Independent Constraint Satisfaction and its Application to Sewerage System Control

Kokolo IKEDA
Kyoto University
Yoshidanihonmatsu-cho
Sakyo-ku, Kyoto-shi, Japan
kokolo@media.kyoto-u.ac.jp

Kei AOKI
Tokyo Institute of Technology
4259, Nagatsuta-cho,
Midori-ku, Yokohama, Japan
aoki@fe.dis.titech.ac.jp

Akihiro NAGAIWA
Toshiba Corporation
1, Toshiba-cho,
Fuchu-shi, Tokyo, Japan
akihiro.nagaiwa@toshiba.co.jp

Sigenobu KOBAYASHI
Tokyo Institute of Technology
4259, Nagatsuta-cho,
Midori-ku, Yokohama, Japan
kobayasi@dis.titech.ac.jp

Abstract-

Most of real world problems contain complex and various constraints, and this goes for the sewerage system control problem, our target. For handling them, the penalty depending on the degree of violation is often used. However, additive penalty method (APM) often leads the fatal compromise to a local optimum, in this paper we introduce independent constraint satisfaction (ICS). Another difficulty of this problem is the inaccuracy of inflow forecasting, we show the eager re-scheduling is superior to the lazy one.

1 Introduction

It is frequently employed for decision making to consider the future aspect and arrange a schedule to decide the action. As "scheduling" is a classic optimization problem and numerous approaches has been developed, evolutionary computation including Genetic Algorithms (GAs) is one of the most promising way [Yamada 96].

In this paper, we challenge the sewerage system control problem. Briefly, the purpose is to minimize the fluctuation of water flowing into the sewerage treatment plant. For this goal, future inflows into each hookup ponds are estimated, and future (f.g. half a day' s) pump discharges of every ponds are scheduled carefully.

One of the most important characteristics of this problem is that the estimation of inflows is not enough accurate though scheduling is based on the estimation. Against this difficulty, a great deal of effort has been made to insure accurate estimation. However, as the estimation of sewerage inflows is much harder than of water supply demands, in many cases front-line operators of each ponds must regulate their control to prevent ponds from overflowing. On the other hand, the approach to re-schedule frequently in order to adjust the schedule to the current condition, is also possible. For this approach, the process of estimating and scheduling must be done enough fast.

Generally scheduling problems contain multiple and complex constraints. In our problem, the water level of a pond is forbidden to run over a certain range. For satisfying these constraints, in a case a problem-dependent heuristic logic is used, and in another case a penalty function related to the degree of violation is introduced. In this paper, we discuss the defect of additive penalty method and propose a solution to handle constraints adequately. Eminent performance of the proposal methods are shown on experiments using a simulator and certain actual data.

2 Constraint Handling by Independent Constraint Satisfaction

It is very important issue how to handle constraints when we solve some sort of optimization problems. Especially, real world problems contain multiple and complex constraints in many cases. At first, frequently used methods for constraint handling are grouped as follows. From here, without loss of generality, the discussion is specified to **minimization** problems for simplicity.

1. **Discard**
   Drop invalid solutions. This is easily mounted, but is limited to the case that valid (with no violation) solution is easily gained.

2. **Repairing Operation**
   Modify the violated solution to valid one using a heuristic procedure [Giffler 60]. This procedure likely sacrifices the primary quality of the solution to the purpose to modify it to valid one. Furthermore, the more constraints are imposed, the more difficult to compose the procedure.

3. **Penalty**
   Measure the degree of violation as the penalty, and minimize it gradually in order to make the solution satisfy constraints. In most cases, the penalty is used additively to the primary objective function (Additive Penalty Method, APM: [Francisco 2000]), two optimizations to minimize the primary objective function and to minimize the penalty are done at once.

4. **Multi-Objective Optimization**
   Consider the problem as 2-objective optimization, to minimize the primary and the penalty. Then algorithms developed in the MOOP field can be employed [Deb 2000].

In this section we discuss the defect of APM, and present a solution to using penalty adequately.

Consider a simple graph coloring problem, to color all nodes s.t. both ends of each edge are differently colored. It is the number of used colors $c(x)$ to be minimized, and the penalty $p(x)$ can be introduced by counting the number of edges whose ends have a same color. The objective function is calculated as $f(x) = c(x) + 2p(x)$. It is very easy to reduce the penalty by using new color on a violating node, but to iterate this procedure results in the facile solution.

Now let S be the whole search space, and $S_n \in S$ be the subspace (whose graph is) using n colors. Assume just 4 colors are required for a graph. Though S_4 contains the optimal solution, almost all solutions of S_4 violate constraints seriously; so the average of $f(x), x \in S_4$ may be bigger than of S_5 (Fig. 1). In such cases, i.e. the subspace which includes the optimal is averagely worse, GAs often fail to find the optimal and prematurely converge to a local optimal which is included in an averagely better subspace [Ikeda 2000].

The factor responsible for this phenomenon is to use penalty additively and to search whole space at once, whence optimizers search hopeful subspace intensively. For the graph coloring problem, a technique for this difficulty was introduced, i.e. to search each subspaces independently. One trial is only for S_3 in which the number of colors is never changed and only the penalty is minimized, and another is only for S_4, or S_5. By this separation from other subspaces using different number of colors, the premature convergence to an easygoing solution is prevented.

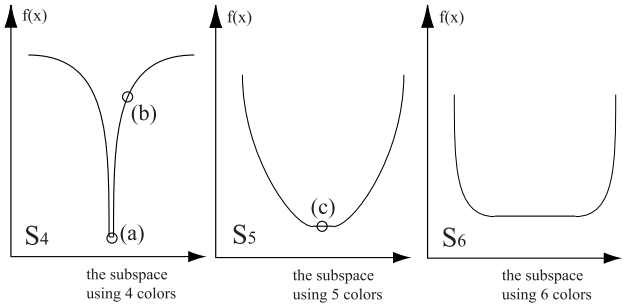

Figure 1: A part of the fitness landscape of a graph coloring problem, separated to subspaces according to its number of colors being used. S_n is the subspace of solution with n colors: S_4 includes the optimum (a), but almost all solutions are awfully violating and have bad value like (b). On the other hand, although S_5 and S_6 have only the local optima such as (c), the average fitness is better than S_4. In such case, called **UV-structure**, the intensive search is done not on S_4(V-valley) but on S_5 or S_6(U-valley), therefore often converge to a local optima like (c). Such undesirable behavior is called as **UV-phenomenon** [Ikeda 2000].

This technique can be generalized for extensive problems with constraints. As the whole space is separated to some subspaces and valid solutions are searched independently, we call this Independent Constraint Satisfaction, ICS. The framework of ICS is as follows.

Independent Constraint Satisfaction

1. Let S be the search space, $x \in S$ be a solution, $o(x) \in R$ be the primary objective function to be minimized, and $p(x) \in R$ be the penalty.

2. Separate S to some subspaces $S_1, S_2, ..., S_N$, s.t. $b_i \leq o(x) \leq b_{i+1}$ for all $x \in S_i$. That is to say, all members of a subspace hold similar values. S_1 is the most desirable but most difficult to find a valid solution, and S_N is the opposite.

3. **From** each S_i, minimize $o(x)+p(x)$ for a certain period, under the condition $o(x) \leq b_{i+1}$. The condition $b_i \leq o(x)$ is not necessary because the purpose of ICS is to guarantee the upper bound of $o(x)$.

4. Select the best solution from all valid ones.

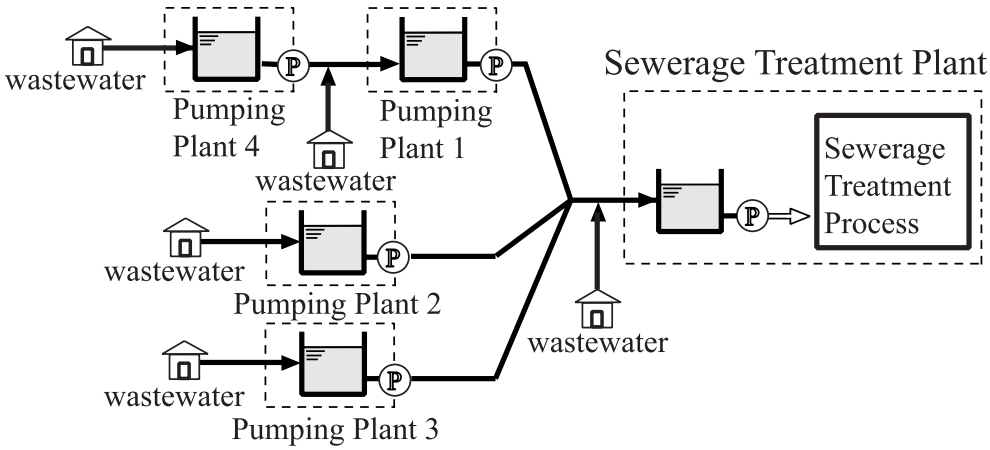

Figure 2: A sewerage system formed by 5 ponds; circled 'P' shows a pump.

For the separation of S at step 2. and for the restriction of $o(x)$ at step 3, though generally they can be executed by the generate-and-select method, a simple device would help the computing efficiency if it can be introduced. As the generate-and-select method uses only the primary value $o(x)$ and solutions are compared by using only the penalty function $p(x)$, even if a large number of variables and many constraints were given, a search with ICS would not be technically harder.

This ICS strategy is essentially the same to the ϵ-constraint method (we call ECM here) for multi-objective optimization [Bhaskar 2000]. ECM selects one objective f_i, sets upper-bound ϵ_j for other objective $f_{j\neq i}$, and minimize f_i as a single-objective optimization under the condition $f_{j\neq i} \leq \epsilon_j$.

By considering the primary objective function $o(x)$ and the penalty $p(x)$ as 2-objectives to be minimized, ECM can be used for this 2-objective optimization, $p(x)$ is minimized under the condition $o(x) \leq \epsilon$. This is one trial of ICS on a subspace $o(x) \leq \epsilon$. So, as with the ECM, it is left to the solver how to fix N, separate S, i.e. how to fix $b_1 ... b_{N+1}$, and in which order subspaces are optimized.

Another form of ICS is as follows. At first an upper bound UB is fixed, and one search is done under the condition $o(x) \leq UB$. Next UB is decreased if all constraints are satisfied (or is increased if not), and a new population is optimized again. By repeating such trials, UB is controlled to an adequate value. We call this form 'adaptive ICS'.

ICS requires multiple independent trials, some of users may consider it is not efficient. However, the clash between the minimization of the primary objective function and the satisfaction of constraints is frequently serious. We insist that, ICS will greatly help to tackle real-world problems with multiple constraints.

3 Sewerage System Control

In modern life sewerage system is one of the most important lifelines, being absolutely necessary especially in urban area. Recently, the automatic/support control of operation such as the pumping control, is considered and partly introduced on grounds of the wide spread of sewerage service and the need for cost saving. However, most sewerage plants are controlled by experts in fact, more effective automatic controller using latest optimizers is desired. Our approach is also available in the field of the water supply control.

3.1 Overview

A sewerage system consists of one final treatment plant and several hookup pumping plants, and they are connected in one direction (Fig. 2). Each plant equips a storing pond and some pumps, and how much water it pumps up is decided in every time-step. The sewerage system control in the format of optimization problem is organized as follows:

- **The primary purpose: stable sewerage treatment**
  The precipitation process and some biochemistry processes are necessary for purifying sewerage water in the final treatment plant. As these processes are long haul, it costs much to change the plenty of medicine or organic matter. Therefore, the primary purpose is to minimize the fluctuation of water processed in the final plant, and the fluctuation is measured by counting the number of pumping switches.

- **Constraints: bounds of water level**
  The water level of each pond **must** be bounded in a range not to be damaged (so, as it were, really

the primal purpose is to accept all sewerage waters safely). Of course, the bigger the pond is, the easier the smooth control is. However, in reality, by the locational consideration and maintenance cost, hard restriction is imposed.

- **State Input to the decision maker**
  We assume that scattered plants are centrally controlled by a point, and there is no time lag about the water level information. The decision maker designs the schedule of operations with the use of the current time, current water level, current pumping volume, and additionally the estimation of future inflows into each hookup ponds. It is impossible to estimate them exactly, because every house drain water out arbitrarily, but it is possible to catch the rough pattern of them (Fig. 3). As the sewerage system we deal in this paper is split-flow type, no rainwater enters to sewerage drains.

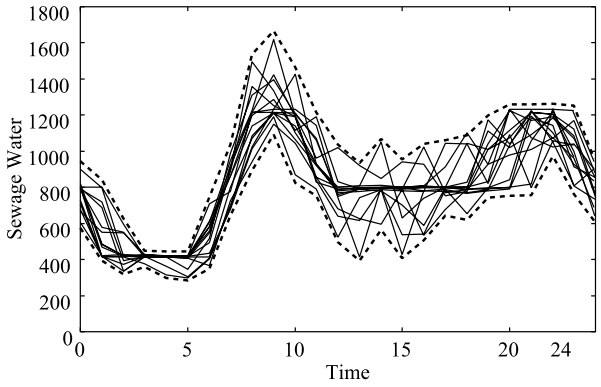

Figure 3: Time series data of the sewerage inflows from houses to the hookup plants-4 ; of 14days ; each solid line is of each day, two broken lines are of the maximum and the minimum volume in each hour.

- **Action Output from the decision maker**
  A plant has several pumps, and controls the water pumping volume by the discrete combination of their activation. At every time-step, the decision maker must decide the pumping volume of every plant. Or, it is required in some cases to decide not only pumping volumes but also the combination of their activation in consideration of their driving costs.

- **Secondary purposes**
  Real-life problems generally involve many secondary purposes, also the sewerage control problem does. For example, to save the electric cost by taking advantage of night rates, or to minimize the fluctuation of water processed not only in the final plant but also in hookup ponds. In this paper, we regard them only if they have no adverse affect on the primary value and the penalty.

3.2 Conventional methods

Up to now, for the sewerage (or water supply) system control problem, based on taking them as an integer programming problem, linear programming (LP), dynamic programming (DP), branch and bound method (BAB) and so on have been researched. However, as the integer programming problem is so-called NP-hard, there is the danger that these methods can not solve in a practical period when challenging against large scale problems. Therefore several ways, multistage LP with some heuristic ([Kurisu 1994]) or planning by stochastic optimization, are also employed in order to gain satisfying solutions fast.

Another, distributed reinforcement learning is attended as the method to control connected plants under the ill-informed situation; it was applied to sewerage control also, and learned enough favorable policy to be used actually [Aoki 2003]. By using past sewerage inflows as the learning data, the learner gains its policy in advance. When the coming data has similar pattern to the learning data, the learner can make the decision immediately. On the other hand, when the circumstance changed by 'the peculiar day' or the failure of pumps, the learner must learn again using such data or breakdown.

3.3 Control by Genetic Algorithms

A genetic algorithm had been already applied to the water supply problem [Sakamoto 2000]. An individual represents the pumping schedule of all plants, of a day. To calculate the fitness value of a individual, a computational simulation is run using the estimation of water supply demands and the pumping schedule. In [Sakamoto 2000], invalid solutions violating water level constraints are repaired by a problem-dependent backtrack logic, or simply discarded. Though the repairing operation contributes to finding valid solutions, it conflicts with the primary purpose, control stability, because it changes the pumping volume with short sight, at the time the violation is predicted.

For applying this approach to the sewerage control, in addition to the difficulty of constraint handling, the deterioration of the estimation becomes a serious problem. Though very accurate estimation is being done at the field of water supply because the supply deficiency must be avoided at any price, the accuracy of sewerage water estimation is comparatively low. Practically

even if the schedule of a day is made by GAs, as the estimation error is accumulated at latter of the day, the system never work well unless experts of each plants correct the schedule as the occasion may demand.

3.4 Proposal approach

In this paper, thinking about the aspect of the sewerage system control and the defect of current GAs, we propose an approach for better control. Main two ideas are follows:

- **Re-scheduling at every time-step**
  In the conventional planning method, a schedule is of a whole day, made at the early morning, and plants follow the schedule all the day unless it breaks down. However, as already mentioned, the estimation of sewerage inflows is not enough accurate, then makeshift controls are frequently done on account of the mismatch between the estimated water level and actual one. In our approach, the schedule is reconsidered every time-step on the ground of the latest water level and the latest estimation of inflows. That is to say, a schedule of a long period (in this paper 12 hours) is used only for the next time-step (say 20 minutes), not for the future use. Herewith, the decision making based on the latest information and the adequate control even under the rough estimation become possible.

- **Separation of the stable control and the violation rejection**
  Using the repairing operation or APM for the constraint satisfaction is considered as the reason of producing a lot of easygoing solutions only satisfy the constraints but are flopping. In our approach ICS is employed; the number of pumping switches, the primary objective, is constant on one GA and only the penalty is minimized. After several GA with variant numbers of pumping switches, the best schedule is selected. In this procedure, no operation for constraint satisfaction sacrifice the primary objective, it is expected that better schedules are gained.

  In association with the graph coloring problem stated at Section 2, the number of pumping is as the number of used colors. Although they are to be minimized, the smaller they are, the more likely constraints are violated. Therefore, we believe, constraints should be satisfied independently in each subspace which has same number of used colors, or pumping.

4 Experiments

4.1 Formulation and Experimental settings

In this paper, the model of a sewerage system consisting of 5 plants is used [Aoki 2003]. Each plants P_i equips a storing pond whose area of base $B^i[m^2]$, the highest water level $h^i_{MAX}[m]$, the lowest $h^i_{MIN}[m]$, and n^i-pumps whose pumping volume $P^i_{1..n^i}[m^3/h]$. They are connected as Figure 2, and these actual values are shown in Table 1.

The behavior of our simulator employed is as follows. This simulator is used both cases, (1) when the GA evaluate the fitness of an individual, and (2) when performances of candidate algorithms are measured and compared each other. The computation of the simulator is done once each time-step $T_{step}[min]$. When $T_{step} \geq 5$, the simulator behaves enough analogously like as a real sewerage system.

Simulator behavior

1. Initialize the time, water levels, pumping states.
2. Observe the state input of time-step t : time T_t, water levels $h^i(t)$, last pumping volumes $a^i(t-1)$.
3. Calculate practical/estimated inflows of one time-step $o^i(t)$ led from T_t.
4. (Pumping volumes $a^i(t)$ are decided in a certain manner outsize.)
5. Update water levels: let $f^i(t)$ be the inflow from upstream plants,
   $h^i(t+1) := h^i(t) + \frac{o^i(t)+f^i(t)-a^i(t)}{B^i}$.
6. If $t < t_{MAX}$, the number of steps the simulation runs, update the time-step to $t+1$ and continue from 2.
7. Return two values, the number of switches of pumping volumes of the final treatment plant and the degree of water level violation.

4.2 Overview of compared algorithms

All algorithms compared in this paper plan a schedule up to H_{SC} hours by considering current water levels, pumping states, and estimated sewerage inflows. The schedule is optimized by GA so that the number of switching is minimized and no violation of water level is predicted to the extent of estimation.

GA^{lazy}_{APM} makes a schedule once by using GA with

Table 1: Constant values of the target sewerage system

no.	plant	flows into	base $B_i[m^2]$	water level $h^i_{MAX} \sim h^i_{MIN}[m]$	pumping volume $P^i[m^3/h]\times$ equips
P_1	final plant	-	2000	$0.8 \sim -2.5$	$980 \times 2, 2000 \times 2$
P_2	hookup plant 1	final plant	1000	$0.8 \sim -2.5$	$500 \times 2, 1000 \times 2$
P_3	hookup plant 2	final plant	1000	$0.8 \sim -2.5$	$300 \times 2, 500 \times 2$
P_4	hookup plant 3	final plant	300	$0.8 \sim -2.5$	$150 \times 2, 400 \times 2$
P_5	hookup plant 4	hookup plant 1	600	$0.8 \sim -2.5$	$300 \times 2, 500 \times 2$

APM, and follows it over steps until H_{SC} hours pass or a future overflow is predicted by the simulator. This algorithm is fairly analogous to the actual operation.

GA_{APM} makes a schedule by using GA with APM also, but follows only the first step of it. Next step this algorithm re-schedules based on the latest state inputs.

GA^{lazy}_{ICS} makes a schedule once by using GA with ICS, and follows it over steps as GA^{lazy}_{APM}. The GA with ICS at first tries to find the valid schedule in the subspace that the number of switches of the final plant equals to zero. When no valid schedule is found, next it searches in the subspace of one switch, and next twice until the valid schedule is found or it runs past the time limit.

GA_{ICS} makes a schedule by using GA with ICS, and follows only the first step of it. This is our final proposal.

When no valid schedule is found within the time limit, the schedule with the minimum penalty is adopted resignedly. By comparing their performance, we insist that our proposal (1) re-scheduling at every time-step and (2) separation of the stable control and the violation rejection, are actually helpful.

4.3 Operators and Parameters

Configurations of the GA employed are as follows.

- **(Gene Coding)** One individual S represents a schedule up to H_{SC} hours, i.e. S=$[s_1, ..., s_{H_{SC}}]$ where $s_t = (a^1_t, ..., a^5_t)$ is pumping volume of plants after $t-1$ hours to t hours. It is a point to notice that, in the phase of planning, pumps are scheduled an hourly, independent on the time-step scale T_{step}. Hence, the search space of this problem is $5H_{SC}-$dimensional.

- **(Objective Function)** The primary objective function of an individual S to be minimized is, $o(S)$=$\sum_{t=1}^{H_{SC}} \{1 - \delta(a^1_t - a^1_{t-1})\}$, i.e. the number of switches of pumping volume at the final plant P_1. In addition, when $\mid a^1_t - a^1_{t-1} \mid \geq 1500\ [m^3/h]$, the switch is counted twice, $o(S)+ = 2$.

- **(Constraint Violation)** For each individual S, one simulation on the schedule is done up to H_{SC} hours. When a water level violation is observed on P_i at the time after t hours, 2^t is added to the penalty $p(S)$.

- **(Secondary Objective)** When two individuals are equivalent in the sense of penalty p and primary objective o, the winner is decided by comparing these control latitude. By writing the expected water level of P_i after t hours as h^i_t, and by defining the local latitude m^i_t be $\min(h^i_{MAX} - h^i_t, h^i_t - h^i_{MIN})$, we calculate the latitude of a schedule S as $m(S) = \min_{i,t}(m^i_t)$. The bigger $m(S)$ is , the easier the adaptation against the distortion of the estimation.

- **(Crossover Operator)** Let one parent S^* be $[s^*_1, ..., s^*_{H_{SC}}]$, and another parent S^+ be $[s^+_1, ..., s^+_{H_{SC}}]$, the crossover operator employed is the 2-point crossover for this representation. That is to say, at first select $1 \leq t_1 \leq t_2 \leq H_{SC}$ randomly, then the child S^c is produced as $[s^*_1, ..., s^*_{t_1-1}, s^+_{t_1}, ..., s^+_{t_2}, s^*_{t_2+1}, ..., s^*_{H_{SC}}]$.

- **(Mutation Operator)** At first select $1 \leq t' \leq H_{SC}$ randomly, then for all a^i_t under the condition $t' < t$, change the state of pumps, working to rest or opposite, with the probability of p_{mute}.

- **(Alternation)** Let the population size be N_{pop}. For the pair of parents S^1, S^2, at first N_{child} children are produced by N_{child} crossovers, and next the mutation operation is applied to all children. Finally, the best one among S^1 and children is selected, the best one is introduced instead of S^1.

4.4 Margin as a safeguard

Our GA optimizes its population not to violate the water level restriction through the utilization of the estimation of inflows. However, saying the same thing again and again, the estimation is not enough accurate. So, if the schedule is made in the full-range of the

limit water level, the actual water level often violates its constraints. There, as a safeguard, $r_{mgn}\%$ margin of the water level is imposed. That is to say, GA runs not under the actual limit h^i_{MIN} to h^i_{MAX} but under the narrowed limit $h^i_{MIN}+h_r$ to $h^i_{MAX}-h_r$, where h_r=$(h^i_{MAX}-h^i_{MIN})\times\frac{r_{mgn}}{100}$.

The more serious the violation damages and the less accurate the estimation is, the larger margin r_{mgn} is required. The larger margin is employed, the harder constraint is imposed upon the control, and the more difficult the stable control is.

4.5 Re-scheduling Frequency

If the decision maker follows a schedule longer after the schedule is made, the mismatch between the estimated water level and actual one is accumulated. On the other hand if the schedule is frequently revised, the mismatch is minimized as possible, then smaller margin can be employed with remote risk. That is to say, as GA_{APM} and GA_{ICS} re-schedule every T_{step} minutes, the shorter the time-step is, the better control is expected. Comparative experiments about the effect of the margin level and the re-scheduling frequency are shown at subsection 4.7.

4.6 Experiments on 4 algorithms

In this subsection, 4 algorithms GA^{lazy}_{APM}, GA_{APM}, GA^{lazy}_{ICS}, GA_{ICS} are compared under two conditions, (A) if the estimation were perfect, and (B) when the estimation is done by averaging past inflows at the time instant. We set parameters as, the scheduling range $H_{SC}=12[h]$, re-scheduling frequency $T_{step}=20[min]$, mutation rate p_{mute}=0.03, population size $N_{pop}=50$, the number of children $N_{child}=2$, and the margin r_{mgn} is, 0% at (A) because of no estimation mismatch, 4% at (B). One scheduling is up to 3000, 10000, 30000 evaluations; for these optimizations, about 1, 3, 10 seconds are required respectively using a 1GHz CPU.

Actual inflows into plants of 14 days are used for experiments, and the performance of an algorithm is measured by the number of switches of the final plant per a day, and by the number of violations per a day. 20 trials are done for each pair of algorithms and case (A) and (B), they are averaged and summarized in Table 2.

By these results, we can conclude as follows:

- By introducing the margin as safeguards, the overflow is controlled within the acceptable level. The effect according to the margin size is described at the next subsection.

Table 2: The comparison of 4 algorithms in 2 cases: The averaged number of switches and violations per a day

algotirhm	cpu time	(A)no mismatch		(B)rough estimation	
	(sec)	switches	overflows	switches	overflows
GA^{lazy}_{APM}	1	3.08	-	3.85	0.17
	3	3.24	-	4.25	0.085
	10	2.99	-	4.07	0.14
GA^{lazy}_{ICS}	1	3.04	-	3.68	0.18
	3	2.59	-	3.31	0.21
	10	2.64	-	3.29	0.24
GA_{APM}	1	2.20	-	2.84	0
	3	1.83	-	2.69	0.007
	10	2.00	-	2.57	0
GA_{ICS}	1	1.45	-	1.92	0.028
	3	1.37	-	1.71	0.050
	10	**1.25**	-	**1.67**	0.021

- Though the same algorithm APM is used on GA^{lazy}_{APM} and GA_{APM}, or ICS is used on GA^{lazy}_{ICS} and GA_{ICS} respectively, their performances differ by from 1.5 to 2 times, it is shown that the re-scheduling strategy contributes to the performance improvement.

- GA_{ICS} is $30-50\%$ better than GA_{APM}. At least in this problem the Independent Constraint Satisfaction is superior to Additive Penalty Method. In addition, especially in GA_{ICS}, the longer time is given to the algorithm, the better performance is presented.

- Of course the assumption of (A), the estimation is perfect, is unrealistic. However, it is shown that a little improvement is expected if the estimation would be more accurate.

4.7 The dependency of the margin and the re-scheduling frequency

At the previous experiment in the case of (B), only the margin $r_{mgn}=4\%$ is used for the time-step $T_{step}=20[min]$, experientially known as the most adequate value. In this subsection, we examine the influence of this two parameters on the performance, the algorithm is fixed to GA_{ICS} and the computation time to 10 seconds. Here, r_{mgn} vary from 1% to 8% , and T_{step} vary from 10 minutes to 1 hour. Table 3 shows the result of 20 trials.

It is clear that, the smaller margin is employed, the fewer switches are required but the more overflows are observed. Further, by re-scheduling enough frequently, the risk of overflows is enough minimized. The best performance, 1.52 switches with 0.7% overflow rate, is

Table 3: The comparison of performance with various T_{step} and r_{mgn} : measured by the number of switches (Swt.) and the number of overflows (Ovf.)

T_{step}	$r_{mgn} = 1\%$		$r_{mgn} = 2\%$		$r_{mgn} = 4\%$		$r_{mgn} = 8\%$	
[min]	Swt.	Ovf.	Swt.	Ovf.	Swt.	Ovf.	Swt.	Ovf.
60	1.84	2.21	1.87	1.60	2.09	0.92	2.34	0.20
30	1.74	0.98	1.62	0.55	1.68	0.15	1.82	0.007
20	1.55	0.65	1.55	0.23	1.67	0.021	1.77	0
10	1.48	0.09	**1.52**	**0.007**	1.64	0	1.80	0

about 15% superior to the former research using the equivalent environment [Aoki 2003].

5 Conclusion and Future work

Scheduling problems, especially real world problems, often consist of multiple and complex constraints. For handling these constraints, it is promising way to introduce the notion "penalty". Conventionally this is used additively to the primary objective (APM), in this paper we discussed the defect of APM, i.e. the premature convergence to a easygoing solution.

We introduced independent constraint satisfaction (ICS) for constraint handling. From a view, ICS is a parallel search strategy among subspaces divided to each primary values, and from another view ICS is the ϵ-constraint method for multi-objective optimizations.

The sewerage system control problem is an actual scheduling problem consisting of strong constraints. In addition to the difficulty of the constraint handling, the most important characteristic of this problem is that the scheduling is based on the rough estimation. Though conventional controllers follow the schedule once made as long as possible, we recommended the eager re-scheduling in order to minimize the mismatch between estimated inflows and actual ones.

The eminent effectivity of our two main ideas was confirmed by experiments using an actual data. We insist that these ideas, especially ICS, broaden the applicability of computational optimization commencing with GAs for real-world scheduling problems consisting of hard constraints.

For the ICS, the methodology how to divide the whole space to subspaces and in which order they are searched, is desired. On the other hand for the sewerage system control, considering actual use, how to treat secondary (4 or more) objectives is also the future work.

Bibliography

[Aoki 2003] K.Aoki, H.Kimura, A.Nagaiwa, S.Kobayashi: "Adaptive Control of Sewerage Systems using Distributed Reinforcement Learning", IEE Japan, Vol. 123-D No.4 pp.462-469 (2003)

[Bhaskar 2000] Bhaskar V, Gupta SK, Ray AK. Applications of multiobjective optimization in chemical engineering. Reviews in Chem Eng 2000; 16(1): pp. 1-54 (2000)

[Deb 2000] Deb, K. : An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering (2000)

[Francisco 2000] Francisco O. Jr. , James R.S. : A Genetic Algorithms with a modified Desirability Function Approach to Multiple Response Optimization, Electronic Proceedings for IIE 2002 Annual Conference , No.2125 (2000)

[Giffler 60] Giffler, B. and Thompson, G. : Algorithms for Solving Production Schedulng Problems, Oper. Res. , Vol. 8, pp. 487-503 (1960)

[Ikeda 2000] Ikeda, K. and Kobayashi, S. : GA based on the UV-structure Hypothesis and Its Application to JSP, 6th Parallel Problem Solving from Nature, pp.273-282(2000)

[Kurisu 1994] Kurisu, Nisiya, Tachi, Adachi : "An application of Dynamic Distribution Scheduling to Water Supply Scheduling Using Mathematical Programming and Heuristic method", The Society of Instrument and Control Engineers, Vol.30, No.2, pp. 198-207 (1994)

[Sakamoto 2000] Y.Sakamoto, F.Kurokawa, M.Sano, T.Yamada, T.Ashiki, H.Yuki: "Quasi-Optimization of Water Distribution Scheduling Based on GA", IEE Japan, Vol 120-D No. 8/9 pp.987-999 (2000)

[Yamada 96] Yamada, T. and Ryohei, N. : Scheduling by Generic Local Search with Multi-Step Crossover, 4th Parallel Problem Solving fron Nature, pp. 960-969 (1996);