Journal of the Operations Research
Society of Japan
2007, Vol. 50, No. 4, 563-575

BOUNDS FOR STAFF SIZE IN HOME HELP STAFF SCHEDULING

Atsuko Ikegami
Seikei University

Aki Uno
Dai-ichi Life Information Systems

(Received October 30, 2006; Revised March 3, 2007)

Abstract Home help organizations provide services at respective users' homes at a time that is convenient for the user. Helpers with time window constraints for their working hours must be assigned to these services to ensure that the services are provided. Producing adequate schedules usually takes the schedulers a considerable amount of time due to many types of constraints and requirements. It is difficult even to estimate the number of helpers needed.

In this paper, we introduce a mathematical programming formulation of the home help staff scheduling problem, and propose two types of lower bounds for the number of helpers needed on a specific day. We then also calculate the number of helpers using a simple heuristic algorithm in order to determine the upper bound. Combining the bounds with the information obtained when determining the bounds was shown to be useful for arranging the appropriate helpers for each day before making a detailed schedule, even though each of the algorithms that obtains the bounds is very simple.

Keywords: Health care, staff scheduling, mathematical modeling, transportation, network flow

1. Introduction

In this paper we deal with the problem of scheduling staff for home help.

There are usually three types of helpers working at a home help organization: fulltime, part-time and registered helpers. Each type of helper has different *time window constraints* concerning their working hours. A user requires specific types of services with *time constraints*, e.g., a user requires physical care Tuesday 8:00-9:00 and housekeeping Wednesday 10:00-11:30, where the service takes 'the whole hour' rather than 'a shorter time within the time window'. In addition, there are *provision constraints* that concern the relationship between a user and the helpers, and the relationship between the type of service the user requires and the types of service the helper can provide. Under these restrictions the scheduler must assign an appropriate helper to each service while considering *traveling time* between users' homes as well as the workload of each helper, the payroll and so on.

Although research for this scheduling problem has not yet started, we realized that this problem has similar characteristics to the nurse scheduling problem [6][7][8] in terms of the quality of services and the workload of staff. In addition, when we focus on scheduling for a specific day, we can regard it as a vehicle routing problem with time constraints [1][5], although there are also provision constraints.

It is difficult to assign helpers to services appropriately due to all of the abovementioned constraints and many other requirements. The scheduler therefore needs a considerable amount of time to produce an adequate schedule. However, a practical schedule cannot be created when there is a lack of helpers even for a single day. It is also difficult to estimate the number of helpers needed due to many types of constraints. Therefore, this paper discusses

lower bounds for the number of helpers needed for a specific day. We deal with the problem of finding a feasible schedule for a specific day as a subproblem of this scheduling problem. Then, by setting two types of relaxations for the subproblem as a transportation model, the number of helpers needed can be estimated. The upper bound for the number of helpers is also calculated by obtaining a feasible schedule using a simple heuristic algorithm. Finally, we discuss the plan for developing scheduling algorithms.

To explain our ideas concretely, we use a small-sized subproblem throughout, which finds a schedule for a single day.

2. Model

We define the 'home help staff scheduling problem' as follows. When services with time constraints are requested by users, helpers with time window constraints for their working hours must be assigned to these services, while considering provision constraints and traveling time between users concerning the services, so as to ensure that the services are provided. It is absolutely essential in this problem that all of the services required are provided. As well as the abovementioned constraints, a scheduler must consider many other less strict requirements, e.g., minimizing waiting time (interval time), minimizing the number of helpers, balancing workloads or pay between helpers, balancing helpers who provide services to a specific user for a scheduling period, and so on. Therefore, we must include these requirements in the objective function of this problem as goals.

To formulate this problem, we use *decision variable* x_{ijkh}, which equals 1 if helper i deals with service h immediately after service k on day j, otherwise 0. The necessary conditions for $x_{ijkh} = 1$ are (a) helper i satisfies the provision constraints for service k and h, (b) the time window for helper i includes the time ranges for these services, and (c) there is enough time for the helper to reach the house and provide service h after finishing service k. To express these conditions in a formulation of this scheduling problem, we use notations p_{ijk} and q_{jkh}. Only when the time window constraints of helper i and the time constraint of service k on day j, and the provision constraints between the helper and the service are satisfied, do we set $p_{ijk} = 1$, otherwise $p_{ijk} = 0$. And only when there is adequate time to travel between the users concerning service k and h on day j, do we set $q_{jkh} = 1$, otherwise $q_{jkh} = 0$.

We formulate this problem by describing the objective function in a flexible form such that we are able to set it appropriately according to given goals, as follows.

Notation

$M = \{1, 2, \ldots, m\}$: the set of ID-numbers for helpers involved.
$N = \{1, 2, \ldots, n\}$: the set of days in the scheduling period.
$U_j = \{k \mid k$ is a service that is required on day $j\}$: the set of services that are required on day j.
u_{ij}: the upper bound for the number of services that helper i can provide on day j.
p_{ijk}: 1 exactly if (i) helper i satisfies the provision constraint for service k on day j, and (ii) the time interval assigned to service k on day j is contained in the time window of helper i for day j, and 0 otherwise.
q_{jkh}: 1 if it is possible to visit the user concerning service h immediately after the user concerning service k on day j subject to the time constraints and the traveling time*, otherwise 0. For convenience, we set $k = 0$ as the help station where helpers start and

*When we must consider prep time in between services, we can set q_{jkh} according to the traveling time and the prep time. Alternatively, we can deal with prep time by including it as part of traveling time.

return each day, and then set $p_{ij0} = 1, i \in M, j \in N$, $q_{j0k} = q_{jk0} = 1, k \in U_j, j \in N$ and $q_{j00} = 0, j \in N$.
G: a finite set of goals g.
$X = \{x_{ijkh} \mid i \in M,\ j \in N,\ k, h \in U_j \cup \{0\}\}$: the set of *decision variables*.
$f_g(X)$: the function to measure the shortcomings of a solution when goal $g \in G$ is not satisfied.

Formulation

$$\text{Minimize} \sum_{g \in G} f_g(X) \tag{2.1}$$

subject to

$$\sum_{i \in M} \sum_{k \in U_j \cup \{0\}} x_{ijkh} = 1, \qquad h \in U_j,\ j \in N, \tag{2.2}$$

$$\sum_{k \in U_j \cup \{0\}} x_{ijkh} = \sum_{k \in U_j \cup \{0\}} x_{ijhk}, \qquad i \in M,\ h \in U_j,\ j \in N, \tag{2.3}$$

$$\sum_{h \in U_j} x_{ij0h} \leq 1, \qquad i \in M,\ j \in N, \tag{2.4}$$

$$\sum_{k \in U_j \cup \{0\}} \sum_{h \in U_j} x_{ijkh} \leq u_{ij}, \qquad i \in M,\ j \in N, \tag{2.5}$$

$$x_{ijkh} \leq p_{ijk} \cdot p_{ijh} \cdot q_{jkh}, \qquad i \in M,\ j \in N,\ k, h \in U_j \cup \{0\}, \tag{2.6}$$

$$x_{ijkh} = 0 \text{ or } 1, \qquad i \in M,\ j \in N,\ k, h \in U_j \cup \{0\}. \tag{2.7}$$

Constraint (2.2) makes certain that each service is provided by one, and only one, helper. Constraints (2.3) and (2.4) keep the network flow even and consistent. Constraint (2.5) ensures that the number of services is less than/equal to a given upper bound. Constraint (2.6) considers provision constraints, time constraints for services, time window constraints for helpers, and traveling time between users.

3. Relaxations of a Subproblem

The graph in Figure 1 shows the number of available helpers and the number of users who require services at a specific time on a specific day in a set of real data (Table 1 and Table 2). While there does not appear to be a lack of helpers, because all of the striped areas associated with users are covered by the frame associated with helpers, this graph does not consider provision constraints, traveling time or prep time, although originally they need to be considered. In actual fact, there is a lack of helpers.

To determine whether there are enough helpers for the services required on a specific day, we set a subproblem as follows; the subproblem finds a feasible schedule for a specific day. We then propose two types of lower bounds for the number of helpers needed on a specific

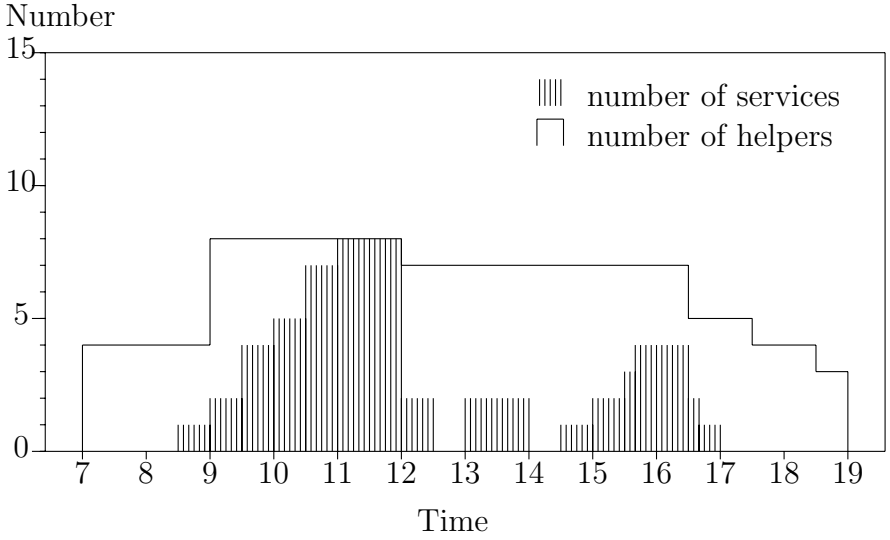

Figure 1: Number of available helpers and number of users who need services at a specific time

Table 1: Time window constraints for available helpers

helper	start time	finish time
1	9:00	16:30
2	9:00	17:30
3	7:00	19:00
4	9:00	16:30
5	7:00	12:00
6	9:00	18:30
7	7:00	19:00
8	7:00	19:00

Table 2: Time constraints for services

service	start time	finish time
1	10:30	12:00
2	15:30	16:30
3	9:30	12:30
4	9:00	11:30
5	8:30	10:30
6	15:00	17:00
7	10:30	12:00
8	10:30	12:00
9	14:30	16:30
10	13:00	14:00
11	15:40	16:40
12	10:00	12:00
13	11:00	12:00
14	9:30	12:00
15	13:00	14:00
16	11:30	12:30

day by solving two types of relaxations of the corresponding subproblem, because schedulers need lower bounds to arrange for additional helpers when there is a lack of available helpers.

The first type of relaxation relaxes provision constraints and upper bound constraints. This allows any helper to provide any service while ignoring his/her workload. The second type of relaxation relaxes time constraints for services. This relaxation allows any helper to provide services at a convenient time, in a convenient order.

Thus, in the first type of relaxation, we set $p_{ijk} = 1$ and $u_{ij} = \infty$ for any helper i and any service k, and in the second type of relaxation, we set $q_{jkh} = 1$ for any services k and h. As a result, each of these relaxations can be solved easily because we can deal with them as the Hitchcock transportation problem [4].

3.1. Lower bound (1)

In the transportation problem obtained by the first type of relaxation, the supply nodes correspond to the home help station with enough helpers and each helper who has just finished a service and can now work again. The demand nodes correspond to each service and the home help station that helpers return to at the end of the day. A supply node and a demand node are connected by an arc if there is a helper available to provide the service given the traveling time, prep time and time constraints for the service. The assumed amount of supply on the help station node is the total number of services and the assumed amount of supply on the other nodes is 1. Similarly, the assumed amount of demand on the help station node is the total number of services and the assumed amount of demand on the other nodes is 1. When we solve this transportation problem by setting $cost = 1$ to each arc from the home help station to each service and set $cost = 0$ to the other arcs, we can minimize the number of helpers needed [2][3] (See Figure 2).

For the abovementioned data (Table 1 and Table 2) with the traveling time (Table 3)† and 10 minutes prep time between services, we obtained 9 helpers as the lower bound (*lower bound(1)*). This means that there is a lack of at least one helper.

Table 3: Traveling time between users who required services

user	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	20	15	10	10	10	10	10	10	10	15	20	20	10	30
2	0	0	20	15	10	10	10	10	10	10	10	15	20	20	10	30
3	20	20	0	30	15	15	15	15	10	15	15	10	20	15	10	20
4	15	15	30	0	20	20	20	20	20	25	20	25	40	40	20	40
5	10	10	15	20	0	3	3	3	5	10	3	5	25	20	5	20
6	10	10	15	20	3	0	3	3	5	10	3	5	15	20	5	20
7	10	10	15	20	3	3	0	3	5	10	3	5	15	20	5	20
8	10	10	15	20	3	3	3	0	5	10	3	5	15	20	5	20
9	10	10	10	20	5	5	5	5	0	10	5	10	15	15	1	25
10	10	10	15	25	10	10	10	10	10	0	8	5	10	10	10	15
11	10	10	15	20	3	3	3	3	5	8	0	5	15	20	5	20
12	15	15	10	25	5	5	5	5	10	5	5	0	15	15	10	20
13	20	20	20	40	25	15	15	15	15	10	15	15	0	5	20	15
14	20	20	15	40	20	20	20	20	15	10	20	15	5	0	20	20
15	10	10	10	20	5	5	5	5	1	10	5	10	20	20	0	25
16	30	30	20	40	20	20	20	20	25	15	20	20	15	20	25	0

In addition, when we solve this transportation problem concerning the first K services sorted by start time in the day, then continue to increase K (i.e., $K = 1, 2, \ldots,$ *the total number of the services required in the day*), we can obtain the lower bound for the number

†We set the traveling time between the help station and any user as 0, because helpers visit users' homes directly from their homes and return to their homes directly when they do not have time to visit the help station before/after work.

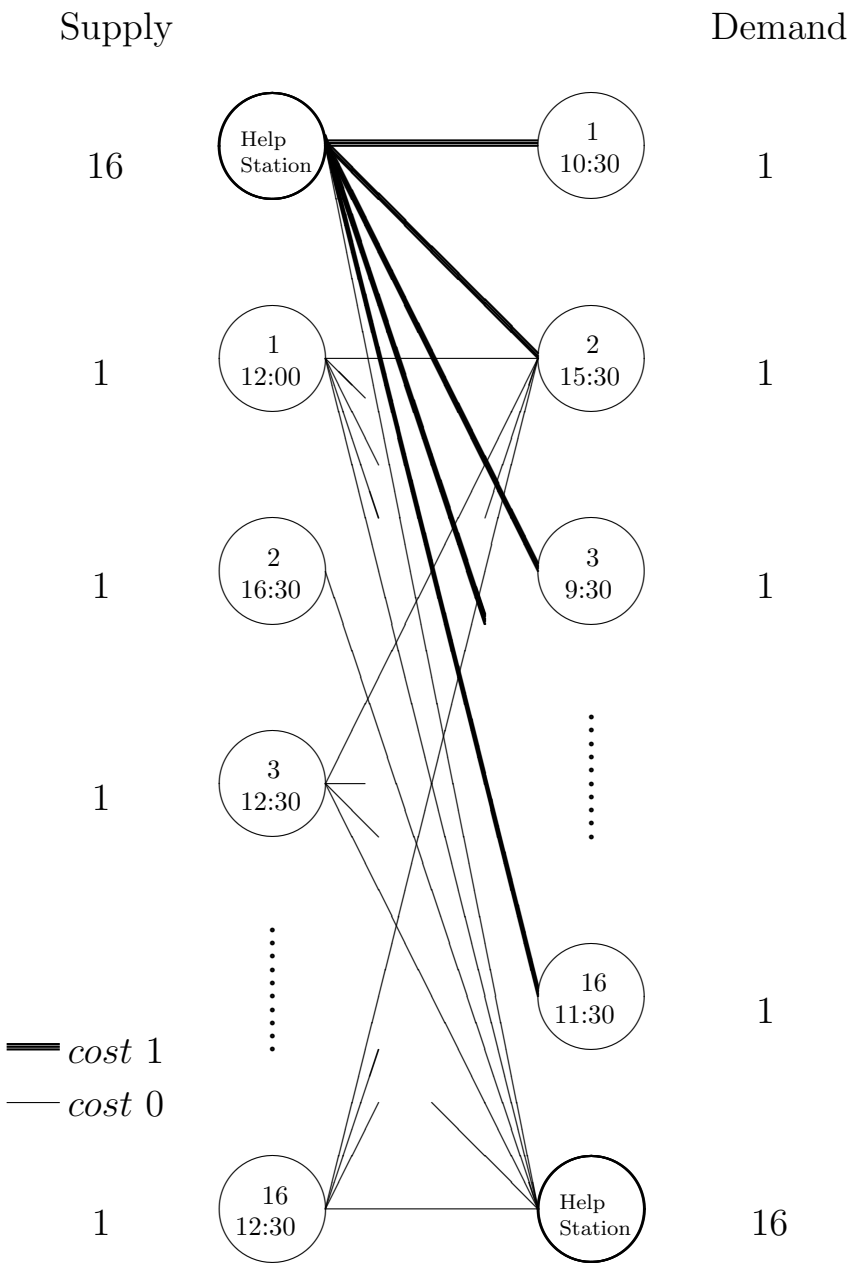

Figure 2: Transportation problem obtained by the first type of relaxation

of helpers needed at that particular time period (*lower bound (1)a*). Similarly, when we solve this transportation problem concerning the last K services sorted by finish time while increasing K, we can obtain the lower bound for the number of helpers who should stay in order to provide the remaining services (*lower bound (1)b*). When represented on a graph with the numbers of helpers on the ordinate and time on the abscissa (see Figure 3), we see the rough outline of the time when helpers should start work (lower bound (1)a) and the time when helpers may finish work(lower bound (1)b). Furthermore, when counterbalancing these numbers, the number of helpers needed at each time can be also estimated. We can see that an additional helper is needed from 11:00 to 11:30. Therefore, we can obtain another lower bound by adding the number of additional helpers to the number of helpers available on the day (*lower bound (1)c*). In this example, the lower bound is $8 + 1 = 9$.

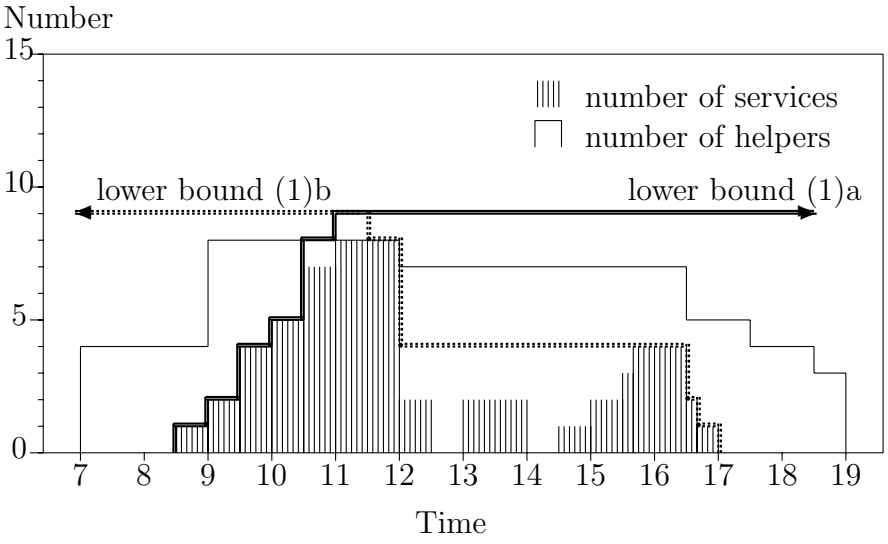

Figure 3: Lower bound for the number of helpers needed at a specific time

3.2. Lower bound (2)

The second type of relaxation is a relaxation of the time constraints for services. This means that any helper who satisfies the provision constraints for a specific service can provide the service by changing the time of the service. This relaxation also produces a transportation problem where the supply nodes correspond to each helper available on the day and a dummy helper who can provide any service, and the demand nodes correspond to each service and a dummy service that is needed in order to balance total supply with demand (refer to Figure 4). The number of services a helper can provide on that day (i.e., u_{ij}) is set as the amount of supply for the corresponding supply node. When we solve this problem by setting $cost = 1$ to each arc from the dummy helper to each service and setting $cost = 0$ to the other arcs that are described according to the provision constraints, we can check whether there is a lack of helpers. When the dummy helper provides any service, we cannot obtain a feasible schedule.

Table 4 shows the provision constraints between the services (Table 2) and the helpers (Table 1). A blank cell means the helper cannot provide the service, whereas 1 means the helper can[‡]. When we solved the transportation problem for this set of data and decreased the value of u_{ij} from the usual upper bound of 3 or 4, we found that the dummy helper

[‡]The provision constraints depend on the type of service, e.g., helpers who can provide physical care to a specific user are not same as helpers who can provide housekeeping to the user. In Table 4, the provision

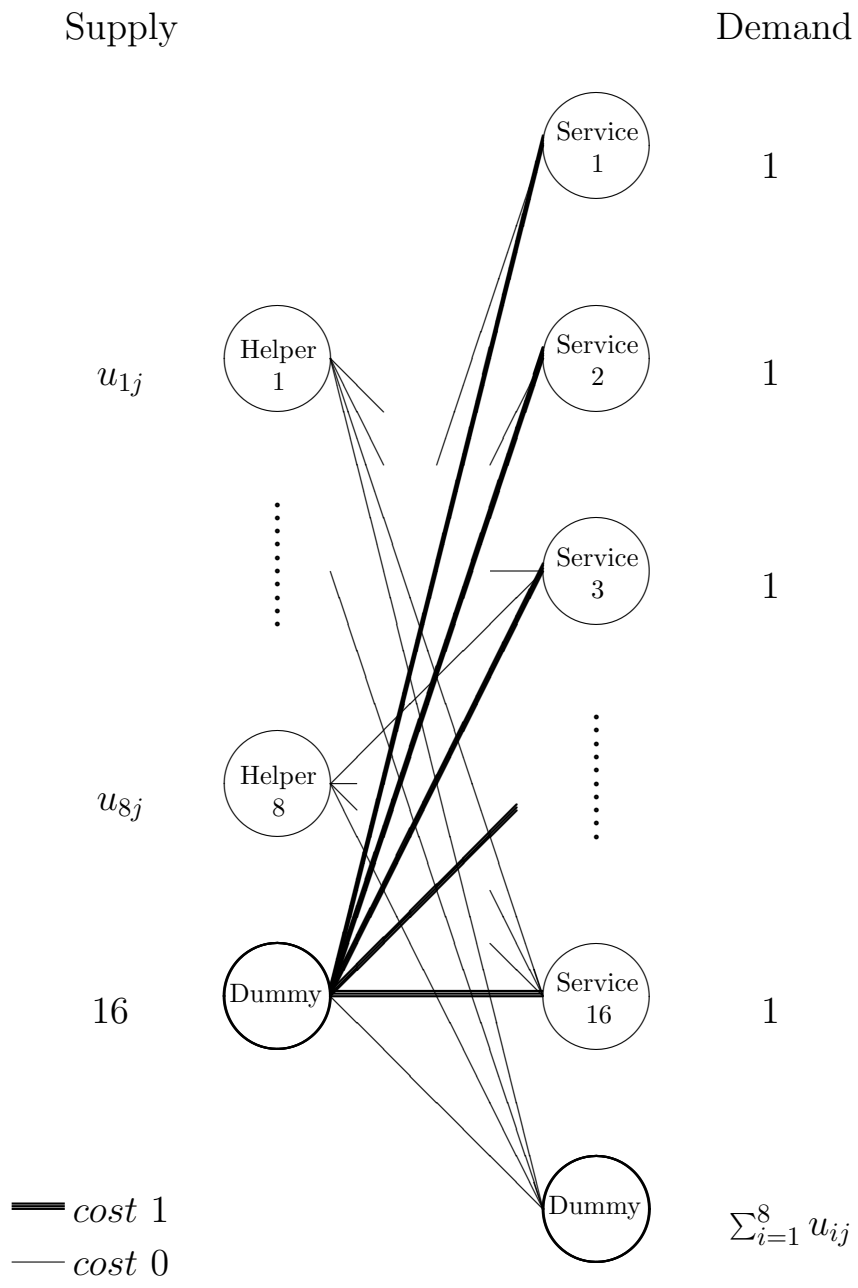

Figure 4: Transportation problem obtained by the second type of relaxation

provided service 6 when $u_{ij} = 2, i \in M$. Thus, we cannot obtain a feasible schedule when each helper can provide no more than 2 services.

Table 4: Provision constraints between 16 services and 8 helpers

	helper							
service	1	2	3	4	5	6	7	8
1							1	
2							1	
3			1		1			1
4	1							1
5	1	1					1	1
6							1	
7	1	1	1					
8	1	1	1					
9		1			1			
10	1			1				1
11		1				1	1	1
12		1			1	1	1	
13				1				1
14	1	1				1		
15	1					1		
16	1	1		1	1	1		

When combining the results from the two types of relaxation, we can see that at least one extra helper is needed. In more detail, the scheduler needs to find preferably one helper who can work from 11:00 and can provide service 6 from 15:00. The time window of the additional helper who was chosen and new provision constraints are shown in Table 5 and Table 6, respectively.

Table 5: Time window constraints for the additional helper

helper	start time	finish time
9	7:00	19:00

Table 6: Provision constraints between 16 services and 9 helpers

	helper								
service	1	2	3	4	5	6	7	8	9
1							1		
2							1		
3			1		1			1	
4	1							1	
5	1	1					1	1	1
6							1		1
7	1	1	1						1
8	1	1	1						1
9		1			1				
10	1			1				1	
11		1				1	1	1	
12		1			1	1	1		
13				1				1	1
14	1	1				1			
15	1					1			
16	1	1		1	1	1			1

constraints are described according to the type of the service, although we do not describe the type of service explicitly in this paper.

4. Upper Bound

To determine the range where we must search for the minimum number of helpers needed, we also calculate the upper bound for this number.

We use a simple scheduling algorithm for a single day, where we minimize the number of services not yet covered and then minimize the number of helpers needed. For convenience, we define the objective function as follows: minimize *the number of services not yet covered*$\times BIG+$*the number of helpers needed*, where we set BIG far greater than the number of helpers available on the day.

In this algorithm, we first create a set of feasible schedule patterns (feasible routes) for helper $i \in M$ subject to Constraints (2.3) (2.4) (2.5) and (2.6) concerning a specific day, and keep the patterns in P_i. Next, we set a day-off pattern to each helper and then repeat, changing the patterns while minimizing the objective value based on Tabu Search, where we use TL as the length of a tabu list. The algorithm ignores patterns that were resigned in the previous TL iterations when updating the current trial solution. Let r_{ip} be the iteration number that indicates when pattern p of helper i has been resigned. In addition, we define ITE as the limit of the number of iterations for stopping the algorithm. Let z_i be the optimum objective value for helper i, and i^* be as $z_{i^*} = \min_{i \in M} z_i$. The algorithm can then proceed as follows.

Algorithm

Step 1. For all $i \in M$, create a set of feasible schedule patterns for P_i, and set $r_{ip} = -\infty$ for all $p \in P_i$.

Step 2. Create a day-off pattern p_0 and set $P_i = P_i \cup \{p_0\}$ for all $i \in M$.

Step 3. Set $p^i = p_0$ for all $i \in M$.

Step 4. Set $iteration = 1$.

Step 5. For all $i \in M$, set $TABU_i = \{p | r_{ip} \geq iteration - TL, p \in P_i\} \cup \{p^i\}$ and find a pattern $p \in (P_i \setminus TABU_i)$ that minimizes the objective value when supposing the pattern is employed instead of p^i.
We call the best pattern p_i and its objective value z_i.

Step 6. Chose i^* so as to satisfy $z_{i^*} = \min_{i \in M} z_i$, and employ p_{i^*} for the next trial solution: $p^{'} = p^{i^*}, p^{i^*} = p_{i^*}$.
Set $r_{i^* p'} = iteration$, and $iteration = iteration + 1$.

Step 7. If iteration $< ITE$, go to Step 5. Otherwise, stop.
If $\lfloor z_{i^*}/BIG \rfloor = 0$, accept a set of current $p^i, i \in M$ as a feasible solution.

When we can obtain a feasible solution, we use the value of z_{i^*} as the upper bound for the number of helpers needed, given the helpers available on the day.

Here we show an example where we dealt with real monthly data given by a scheduler who works at a help station of a social welfare foundation in Tokyo. In this organization, the scheduler has a specific number of helpers available to make the schedule on a given day. We call this situation *Case I*. There are times, however, when a helper is sick or not available for some reason, making it difficult to complete the schedule. It is at these times that fulltime staff, who can be more flexible, can become available to provide services. We call this situation *Case II*. Table 7 shows the total number of feasible patterns of helpers for each case for each day of the first week and Table 8 shows the upper bound and lower bounds of helpers.

For this data, lower bound (2) is useful for judging the feasibility given the available helpers, because the provision constraints are relatively tight. As traveling time to provide

services increases, lower bound (1) also becomes good indicator of feasibility. Table 8 shows that lower bound (1)c also indicates the infeasibility for Case I on the fifth day.

The idea of upper and lower bounds has been widely used in many of optimization algorithms, and here we would like to show that the size of the difference between them can be utilized to develop an efficient algorithm for scheduling.

Table 7: Total number of feasible schedule patterns for available helpers

			Case I					No.of	Case II (+ additional helpers)				
	No.of	No.of	u_i for each helper					additional	u_i for each helper				
day	services	helpers	2	3	4	5	6	helpers	2	3	4	5	6
1	26	14	192	293	335	343	343	2	330	589	725	753	753
2	19	8	125	220	267	273	273	2	206	370	435	441	441
3	10	4	82	127	133	133	133	1	106	164	173	173	173
4	31	11	165	264	309	315	315	2	298	606	859	963	981
5	27	18	175	127	223	223	223	2	283	421	466	476	476
6	23	12	125	168	188	192	192	2	241	434	559	590	590
7	29	12	249	472	621	645	645	2	415	875	1182	1239	1241

Table 8: Upper bound and lower bounds for the number of helpers needed

	upper bound for Case I					upper bound for Case II						
	u_i for each helper					u_i for each helper					lower bound	
day	2	3	4	5	6	2	3	4	5	6	(1)	(1)c
1	×	11	10	10	10	14	10	10	10	10	9	–
2	×	×	×	×	×	△	9	9	9	9	7	–
3	×	4	4	4	4	5	4	4	4	4	3	–
4	×∗	×∗	×∗	×∗	∗	×	×	△	△	△	10	12 (11+1)
5	15	15	15	15	15	14	13	13	13	13	12	–
6	×	×	×	×	×	12	8	8	8	8	8	–
7	×	×	×	×	×	△	12	11	11	11	10	–

× : infeasible (using lower bound (2)), ∗ : infeasible (using lower bound (1)c), △ : we cannot find any feasible solution.

5. Conclusion

In this paper, we proposed two types of lower bounds for the number of helpers needed on a specific day. These lower bounds can be obtained easily because they are based on the Hitchcock transportation problem. The upper bound can also be obtained easily using a simple heuristic algorithm. Combining the bounds with the information obtained when determining them makes it possible for schedulers to estimate the degree of shortage of helpers and which service is difficult to provide. This information is very useful for arranging the appropriate helpers for each day before making a detailed schedule, e.g., the scheduler can guess which helper to ask to work as the additional helper.

The next phase of this study will apply the principles of the *subproblem-centric approach* [6]. The subproblem-centric approach is proposed when solving a problem of block-angular structure. It obtains feasible solutions by solving each subproblem while setting the degree of violation of linking constraints as the objective function of each subproblem. We plan to use the subproblem-centric approach to obtain feasible solutions for the entire problem by solving each subproblem repeatedly while including the original objective function $\sum_{g\in G} f_g(X)$ into the objective function of each subproblem.

Although we must discuss the algorithms for the vehicle routing problem with time constraints for each subproblem, the subproblem-centric approach can also obtain feasible schedules. It finds the best work pattern for each helper so as to minimize the number of services yet to be covered and minimizes $\sum_{g\in G} f_g(X)$. The algorithm that we used to

determine the upper bound in the previous section is also based on the subproblem-centric approach, where the algorithm minimizes the number of helpers needed in addition to minimizing the number of services yet to be covered.

Table 9 shows the solution obtained by this algorithm, when we set a simple objective function as an experiment. We set minimizing the number of services yet to be covered and minimizing the total interval time as the objective function. (We confirmed that this solution is one of the optimal solutions for this objective function.)

Table 9: An example schedule for a specific day

	service									interval
helper	1			2			3			time
1	4	9:00	11:30							0
2	8	10:30	12:00	9	14:30	16:30				150
3	7	10:30	12:00							0
4	13	11:00	12:00							0
5	12	10:00	12:00							0
6	14	9:30	12:00	15	13:00	14:00	11	15:40	16:40	160
7	1	10:30	12:00	2	15:30	16:30				210
8	3	9:30	12:30	10	13:00	14:00				30
9	5	8:30	10:30	16	11:30	12:30	6	15:00	17:00	210
									total	760

We also developed a prototype of the scheduling support system using spreadsheet software. The prototype system has a simple interface for input/output and includes our algorithm. The scheduler, who works at a home help station in Tokyo, created real schedules for two months by obtaining daily schedules using the prototype system. The resultant schedules used at the home help station greatly reduced the scheduler's workload and scheduling time.

Acknowledgements

We are extremely grateful to the home help station of Shisei Gakusha Tachikawa Social Welfare Foundation. We also wish to thank Mr. Y. Ogata, Mr. K. Hashimoto and Mr. K. Sakamoto for their assistance. This research was partially funded by a Grant-in Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan ((C) No.16510120) and financial support from the High-Tech Research Center Program from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] N. Christofides: Vehicle Routing. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.): *The Traveling Salesman* (John Wiley & Sons, New York at al., 1985), 431–448.

[2] V. Chvátal: *Linear programming* (W.H. Freeman and Company, New York, 1983).

[3] G.B. Dantzig and D.R. Fulkerson: Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, **1** (1954), 217–222.

[4] F.L. Hitchcock: The distribution of a produce from several sources to numerous localities. *Journal of Mathematical Physics*, **20** (1941), 224–230.

[5] A. Ikegami and A. Niwa: The vehicle routing problem with time constraints. *Journal of the Operations Research Society of Japan*, **38** (1995), 107–123 (in Japanese).

[6] A. Ikegami and A. Niwa: A subproblem-centric model and approach to the nurse scheduling problem. *Mathematical Programming*, **97** (2003), 517–541.

[7] H.E. Miller, W.P. Pierskalla and G.J. Rath: Nurse scheduling using mathematical programming. *Operations Research*, **24** (1976), 857–870.

[8] D.M. Warner: Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research*, **24** (1976), 842–856.

Atsuko Ikegami
Faculty of Science and Technology
Seikei University
3-3-1 Kichijoji-kitamachi, Musashino-shi,
Tokyo 180-8633, Japan
E-mail: atsuko@st.seikei.ac.jp