Information Operator Scheduling by Genetic Algorithms

Takeshi Yamada, Kazuyuki Yoshimura, and Ryohei Nakano

NTT Communication Science Laboratories,
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, Japan
{yamada, kazuyuki, nakano}@cslab.kecl.ntt.co.jp
http://www.kecl.ntt.co.jp/as

Abstract. In this paper, we discuss an approach to an operator scheduling problem in a large organization over time with the aim of maintaining service quality and reducing total labor costs. We propose a genetic algorithm (GA) with a parameterized fitness function inspired by homotopy methods and with null mutation to handle a variable number of operators. The proposed method is applied to the practical problem of scheduling operators in a telephone information center. Experimental results show that the proposed method performs consistently better than a GA method previously developed.

1 Introduction

In the operator scheduling problem for customer service operations at a telephone information center, we are given a set of working shifts with known start and end times and number of short breaks to be taken during the work session. The primary objective is to minimize staff shortages against number of customer calls over time. This objective is so important to maintain service quality that it is treated as a constraint such that the shortage must be zero. The secondary objective is to minimize labor costs or a surplus of operators for actual needs. Other objectives such as overtime and employee satisfaction are not considered. This problem reflects the very significant needs of a large organization such as an information service center for telephone directory assistance. Constructing a good schedule by hand, however, can be very difficult. Nippon Telegraph and Telephone Corporation (NTT), for example, has more than one hundred such centers all over Japan and currently suffers huge deficits. There is urgent demand to automatically supply efficient schedules in a short time corresponding to frequently changing work shift patterns and distribution of customer calls.

Genetic algorithms (GAs) have been successfully applied to a variety of scheduling problems including jobshop and flowshop [2, 7, 4, 6, 8]. Yoshimura and Nakano [9] first applied GAs to the information operator scheduling problem. They proposed a GA with mutation especially dedicated to the problem and a partial reinitialization method with good success. The more general form of the problem is discussed in [3] under the name of the employee scheduling

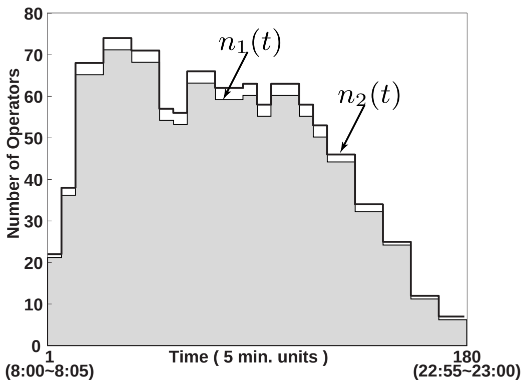

Fig. 1. Time distribution of required information operators

problem, where they proposed tabu search approach to solve the problem and compared with other methods.

The organization of this paper is as follows. Section 2 explains the information operator scheduling problem and its objective functions. In Section 3, we briefly review the GA approach previously proposed by Yoshimura and Nakano [9] and then modify their mutation to handle a variable number of operators. In Section 3.4 a GA with ranking based selection, duplicate elimination and local search is proposed. A new approach using a parameterized fitness function is proposed in Section 4. Experimental results using real data supplied by NTT are reported in Section 5.

2 The Information Operator Scheduling Problem

The number of human operators required to deal with inquiry calls from customers changes over time, and its time distribution is given based on statistical data at each center. Figure 1 shows such an example sampled at one of NTT's largest centers. The service starts at t = 8:00 and ends at t = 23:00. The time interval is measured in units of five minutes, therefore the total service time interval of 15 hours corresponds to $T = 180$ time units. The vertical axis represents the number of required operators for each time unit and is denoted by $n_1(t)$. A solid line at the top of the $n_1(t)$ histogram represents tolerable surplus $n_2(t)$: an acceptable margin of at most 5 % at each time unit to absorb daily fluctuations.

A *shift type* specifies the work starting and ending times and the number of breaks to be taken during the work session. The number of breaks depends on the length of the shift type, and the length of one break is fixed at 10 minutes (= 2 time units). A *break pattern* is a placement of breaks under a given *shift type.* A *working pattern* of an operator can be represented by specifying its *shift type* and *break pattern.* An operator can choose any shift type from a list of admissible shift types and any break pattern under the constraint that the length of any

Table 1. List of admissible working shift types

No.	Shift Type	Breaks	No.	Shift Type	Breaks
1	$8:00 \sim 12:00$	4	9	$14:00 \sim 18:00$	4
2	$8:30 \sim 12:00$	3	10	$14:00 \sim 19:00$	5
3	$8:30 \sim 12:30$	4	11	$17:00 \sim 20:00$	3
4	$8:30 \sim 13:00$	4	12	$17:00 \sim 21:00$	4
5	$9:00 \sim 13:00$	4	13	$17:30 \sim 22:00$	4
6	$9:00 \sim 14:00$	5	14	$17:30 \sim 23:00$	5
7	$13:00 \sim 17:00$	4	15	$19:00 \sim 23:00$	4
8	$13:00 \sim 17:30$	4			

continuous working period must be between 30 and 60 minutes. Table 1 shows an example set of admissible shift types and the number of breaks. For example, a shift type with starting time 8:00 and ending time $12:00$ has four breaks.

Let us assume there are a total of D operators available per day, and each of these operators is assigned a shift type selected from Table 1 and a break pattern. A schedule is obtained by finding a combination of D working patterns with possibly different shift types and break patterns. Each chosen working pattern corresponds to a (partial) schedule of one operator. Please note that even though the total number D is fixed, the total labor costs differ depending on the total length of the chosen shift types. In a center, operators must work in pairs, thus a working pattern is shared by two operators. To avoid confusion, however, we simply assume that one working pattern corresponds to one operator.

Let $n(t)$ be a number of operators working at time t under a schedule S. Total shortage of operators f_1 and total surplus f_2 are defined in Equation (1), where $\lfloor x \rfloor = x$ if $x > 0$; otherwise $\lfloor x \rfloor = 0$.

$$f_1 = \sum_{t=1}^{T} \lfloor n_1(t) - n(t) \rfloor, \qquad f_2 = \sum_{t=1}^{T} \lfloor n(t) - n_2(t) \rfloor. \tag{1}$$

The objective of the information operator scheduling problem is to minimize f_2 under the constraint that f_1 must be zero. In [9], a single f with a constant $a \in [0, 1]$ in Equation (2) is used as a fitness measure. Another type of fitness function shown in Equation (3) can also be considered where the constant α must be small enough to satisfy $f_1 = 0$.

$$f = \frac{a}{f_1 + 1} + \frac{1-a}{f_2 + 1} \qquad (0 \leq a \leq 1). \tag{2}$$

$$f^{\alpha} = \frac{1}{1 + F^{\alpha}}, \quad F^{\alpha} = f_1 + \alpha f_2 \quad (0 \leq \alpha \leq 1). \tag{3}$$

3 Genetic Algorithms

3.1 Solution representations

A schedule S consists of a set of partial schedules of all the operators and is denoted by $S = \{s_1, s_2, \ldots, s_D\}$. Each partial schedule s_i is a working pattern of an operator and is represented by a string of 10 integer-valued genes as $s_i = a_1 a_2 \ldots a_{10}$, where a_{10} represents its shift type number given in Table 1 and a_9 the number of breaks, whereas $a_1 a_2 \ldots a_8$ represent continuous working length before and after the breaks, thus a break pattern altogether. Each $a_1 a_2 \ldots a_8$ must be between 30 (= 6 time units) and 60 minutes (= 12 time units) as described in Section 2, and only $a_1, \ldots, a_m (m = a_9 + 1 \leq 8)$ is actually used. For example, an operator who starts working at time t_s and ends at t_e as specified by a shift type a_{10}, first works for a duration specified by a_1, then takes the first ten-minute (= 2 time units) break, and resumes work for a duration specified by a_2 and so on. The following equality must be satisfied (for more details, please refer to [9]):

$$t_s + a_1 + 2 + a_2 + 2 + \ldots + a_m = t_e \; (6 \leq a_j \leq 12, m = a_9 + 1). \qquad (4)$$

3.2 Mutation

For each s_i probabilistically selected for mutation with probability p_{mut}, one of the following **M1** to **M4** is applied with the probabilities p_1, p_2, p_3 and p_4 $(p_1 + p_2 + p_3 + p_4 = 1)$, respectively.

M1 Two genes a_{j_1} and a_{j_2} are randomly selected and their values are exchanged.
M2 A gene a_{j_1} with a value greater than 30 minutes is randomly selected and decreased by 5 minutes, another gene a_{j_2} with a value smaller than 60 minutes is randomly selected and increased by 5 minutes.
M3 $a_1, \ldots, a_m$ are randomly regenerated under the constraint Equation (4), while a_9 and a_{10} remain the same.
M4 a_{10} is probabilistically changed to the next (a_{10}+1) or the previous (a_{10}−1) type in Table 1, a_9 to the corresponding number of breaks, and then a_1, ..., a_m are randomly generated with the new a_9, a_{10} under Equation (4).

The mutation defined above assumes the number of genes and the total number of operators D are fixed. However, it is desirable to extend the mutation to allow D to be varied within an upperbound D_0 during the search to find a solution of higher quality. A special gene *null* for a_{10}, meaning that the operator is *off* duty, is introduced for this purpose. The following mutation **M5**, called null mutation, is applied with probability p_5.

M5 (null mutation) : a_{10} is probabilistically changed to *null*.

The mutation **M4** is slightly modified to incorporate this change such that if a_{10} is *null*, it is changed to any type in Table 1 at random.

```
1. **Initialize population:** randomly generate a set of $P$ schedules.
2. Repeat Step 2a to Step 2d $L_1$ times:
   (a) Select two schedules $S_1$, $S_2$ from the population with probabilities inversely proportional to their fitness ranks.
   (b) Apply **crossover** with probability $p_{cross}$ and obtain $T_1$ and $T_2$, otherwise just copy $S_1$ and $S_2$ to $T_1$ and $T_2$.
   (c) For $i = 1, 2$, repeat as follows $L_2$ times:
       Apply **mutation** to $T_i$ and obtain $\overline{T_i}$. If $\overline{T_i}$ is at least as good as $T_i$, replace $T_i$ with $\overline{T_i}$.
   (d) For $i = 1, 2$, if $T_i$ is better than the worst in the population, and no member of the current population has the same fitness as $T_i$, replace the worst individual with $T_i$.
3. Output the best member in the population and terminate.
```

Fig. 2. Genetic local search for information operator scheduling problem

3.3 Crossover

A partial schedule-wise uniform crossover is employed as follows. Let two parent solutions be $S_1 = \{s_{11}, s_{12}, \ldots, s_{1m}\}$ and $S_2 = \{s_{21}, s_{22}, \ldots, s_{2m}\}$. Before applying crossover to S_1 and S_2, their partial schedules s_{ij} are sorted first by a_{10} and then by $a_1, \ldots, a_m$ in the case of ties. Let us denote the results by $S_i = s^*_{i1} s^*_{i2} \ldots s^*_{im}$ $(i = 1, 2)$. A new schedule T_1 is generated by selecting s^*_{1j} or s^*_{2j} randomly for each j $(1 \leq j \leq m)$. Similarly T_2 is generated by selecting a s^*_{ij} for each j that is not selected for T_1.

3.4 Genetic local search

It is well known that there are some problem classes in which GAs are not well suited for fine-tuning structures that are very close to optimal solutions and that it is essential to incorporate local search methods into GAs. The result of such incorporation is often called *Genetic Local Search (GLS)* [5]. In this framework, an offspring obtained by a recombination operator is not included in the next generation directly but is used as a "seed" for the subsequent local search. The local search moves the offspring from its initial point to the nearest locally optimal point, which is included in the next generation.

The mutation discussed in Section 3.2 is used for local search here. Instead of applying mutation only once to each individual generated from the crossover, it is applied repeatedly and the results are accepted only when they are improved (or at least the same). Figure 2 shows the outline of our GLS algorithm based on the steady state model with ranking selection. The reinitialization method introduced in [9] is substituted by the duplicate elimination technique in Step 2d to avoid premature convergence even under a small-population condition.

Table 2. Performance comparison under various parameter conditions

No.	parameters		results			
	fitness	D	$f_1(avg.)$	$f_2(avg.)$	$f_2(best)$	$D(avg.)$
I	f in [9]	105	0	173.5	135	-
II	$f^{\alpha=1}$	105	9.7	118.1	-	-
III	$f^{\alpha:1\to0}$	105	0	143.1	123	-
IV	$f^{\alpha:1\to0}$	100	3.0	102.5	-	-
V	$f^{\alpha:1\to0}$	≤ 120	0	128.7	107	103.2
VI	$f^{\alpha:1\to0}$	≤ 105	0	119.9	97	102.8

4 Parameterized Fitness Function

Yoshimura and Nakano use f in Equation (2) with $a = 0.7$ as a fitness function [9]. They observe that their GA finds a solution with $f_1 = 0$ effectively as long as $a > 0.3$, while the quality of f_2 is not always excellent. In their experiments, f_1 and f_2 start from large values, then f_1 quickly decreases to zero and does not increase again, while f_2 decreases very slowly. Once a solution S with $f_1(S) = 0$ is found, a new solution S' with $f_1(S') > 0$ is difficult to survive because $f(S')$ is inferior to $f(S)$ in many cases. Thus only a limited region where f_1 is always 0 is searched. On the other hand, if f^α with $\alpha = 1$ in Equation (3) is used, both $f1$ and f_2 decrease smoothly, but f_1 does not reach zero or close to zero. One may be able to overcome this dilemma by finding an optimal α in Equation (3), but this itself is quite difficult.

As a possible remedy, we treat α as a parameter, which decreases from 1 to 0 throughout the search. For our purpose, the algorithm in Figure 2 is slightly modified to use the parameterized f^α in which α is first initialized as $\alpha = 1$ in Step 1, and is changed as $\alpha := (1 - \epsilon)\alpha$ in Step 2b after the crossover is applied, where $\epsilon > 0$ is a small constant.

The idea of a parameterized fitness function is inspired by a far more sophisticated approach known as the homotopy method, which has been used for decades to find solutions of nonlinear equations [1]. By initializing $\alpha = 1$, we start from a relaxed problem in which minimizing f_2 is easier at the cost of violating the constraint $f_1 = 0$. α is then gradually decreased to enforce $f_1 = 0$ and finally a schedule with $f_1 = 0$ and reasonably small f_2 is obtained.

5 Experimental Results

Numerical experiments based on the data given in Figure 1 and in Table 1 are carried out under various conditions. Figure 3 shows the average time evolutions of f_1 and f_2 over 40 runs each on a SUN Ultra30 workstation. The programs are written in C language, and each run takes about 25 minutes of CPU time. All experiments were conducted under these conditions: the population size $P = 9$, the crossover and mutation rate $p_{cross} = 0.2$ and $p_{mut} = 0.02$ respectively,

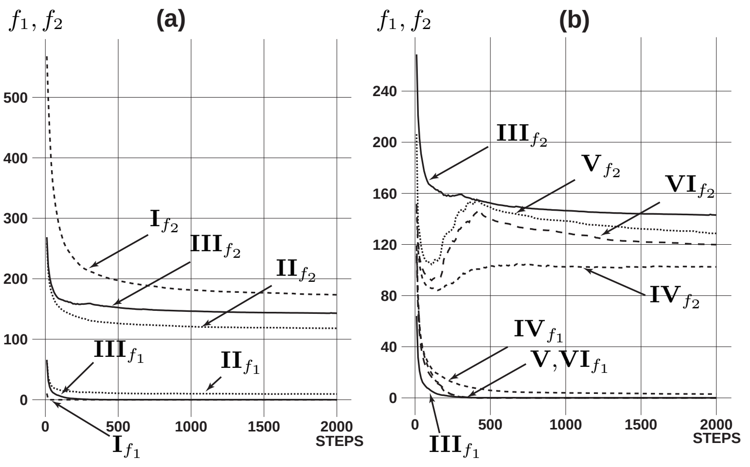

Fig. 3. Time evolutions of f_1 and f_2 under (a) fixed and parameterized fitness functions, and (b) different D settings

probabilities for each mutation: $p_1 = 0.2$, $p_2 = 0.45$, $p_3 = 0.1$ and $p_4 = 0.25$, $L_1 = 250$, $L_2 = 2000$ and $\epsilon = 0.99$ are used. These values are determined based on preliminary experiments. Results are summarized in Table 2. In Figure 3(a), the number of operators D is fixed to 105 as in [9]. In the figure, the results of three different fitness functions are compared: (**I**) the same fitness function used in [9], (**II**) f^α with $\alpha = 1$, and (**III**) the parameterized f^α with α decreasing from 1 to 0. It is clear that by using the parameterized fitness function, the quality of f_2 greatly improves while the constraint $f_1 = 0$ is satisfied at the end of the computation.

In Figure 3(b), the null mutation **M5** is applied as well as **M1**,. . .,**M4** to make D changeable. D is initialized as $D_0 = 120$ under (**V**) and $D_0 = 105$ under (**VI**) respectively, and can be varied during the search with D_0 the upper bound. p_4 is modified from 0.25 to $0.25 \times (1 - 1/N_s)$, and $p_5 = 0.25 \times 1/N_s$ with N_s number of shift types. The results of fixed D with (**III**) $D = 105$ and (**IV**) $D = 100$ are also shown for comparison. It can be seen that changing D dynamically results in better performance, with the optimal D around 102–103. The results under (**IV**) suggest that it is quite difficult to find a good solution with $f_1 = 0$ when $D \leq 100$. The best results are obtained when the perameterized fitness function and the modified mutation is used. Figure 4 shows one of the best schedules obtained under (**VI**). The picture on the right in Figure 4 shows the schedule, where the x axis represents time and the y axis the chosen working patterns. The filled block indicates that an operator is at work, while each small white block is a ten-minute break. Among 105 operators initially assigned, a total of 102 operators were found to be actually necessary, and the total surplus is 97, meaning that only 0.54 operators on average are redundant per time interval. The picture on the left shows the corresponding time distribution of the operators.

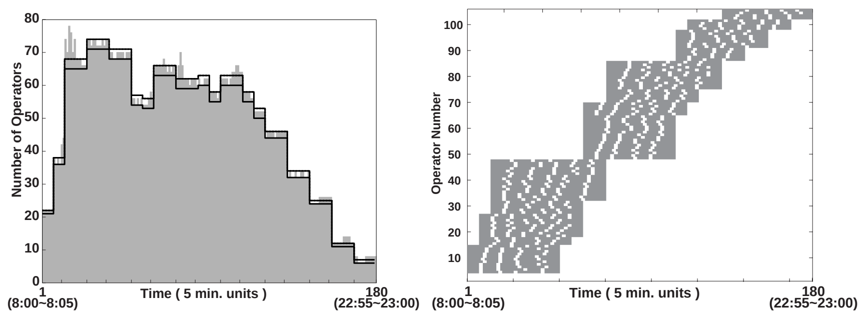

Fig. 4. Example of a solution with $D = 102, f_1 = 0$ and $f_2 = 97$

6 Conclusions

We have developed a genetic algorithm with local search for the information operator scheduling problem. The experimental results show that the use of a parameterized fitness function and null mutation improves the solution quality with a smaller number of total operators, while satisfying the given constraints. Future research will be to investigate the better control of α rather than decreasing it monotonically.

References

1. E.L. Allgower and K. George. Numerical Continuation Methods: An Introduction. Springer-Verlag, 1990.
2. C. Bierwirth, D. Mattfeld, and H. Kopfer. On permutation representations for scheduling problems. In 4th PPSN, pages 310–318, 1996.
3. F. Glover and C. MacMillan. The general employee scheduling problem: An integration of MS and AI. Computers and Operations Research, 13(5):563–573, 1986.
4. S. Kobayashi, I. Ono, and M. Yamamura. An efficient genetic algorithm for job shop scheduling problems. In 6th ICGA, pages 506–511, 1995.
5. N.L.J. Ulder, E. Pesch, P.J.M. van Laarhoven, H.J. Bandelt, and E.H.L. Aarts. Genetic local search algorithm for the traveling salesman problem. In 1st PPSN, pages 109–116, 1994.
6. T. Yamada and R. Nakano. Job Shop Scheduling. Chapter 7 in A.M.S. Zalzala and P.J.Fleming (Ed.) Genetic algorithms in engineering systems. The Institution of Electrical Engineers, London, UK, 1996.
7. T. Yamada and R. Nakano. Scheduling by genetic local search with multi-step crossover. In 4th PPSN, pages 960–969, 1996.
8. T. Yamada and C.R. Reeves. Permutation flowshop scheduling by genetic local search. In 2nd IEE/IEEE Int. Conf. on Genetic ALgorithms in Engineering Systems (GALESIA '97), pages 232–238, 1997.
9. K. Yoshimura and R. Nakano. Genetic algorithm for information operator scheduling. In Proc. of 1998 IEEE Int. Conf. on Evolutionary Computation (ICEC'98), pages 277–282, 1998.