An Iterated Local Search Algorithm for Shift Scheduling Problem with Transportation Cost

Xiaoyi Gu[†,1], Yannan Hu[†,2], Hideki Hashimoto[b,3], Mutsunori Yagiura[†,2]

[†]Dept. Mathematical Informatics, Graduate School of Informatics, Nagoya University, Nagoya, Japan

[b]Dept. Logistics and Information Engineering, Tokyo University of Marine Science and Technology, Tokyo, Japan.

[1]gu.xiaoyi@k.mbox.nagoya-u.ac.jp, [2]{yannanhu, yagiura}@nagoya-u.jp, [3]hhashi0@kaiyodai.ac.jp

***Abstract*—Classical shift scheduling problem aiming at minimizing total cost usually has the focus merely on cost generated by staff undertaking shifts. In our research, with the intention of making the problems more general that can be applied to a variety of real-world settings, we refer to another famous topic called *vehicle scheduling problem* (VSP) and propose a model of shift scheduling problem with transportation cost (or moving cost). The *integer programming* (IP) model will be provided. In order to solve the model, we also propose an effective iterated local search algorithm. The computational results by our algorithm will be compared with some currently famous *mixed integer programming* (MIP) solvers.**

***Keywords*—scheduling problem, iterated local search, transportation cost.**

I. Introduction

Shift scheduling problems as a well-known topic in operation research has already been researched for many years [5]. It is the process of assigning shifts to staff in order to obtain lowest cost [6]. However, in many real-world cases, the number of shifts and staff can be numerous, and with the consideration of different constraints such as time windows, the entire problem can be much more arduous. This study extends classical shift scheduling problem in which the transportation cost (or moving cost) is involved. Taking advantage of the extension of a classical problem, it shows that our model can be much more adaptive to be utilized in a variety of real-world cases. Furthermore, we will formulate our research as a *integer programming* (IP) problem and provide the model, some moderate size of instances can be solved by *mixed integer programming* (MIP) solvers such as CPLEX and GUROBI. However, it is time-consuming; and it is not able for MIP solvers to solve instances with large numbers of staff and shifts. As a result, we also propose an iterated local search algorithm which is efficient and effective to find approximately optimal solutions.

In addition, our research can be also related with another famous topic called *vehicle scheduling problem* (VSP). In the most basic model of VSP, a set of timetabled trips with fixed time interval (between departure and arrival time) are assigned to several identical vehicles. The objective is to find the minimal total cost including the operation and fuel cost of vehicles [8]. Relatively, we can regard shifts in our reserach as trips in VSP while regard staff as vehicles. However, in most VSP, depots are considered in which each vehicles starts its trips from a depot and will return to it after finishing its works. Depots can be both single or multiple [7]. The difference here is that in our research there is no such concept of depot, the cost or time will be only calculated between two consecutive shifts, a staff member starts from his first shift and finish at his last shift. Furthermore, since in the basic model of VSP, vehicles are identical, the cost of each vehicle undertaking trips are also identical. Costa, Isabel and Paixao [1] proposed a single depot VSP with multiple vehicle types firstly in 1995, in which the cost of different vehicles undertaking different trips also varies. Bunte and Kliewer [8] make an overview on different types of VSP. Similar with our model that we consider a working load limit for each staff member (see section II for detailed explanation), vehicle capacities are also considered by most VSP models or its heteromorphosis. However, capacity or load exceeding will lead to infeasible solution in most VSP [3]. On the contrary, we define a soft constraint that provides penalty when a staff member works over his working load limit. Lastly, in most VSP, although travel time between each pair of locations is considered, this part of cost is usually omitted [8]. It is proved that the most basic model of VSP can be solved in polynomial time. However, VSP with multiple depots is proved to be NP-hard [2] [4].

The consideration of transportation cost in our research will be an ingenious idea that makes the entire model applicable to a substantious number of real-world cases. For example, we can regard moving time and cost as preparation time and cost of operating room nurse in order to exploit our model in the famous nurse scheduling problem.

II. Problem Description

We consider our problem based on classical shift scheduling problems. In the problem, we have two different sets of staff and shifts. We give each shifts a fixed starting time and ending time. We intend to allocate all shifts to staff that all the shifts are undertaken, and one shift is exactly undertaken by one staff member. In our problem setting, different shifts are located differently, the result is that when a staff member undertakes two consecutive shifts, he has to move from previous shift to next shift.

Since one staff member cannot undertake two different shifts simultaneously, staff must have enough time to move between two consecutive shifts he undertakes. That is, the ending time of previous shift plus moving time should be earlier than the starting time of next shift. It is noteworthy here is that, in our research, the moving time between two same shifts varies from one staff member to another.

The objective of our problem is to find a decent schedule plan in which the total cost is minimized. Here the total cost includes the labor cost (when a staff member undertakes a shift), moving cost (when a staff member moves between two different shifts) and work load violation penalty. It is noteworthy that we give a work load limit for each staff member, penalty may be generated if one works over his work load.

III. Integer Programming Model

In this section, we will provide the IP model of our research and it can also contribute to giving a more detailed research explanation.

We difine $M = \{1, \ldots, m\}$ as the set of all staff while $N = \{1, \ldots, n\}$ as the set of shifts, and in the research, we use subscript i to refer to one single staff member while using subscript j to refer to one single shift. It is noteworthy that we stipulate shift 0 and $n + 1$ respectively as the beginning and ending of an individual's schedule. Furthermore, we define b_i as the maximum working load whilst p_i as the penalty coefficient. In the research, the starting and ending time of a shift will be defined as s_j and e_j. Simultaneously, the load to be discussed in our research will be shift working hours (labor load) and the traveling time between the workplaces of two consecutive shifts (moving load). The shift load is defined as L_{ij} while the moving load is defined as $L^M_{ijj'}$. Lastly, the shift and moving cost is defined as C_{ij} and $C^M_{ijj'}$ in which the moving cost is calculated with two consecutive shifts j and j'. All the parameters are shown as below.

$M = \{1, \ldots, m\}$: set of staff

$N = \{1, \ldots, n\}$: set of shifts

b_i: working load limit for each staff member

p_i: penalty coefficient used to calculate the penalty cost when some staff members work over b_i

s_j: starting time of shift j

e_j: ending time of shift j

C_{ij}: labor cost for staff member i undertaking shift j

L_{ij}: labor load for staff member i undertaking shift j

$C^M_{ijj'}$: moving cost for staff member i to travel from shift j to shift j'

$L^M_{ijj'}$: moving load for staff member i to travel from shift j to shift j'

There are three parts included in the model which are the working hour load violation penalty, total labor cost and total moving cost. As shown in (1), we are questing for the minimum total cost which is the sum of the three parts. Furthermore, we have three variables in our research which are x_{ij}, $y_{ijj'}$ and z_i. We use variable x_{ij} to mark if shift j is undertaken by staff member i (x_{ij} equals to 1) or not (x_{ij} equals 0); simultaneously, we use another variable $y_{ijj'}$ to mark if staff member i undertakes shift j' right after finishing shift j ($y_{ijj'}$ equals to 1) or not ($y_{ijj'}$ equals to 0). Also, here z_i represents the violating part of working load limit, it will be decided by using constraint shown in (8) and (9). With regards to the constraints in our research. Besides (8) and (9), we have (2) which is to ensure each shift is exactly undertaken by only one staff member. It is noteworthy that we can easily change this number to make the model adaptive to more different real-world settings. We also have (3) which is to ensure there is no time conflict for every staff member that he cannot undertake two different shifts simultaneously and he must be able to start the next shift after moving from previous shift. It is noteworthy that staff cannot arbitrarily early finish his job until the fixed ending time of a shift. Lastly, we have (4) to (7) which is to decide the value of $y_{ijj'}$ automatically based on the value of x_{ij}. Only if staff member i undertakes two consecutive shifts j and j', the value of $y_{ijj'}$ will be given as 1, otherwise 0. Namely, only if x_{ij} and $x_{ij'}$ both equal to 1 and there is no other $x_{ij''}$ equals 1, the value of $y_{ijj'}$ can be given as 1. The IP model is shown as below.

minimize :

$$\sum_{i\in M} p_i z_i + \sum_{i\in M}\sum_{j\in N} C_{ij}x_{ij} + \sum_{i\in M}\sum_{j\in N\cup\{0\}}\sum_{j'\in N\cup\{n+1\}} C^M_{ijj'}y_{ijj'} \tag{1}$$

subject to :

$$\sum_{i\in M} x_{ij} \ (\forall j \in N) \tag{2}$$

$$(e_j + t^M_{ijj'}) \cdot y_{ijj'} \le s_{j'} \cdot x_{ij} \ (\forall i \in M, \forall j \in N) \tag{3}$$

$$y_{ijj'} \le 1 - x_{ij''} \ (\forall i \in M, \forall j, j' \in N, j'' \in \{j'' \mid e_j \le s_{ij''}, e_{j''} \le s_{j'}) \tag{4}$$

$$y_{ijj'} \ge x_{ij} + x_{ij'} - 1 - \sum_{j''\in\{j''\mid e_j \le s_{j''}, e_{jj''} \le s_{j'}\}} x_{ij''} \ (\forall i \in M, \forall j \in N, j' \in \{j' \mid s_j < s_{j'}\}) \tag{5}$$

$$y_{ijj'} \le x_{ij} \ (\forall i \in M, \forall j, j' \in N) \tag{6}$$

$$y_{ijj'} \le x_{ij'} \ (\forall i \in M, \forall j, j' \in N) \tag{7}$$

$$z_i \ge \sum_{j\in N} L_{ij}x_{ij} + \sum_{j\in N\cup\{0\}}\sum_{j'\in N\cup\{n+1\}} L^M_{ijj'}y_{ijj'} - b_i \ (\forall i \in M) \tag{8}$$

$$z_i \ge 0 \ (\forall i \in M) \tag{9}$$

IV. Local Search Algorithm

In this part, we will introduce our proposed local search algorithm for solving the problem explained above. In general, the algorithm can be divided into four steps incorporating initialization, relocation, one-to-one swap, and one-to-multiple swap. Furthermore, the last three steps will also be iterated until reaching a given time to avoid local optimum. We would like to explain them step by step.

A. Initialization

At the beginning point of our algorithm, no shift is assigned to any staff member. As a result, we intend to insert shift one by one into each staff member's schedule. The insertion is based on a simple greedy method, that is, we insert a shift into a staff member's schedule with lowest labor cost C_{ij}. Under the circumstance that multiple staff members have lowest labor cost C_{ij} simultaneously when undertaking shift j, the shift will be inserted into the first staff member with lowest labor cost. It is noteworthy that the greedy insertion method may cause time conflict since both the time interval of shifts and the moving time between two shifts are not considered during the entire process. However, in order to enlarge the solution space, we would like to introduce another *evaluation function* to calculate the total cost. As shown in (10), we define another penalty to calculate the time overlap part. Here the *evaluation function* f_e equals to the objective f_{obj} which is defined in chapter III as shown in (1), plus the penalty cost by time conflict. p''_i is the penalty coefficient while $y_(ijj')$ is the variable which equals to 1 only if staff member i undertakes two consecutive shifts j and j'. The last part $(s_(j') - e_j - t_(ijj')^M)_+$ calculates time overlap, when there is no time overlap it will be 0. In the entire process of our algorithm, the *evaluation function* will be used instead of the *objective function* (1) in order to make the solution space larger. Obviously, we can simply change p''_i to a relatively large number to eliminate all time conflict later. The time complexity of initialization will be $O(mn)$.

$$f_e = f_{obj} + \sum_{i \in M} \sum_{j \in N} \sum_{j' \in N} p''_i y_{ijj'} (s_{j'} - e_j - t^M_{ijj'})_+ \tag{10}$$

B. Relocation

After initialization step, all shifts have been already assigned to staff, we intend to relocate shifts in order to obtain lower cost. In this step, we will check from the first shift of the first staff member and intend to insert it into every different staff members' schedule. The result will be kept if the total cost calculated by *evaluation function* (10) becomes lower; otherwise change will not be adopted. This step will be processed until the last shift of the last staff member has been attempted to insert into all other staff members' schedule. As shown in Fig. 1, the second shift of staff member 1 is relocated and inserted into staff member 2's schedule between the second and third shifts. The time complexity of relocation will be $O(mn)$.

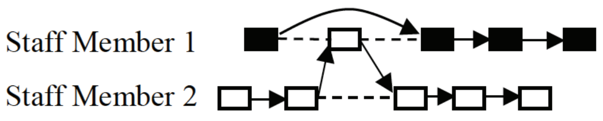

Fig. 1. Relocation

C. One-to-one Swap

The next step in our algorithm is called one-to-one swap. In simple terms, we intend to swap two shifts of two different staff members and keep the result if lower cost is found. It is similar with relocation, we start this step from the first shift of the first staff member and intend to swap it with every shift from all other staff members. However, after a lower cost has been found, the result will be kept, and the algorithm will return to relocation. Distinctively, here we intend to insert all the shifts merely into the schedule of the two staff members who is affected by one-to-one swap. For examples, after we swap two shifts from staff member 1 and 2, lower cost has been found and the result has been kept. The algorithm will return to the step of relocation, instead of trying inserting shifts into all staff members' schedule, we only need to try inserting shifts (of other staff members except staff member 1 and 2) into the schedule of staff member 1 and 2. In the case of trying inserting shifts of staff member 1 or 2, we just need to try inserting a shift into the other staff member's schedule. As shown in Fig. 2, we have swapped the third shift of staff member 1 with the third shift of staff member 2. The time complexity of one-to-one swap will be $O(n^2)$.

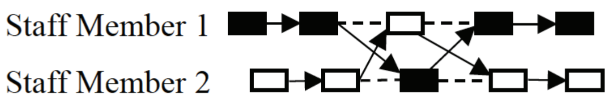

Fig. 2. One-to-one swap

D. One-to-multiple Swap

The last step of our proposed algorithm is called one-to-multiple swap. The process of this step is almost same as one-to-one swap that has been introduced just before. Differently, it swaps from the first shift of the first staff members with every two consecutive shifts of other staff members (one-to-two swap). And similarly, the result will be kept, and the algorithm will also return to relocation if lower cost is found. The relocation here is also same as which in one-to-one swap, all the shifts will merely be inserted into the schedule of the two staff members who is affected by one-to-multiple swap. After the one-to-two swap has been done, it also tests one-to-three swap further, and following process of relocation will be the same as one-to-two swap. As shown in Fig. 3., we have swapped the third shift of staff member 1 with shift 3 and 4 of staff member 2, and in Fig. 4., we have swapped the third shift of staff member 1 with shift 2, 3 and 4 of staff member 2.

The time complexity of $O(n^2)$.

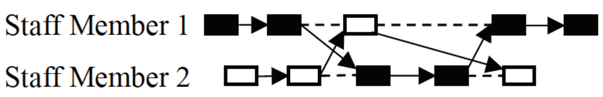

Fig. 3. One-to-two swap

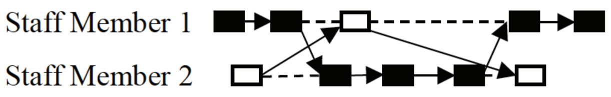

Fig. 4. One-to-three swap

E. Iteration

After the first four step has been done, local optimum will be reached. However, in order to avoid local optimum and intending to obtain a better solution, we also propose iteration process for our algorithm. Here we have another step called **KICK**, in which we arbitrarily swap two shifts (not consecutive) of one staff member with another staff member's two random shifts (also not consecutive). If the staff member selected has only one shift, we will check another staff member. The entire algorithm will be processed until the given time limit is reached.

The whole process of our proposed algorithm is arranged in the table below described as Algorithm 1.

Algorithm 1 Iterated Local Search

```
1: Initialization, insert each shift one by one with a greedy
   method based on lowest labor cost.
2: Relocation, relocate each shift to find lower total cost, until
   every shift is checked.
3: One-to-one swap, swap each pair of two shifts from two
   different staff members' schedules. Try inserting every shift
   into the schedule of the two staff members who is affected by
   the swap if lower cost is found. This step will be processed
   until every shift is checked and no lower can be found.
4: One-to-two swap, swap each shift with every pair of two
   consecutive shifts of another staff member. Return to step 2 if
   lower cost is found. Try inserting every shift into the schedule
   of the two staff members who is affected by the swap if lower
   cost is found. This step will be processed until every shift is
   checked and no lower can be found.
5: One-to-three swap, swap each shift with every pair of three
   consecutive shifts of another staff member. Return to step 2 if
   lower cost is found. Try inserting every shift into the schedule
   of the two staff members who is affected by the swap if lower
   cost is found. This step will be processed until every shift is
   checked and no lower can be found.
6: Time limit check, check if it reaches given time limit, if not
   execute KICK and return to step 2, otherwise the algorithm
   ends.
```

V. Computational Results

The proposed algorithm is coded in C++ and run on a computer with a 1.80 GHz Intel Core i7-4500U processor and 8.00 GB memory. Results solved by CPLEX 12.1.0 and GUROBI 8.0 on the same computer will also be provided as a benchmark to test our algorithm.

In our research, the instances are generated arbitrarily by using C++. As shown in TABLE I, we have the results which are solved by CPLEX, GUROBI,our proposed algorithm without iteration (end by step 5, see Algorithm 1 explained in section IV, abbreviated in LS) and the iterated local search (ILS) we proposed. The instance here, for example 3-10, represents that there are 10 shifts waiting for being assigned to 3 staff members. The Gap here is calculated by the MIP solver.

As shown in TABLE I, especially when we test the instance with 10 staff members and 30 shifts, we can obtain the optimal result by 3.48 seconds (1231.69s for CPLEX and 328.45s for GUROBI) with no iteration. Also, when the number of staff members and shifts become larger, although the results by one-off local search becomes slightly worse than which is solved GUROBI, the process time is much lower. Moreover, it is capable to find a feasible solution (without time conflict) when testing the instance with 5 staff members and 100 shifts in only 184.22 seconds.

Lastly, it is able for our proposed algorithm to obtain optimal solution when testing the instance with 10 staff members and 40 shifts. And better solution is found when testing instance with 5 staff members and 60 shifts.

VI. Conclusions

We propose a novel idea that considering transportation cost in classical scheduling problem, which makes the entire model become more general and more applicable to real-world cases. Our iterated local search algorithm performs decently in solving moderate-size instances. However, amelioration is still required. In future work, we would like to firstly consider other local search methods which may perform better than current algorithm. Additionally, higher processing speed of the algorithm will also be researched.

References

[1] A. Costa, I. Branco and J.M.P. Paixao, "Vehicle Scheduling Problem with Multiple Type of Vehicles and a Single Depot," in *Computer-Aided Transit Scheduling*, J.R. Daduna, I, Branco and J.M.P. Paixao, Eds. Berlin: Springer, 1995, pp. 115–129.
[2] B.A. Foster and D.M. Ryan, "An Integer Programming Approach to the Vehicle Scheduling Problem," *Operational Research Quarterly*, vol. 27, no. 2, pp. 367–384, 1976.
[3] C.D.T. Watson-Gandy, "The Vehicle Scheduling Problem: A Survey," *New Zealand Operational Research*, vol. 9, no. 2, pp. 73–92, 1981.
[4] G. Desauliniers, J. Lavigne and F. Soumis, "Multi-depot vehicle scheduling problems with time windows and waiting costs" *European Journal of Operational Research*, vol. 111, no. 3, pp. 479–494, 1998.
[5] J. Bailey, "Integrated days off and shift personnel scheduling," *Computer & Industrial Engineering*, vol. 9, no. 4, pp. 395–404, 1985.
[6] M. Brucker, *Scheduling Algorithms*, Berlin: Springer, 2007.
[7] R. Freling, A.P.M. Wagelmans and J.M.P. Paixao, "Models and Algorithms for Single-Depot Vehicle Scheduling," *Transportation Science*, vol. 35, no. 2, pp. 165–180, 2001.
[8] S. Bunte and N. Kliewer, "An overview on vehicle scheduling models," *Public Transport*, vol. 1, no. 4, pp. 299–317, 2009.

TABLE I
COMPUTATIONAL RESULTS

	CPLEX			GUROBI			LS		ILS	
Instances	Time	Cost	Gap	Time	Cost	Gap	Time	Cost	Time	Cost
3-10	0.03	**234.6**	0%	0.02	**234.6**	0%	0.11	**234.6**	1800	**234.6**
3-20	0.29	**586.2**	0%	0.62	**586.2**	0%	1.13	**586.2**	1800	**586.2**
10-30	1231.69	**456.5**	0%	328.45	**456.5**	0%	3.48	**456.5**	1800	**456.5**
10-40	7200	626.5	0.75%	6214.4	**621.8**	0%	9.98	624.8	1800	**621.8**
5-60	7200	1317.0	3.48%	7200	1305.7	2.60%	21.70	1309.4	3600	**1304.8**
10-60	7200	1101.8	28.56%	7200	**1019.0**	18.90%	15.23	1043.8	7200	1022.5
3-100	7200	2879.6	1.48%	7200	**2877.1**	1.40%	72.49	2901.2	7200	2878.9
5-100	10800	n/a	n/a	10800	n/a	n/a	184.22	1901.0	7200	**1894.6**