $D(V_0^C) = \{2450000\}$, $D(V_1^A) = \{320000, \ldots, 1500000\}$, $D(V_1^B) = \{100000, \ldots, 1260000\}$, $D(V_1^C) = \{1200000, \ldots, 2450000\}$, $D(O_1^A) = \{0, \ldots, 15\}$, $D(O_1^B) = \{0, \ldots, 15\}$, $D(O_1^C) = \{27, \ldots, 42\}$, $D(I_1^A) = \{0, \ldots, 36000\}$, $D(I_1^B) = \{0, \ldots, 36000\}$, and

$D(I_1^X) = \{0, \ldots, 432000\}$. Constraint propagation with the original set of constraints returns the domains D' such that $D'(I_1^X) = \{0, \ldots, 282000\}$, while constraint propagation with the new and enlarged set of constraints returns the domains D'' such that $D''(I_1^X) = \{0, \ldots, 268650\}$. The latter is stronger in propagation.

6 Experiments

We have tested the system using both real-life and handcrafted data provided by the water supply company. We have chosen three representative sets of data to illustrate the performance of our system. Each set of data has a different characteristic, aiming to test the versatility and robustness of our engine. The three sets of data differ in terms of:

- the duration of the salinity period (n),
- the predicted salinity level of river (S_i^X),
- the daily water consumption of the city (V_i^E),
- the volume thresholds for the reservoirs (H_i^A, H_i^B,H_i^C and H_i^D), and
- the initial volumes and salinity values of the reservoirs (V_0^A, S_0^A, V_0^B, S_0^B, V_0^C, S_0^C, V_0^D, S_0^D).

Figure 2 gives the salinity curves of the prediction data.

The following experiments are executed using a Linux Workstation (Intel Pentium-III 1GHz with 1GB memory) running Fedora Core release 3. We choose ILOG Solver 6.0 [5] as our implementation platform. The time limit for the system to run is set to one hour. Execution is aborted when the time limit is reached, and the best solution located so far is reported.

For this project, there is no way to do comparison with the existing manual method based on spreadsheet. We cannot make any meaningful comparison in terms of the quality of solution since the manual method often fails to obtain a solution satisfying all constraints. We also cannot make any fair comparison in terms of time since one is a manual method and the other is an automated method. Therefore, we present only the results obtained from our system.

Table 2 shows the result of Set 1. The first two columns with heading “salinity” indicate the different combination of desirable and maximum salinity level. The next

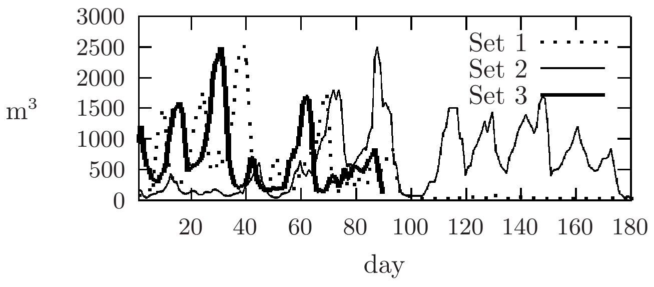

Fig. 2. Salinity Prediction Curves for Data Sets 1 to 3

Table 2. Result of Set 1, Duration = 180 days

salinity		normal			high			old		
desire	max	days	secs	fails	days	secs	fails	days	secs	fails
200	300	157	21	3	126	619	134,520,603	104	44	2,367
250	350	180	21	3	168	621	134,520,602	162	330	30,005
250	400	180	23	3	168	623	1,807	162	333	30,013
250	500	180	29	3	168	627	134,520,602	162	337	30,013
250	600	180	34	3	168	631	1,807	162	342	30,013
250	1,000	180	58	3	168	649	134,520,602	162	364	30,013
300	600	180	35	3	180	32	4	180	215	20,678
300	1,000	180	62	3	180	50	4	180	238	20,678

Table 3. Result of Set 2, Duration = 180 days

salinity		normal			high			old		
desire	max	days	secs	fails	days	secs	fails	days	secs	fails
200	300	–	–	–	–	–	–	–	–	–
250	350	–	–	–	–	–	–	–	–	–
250	400	–	–	–	–	–	–	–	–	–
250	500	107	1,222	269,201,686	107	1,222	134,725,593	–	–	–
250	600	117	926	269,080,063	117	925	134,604,570	–	–	–
250	1,000	117	943	269,063,234	117	943	134,582,923	73	53	520
300	600	137	952	269,096,832	129	1,225	269,196,833	–	–	–
300	1,000	146	647	269,041,205	146	646	78,408	114	52	605

three columns with heading “normal” indicate the results using the NORMAL strategy. We measure the number of days for which the potable salinity is below the desirable level (column “days”), the runtime in seconds (column “secs”) and the total number of fails (column “fails”). The next three columns with heading “high” indicate the results using the HIGH strategy. The last three columns with heading “old” indicate the results of an earlier implementation without the custom heuristics and implied constraints listed in Section 5. Our system performs very well for Set 1 and is able to fulfill all 180 days with the potable salinity below 250 ppm in just 21 seconds. For this scenario, the NORMAL strategy clearly works better than the HIGH strategy. This scenario represents a typical dry season of the city that lasts only 90 days out of the 180 day period. The search is clearly improved comparing to the “old” implementation for the NORMAL strategy is able to find better solution much faster and lesser number of fails.

Table 3 shows the result of Set 2, which is a more difficult scenario than Set 1. Set 2 has a prolonged drought period lasting the entire 180 days, which is *one of the worst in the last 150 years for the city*. For this set of data, our system can maintain the potable salinity always below 500 ppm, but it can only fulfill 107 days out of 180 days with the potable salinity below 250 ppm. It takes around 20 minutes for our system to find this solution. If we can relax the desirable salinity level to 300 ppm and the maximum salinity level to 1000 ppm, our system can return a better solution fulfilling 146 days

Table 4. Result of Set 3, Duration = 90 days

salinity		normal			high			old		
desire	max	days	secs	fails	days	secs	fails	days	secs	fails
200	300	–	–	–	–	–	–	–	–	–
250	350	–	–	–	–	–	–	–	–	–
250	400	–	–	–	21	2,403	404,174,730	–	–	–
250	500	–	–	–	28	1,803	269,394,622	12	4	9
250	600	–	–	–	28	1,803	269,480,570	12	4	9
250	1,000	–	–	–	28	1,805	639,731	12	6	9
300	600	–	–	–	45	1,803	269,441,669	24	4	5
300	1,000	–	–	–	45	1,805	135,072,897	24	6	5

out of 180 days with the potable salinity below 300 ppm. The system is now able to find the solution in 10 minutes. This example illustrates the flexibility of our system. If we allow the desirable salinity level to raise slightly higher, our system would be able to distribute the salinity level of potable water more evenly among the days to improve the quality of solution.

Table 4 shows the result of Set 3, which is an artificially handcrafted scenario. The salinity level of the river for Set 3 is similar to the first 90 days of Set 1. The difficulty of Set 3 lies in the unrealistically high daily water consumption[2] comparing to Set 1 and Set 2 as shown in Figure 3. Set 1 (the bold dotted line) has constant daily water

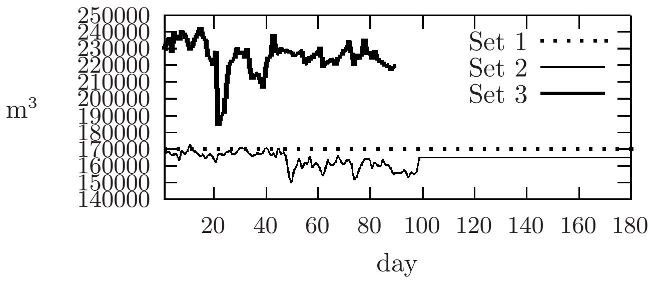

Fig. 3. Daily Water Consumption (V_i^E)

consumption. Set 2 (the thin line) has a fluctuating daily water consumption. Set 3 (the bold line) has a more fluctuating daily water consumption that is usually higher than the maximum amount of water that can flow out of reservoir C (i.e. 211,395 m^3/day). Set 3 suffers from the problem discussed at the end of Section 5.1, which makes the NORMAL strategy fail. The HIGH strategy is able to find a solution fulfilling 28 days out of 90 days with the potable salinity below 250 ppm and maintaining the potable salinity always below 500 ppm. It takes about 30 minutes to find this solution.

[2] The capacities of the reservoirs are: $A = 1,500,000$ m^3, $B = 1,260,000$ m^3, $C = 2,450,000$ m^3, and $D = 2,060,000$ m^3 for comparison with the daily consumption.

7 Discussions

In this project, we collaborate with the International Institute for Software Technology, United Nations University (UNU/IIST). We are responsible for the design and implementation of the core optimization engine, while UNU/IIST is responsible for constructing a web user interface to invoke our optimization engine. We discuss in the following issues regarding system development and deployment.

7.1 Added Values of CP

Our client gave us the problem in late September, 2004, only a couple of months before the beginning of the winter dry season. During that time, the city was suffering from one of the most serious drought in the last 150 years. Due to the urgency of the problem, we were given only 2 weeks to come up with a functional prototype, and to release a fully functional production system in early December, 2004, just before the winter dry season began. This version replicates and automates the functionalities and model of the client's spreadsheet model described in Section 2. We came up with a version to model the table constraint (water flow limit from Reservoir C to D) plus a large number of change requests in another month's time. The project involves the authors coming up with the model and techniques for improving search, and two undergraduate students for the implementation effort. The use of CP allowed us to meet the deadline and come up with the first fully functional production system in just 2 months of development. We spent another 5 months to study and experiment with various search improvements.

We have delivered the system to the water supply company. Installation and user trainings were provided, together with a 12-month maintenance and support period. The system has passed user acceptance test and has been into full production mode since June, 2005. We received positive feedbacks from the users of the water supply company. The only support request was just for a re-installation because of the ILOG product upgrade. However, due to the continuing worsening of the drought condition in the past years, even optimizing the logistical operations of the raw water system alone is insufficient to control the salinity problem. The water supply company is seriously considering physical measures such as reverse osmosis, moving the pumping station to upper stream of the river, and even purchasing fresh water from nearby provinces to effectively handle the salinity crisis.

The optimization engine is abstracted from the web interface, the end-users do not need to understand CP at all. Indeed, our client does not care about the optimization methodology we use, and wanted only a practical solution for the salinity problem that could be developed in 2 months, although our method has no guarantee for optimality.

7.2 Reasons for Choosing Finite Domain

Although the domain of the salinity problem is continuous (real numbers), a more natural choice seems to be modeling the problem using interval constraints instead of finite domain constraints. However, we still decided to use finite domain constraints for the following practical considerations. First, as discussed in Section 4.1, it does not make sense to operate the pumps for a very short time (e.g. 3 minutes) or to transfer a very small amount of water (e.g. 10 m^3). Therefore, we decided to discretize the domains. Second, finite domain constraints have had many successful industrial applications including scheduling, time-tabling, resource allocation, etc. Third, the development schedule was extremely tight and opportunity cost was high. At the time we were given the problem, we simply could not afford a lot of experimentation but had to adopt a proven technology.

7.3 Other Optimization Methodologies

Besides CP, we have investigated with UNU/IIST in applying Evolver [7], which is a genetic algorithm based optimization engine for Microsoft® Excel, to the project. Experimental results show that this approach is less efficient both in terms of execution time and quality of solution. Moreover, Evolver is only semi-automatic, requiring expert human guidance during the search for solutions. This approach is also unstable and unpredictable with regard to convergence. Nevertheless, such an approach is good for fast prototyping.

We also works with the operations research (OR) colleagues in our university to investigate the use of linear programming (LP) [8] for solving the salinity problem. The advantage of using LP is that the domain of the salinity problem is continuous in nature (i.e. real numbers); hence, there is no need to discretize the domains. However, the major obstacle to the LP approach is the *nonlinear* constraints in the problem, i.e. constraints on the law of conservation of salts and the table constraints on the water flowing out of reservoir C. The idea is to construct an *approximate* model of the problem with only linear constraints and objectives. Preliminary results are encouraging, outperforming our engines in selected test cases. The possibility of combining the LP model and the CP model is a promising research direction.

8 Conclusion

By applying CP, we have developed a fully automated optimization engine incorporating a more realistic model for solving the salinity problem. Experimental results demonstrate that the engine is more efficient and can produce higher quality solutions than the human counterpart. Now, even a non-domain expert can make use of our engine to plan for water management operations and experiment with different salinity scenarios in advance.

In summary, the choice of CP has immense impact on the successful delivery of the project. First, the rich constraint language available in commercial constraint solvers allows efficient modeling of the problem. Separation of concerns of CP allows us to focus on programming search heuristics. We were thus able to complete a working prototype and a functional production system within a tight development schedule. Second, CP is flexible and adaptive to changes. During the course of development, our client requested for numerous, often unreasonable, changes to the requirement specification. Without CP as the core technology, we were not sure if we could deal with all the requests in a timely and mostly effortless manner. Third, we were also able to adopt and generalize latest research result in propagation redundancy [4,2,3] to come up with useful implied constraints for our implementation, thus enhancing the constraint propagation in the salinity control engine. And this work on the model is orthogonal to the search strategies we employ. This is again a triumph of separate of concerns.

References

1. Brdys, M., Creemers, T., Riera, J., Goossens, H., Heinsbroek, A.: Clockwise: Constraint logic for operational control of water systems. In: The 26th Annual Water Resources Planning and Management Conference, pp. 1–13 (1999)
2. Choi, C.W., Harvey, W., Lee, J.H.M., Stuckey, P.J.: Finite domain bounds consistency revisited. In: Australian Conference on Artificial Intelligence, pp. 49–58 (2006)
3. Choi, C.W., Lee, J.H.M., Stuckey, P.J.: Removing propagation redundant constraints in redundant modeling. ACM Transactions on Computational Logic (to appear 2007)
4. Harvey, W., Stuckey, P.J.: Improving linear constraint propagation by changing constraint representation. Constraints 8(2), 173–207 (2003)
5. ILOG, S.A.: ILOG Solver 6.0: User's Manual (2003)
6. Marriott, K., Stuckey, P.J.: Programming with Constraints: an Introduction. MIT Press, Cambridge (1998)
7. Palisade Corporation: Evolver 4.0 (2005), Available from `http://www.palisade.com`
8. Vanderbei, R.J.: Linear Programming—Foundations and Extensions, 2nd edn. Springer, Heidelberg (2001)