An iterated local search algorithm for technicians and interventions scheduling for telecommunications

Hideki Hashimoto[1], Sylvain Boussier[2] and Michel Vasquez[2]

[1] Department of Applied Mathematics and Physics,
Graduate School of Informatics,
Kyoto University,
Kyoto 606-8501, Japan
hasimoto@amp.i.kyoto-u.ac.jp
[2] Ecole des Mines d'Ales, Site EERIE,
Parc Scientifque Georges Besse 30035 Nimes cedex 1 France
{Sylvain.Boussier,Michel.Vasquez}@ema.fr

1 Introduction

The problem of *technicians and interventions scheduling for telecommunications*, which we abbreviate as TIST, is a very difficult problem. Indeed, even determining which interventions are abandoned is NP-hard in the strong sense, since the precedence-constrained knapsack problem is included as a special case. After determining hired interventions, the problem is still NP-hard, since the precedence-constrained scheduling, which is NP-hard, is included as a special case. Assigning interventions to days is also NP-hard since the bin packing problem is included as a special case.

2 Outline of our algorithm

We use the iterated local search (ILS)[1], which iterates LS many times from those initial solutions generated by perturbing good solutions obtained by a sophisticated greedy algorithm.

3 Preprocessing heuristics for hired interventions

The hired interventions problem is tackled by using a preprocessing heuristics which selects interventions to be hired. These interventions are cleaned from the problem once for all at the beginning of the algorithm: the heuristics used to do so is based on the minimum number of technicians required for each intervention and the duration of interventions $T(I)$. The first phase is to compute a weight w_I for each intervention I so that $w_I = mintec(I) \times T(I)$. Let Ω_t be the set of indexes of technicians, the value $mintec(I)$, which is a lower bound of the number of technicians for a given intervention I, is given by solving the following linear problem:

$$P(I) \begin{cases} \text{Minimize } \sum_{t \in \Omega_t} x_t \text{ subject to,} \\ \sum_{t/C(t,I) \geq n, t \in \Omega_t} x_t \geq R(I,i,n) \quad \forall i,n, \\ x_t \in \{0,1\} \quad t \in \Omega_t \end{cases}$$

In the second phase, we have to find a subset of interventions $\mathcal{H}$ to be hired so that $\sum_{I \in \mathcal{H}} w_I$ is maximum and the total cost does not exceed the total budget A. The problem of finding this subset is a knapsack problem. Let (KP) be this problem and $S(x_i, x_j) = 1$ if intervention j must start after the completion of intervention i and 0 otherwise. Let Ω_I be set of indexes of interventions, the (KP) can be stated as follows:

$$KP \begin{cases} \text{Maximize } \sum_{I \in \Omega_I} w_I x_I \text{ subject to,} \\ \sum_{I \in \Omega_I} cost(I) \cdot x_I \leq A, \\ x_I \leq x_j \quad \forall I,j \in \Omega_I / S(x_I, x_j) = 1 \\ x_I \in \{0,1\} \quad I \in \Omega_I \end{cases}$$

,where $x_I = 1$ if $I \in \mathcal{H}$ and 0 otherwise. This problem is solved with a greedy algorithm which consists in selecting the interventions of maximum ratio $mintec(I) \times T(I)/cost(I)$ with no successors not hired, while the total cost does not exceed the maximal available budget A.

4 Description of the algorithm

With experimentation, we noticed that some natural criteria like the coefficient (*linked to the priority of intervention*) in the objective function or even the ratio coefficient/duration are not always the more efficient. The main idea of our approach is to find the *best* permutation of the initial priorities. For this purpose, we try several runs of greedy algorithm with the 24 possible permutations of the 4 priorities of the problem and keep the one that gives the best *greedy solution*. Then, we seek to improve this solution with local search.

The algorithm is divided in two phases: (1) find the best permutation of priority assignation for interventions and (2) search iteratively local solutions around solutions genertated by the greedy algorithm which follows the criterion assignation.

4.1 Searching the best permutation assignation

At first glance, the natural order in which we want to insert the interventions is the highest priority first order as (1,2,3,4): that means that we give the weight 28 to interventions of priority 1, 14 for the interventions of priority 2, etc. Unfortunately, the experimentation shows us that this natural order is not always the best. The figures 1 and 2 highlight this fact: in those two figures, interventions of priority 1 are the red ones, those of priority 2 are the green ones and those of priority 3 are the yellow ones. Each line represents a technician: the first technician is represented by the top line and the last technician by the bottom line. Each black box corresponds to an unavailable day for a technician and each vertical line corresponds to the end of a day.

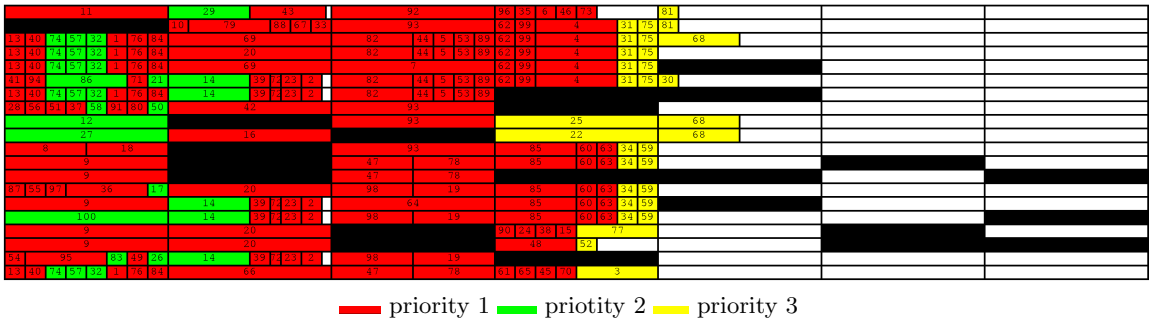

Fig. 1. Solution with objective 17820 for instance data8 of instances set A

The solution represented by the figure 1 is obtained with the permutation (1,2,3,4) and the solution represented by the figure 2 is obtained with the permutation (4,3,1,2) which is not obvious at first glance and corresponds to attribute the weight 28 to interventions of priority 4, 14 to interventions of priority 3, 4 to interventions of priority 1 and 1 to interventions of priority 2. This example illustrates the interest of searching a best order for inserting interventions using all the possible permutations. Of course, those weights are not used to evaluate the solutions but only to guide the greedy algorithm.

4.2 Greedy algorithm

Hence, in the first phase of the algorithm, we try all permutations of weights *(i.e* 24*)* and evaluate them with the a greedy algorithm. This greedy algorithm tries to insert interventions according to

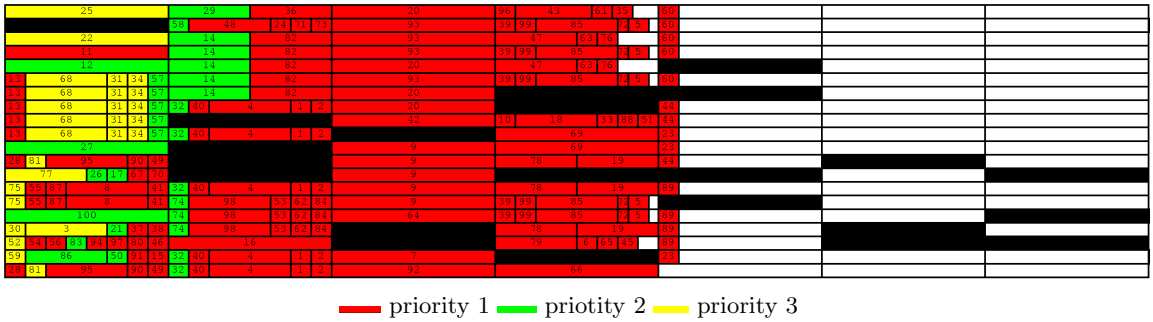

Fig. 2. Solution with objective 17355 for instance data8 of instances set A

the following criteria: (1) the earliest day possible, (2) the team which requires the less additional technicians to perform the intervention and (3) the minimum starting time possible. Then, we update the criteria of the critical interventions (last intervention of each priority) with a higher value and propagate this value to all their predecessors. The aim of this update is to tend to shift the critical interventions to the left. Experimentally, suppose I is a critical intervention, c_I its criteria and p_I its weight, then c_I and all its predecessors are updated with the value $p_I + c_I$.

4.3 Iterated local search

In order to improve the search, the greedy algorithm is executed succesively 24, 12 and 6 times. Each time the best permutations corresponding to the best objective solutions are saved and the criteria are updated. Then, we limit the search around the two best permutations of the last 6 ones and we successively execute the greedy algorithm until we reach the time limit: each time the greedy algorithm improves the current solution, the local search algorithm is executed and starts with this solution. If it improves the solution, it updates the best solution. The idea is to only search local solution around promising solutions.

5 Local search

In this section, we describe a framework of our local search (LS). It starts from an initial solution and repeats replacing the current solution with a better solution in its neighborhood until no better solution is found in the neighborhood. In this problem, after fixing the assignment of interventions to teams and the process order of interventions of each team, we can determine the feasibility of the schedule and the optimal start times of interventions. Hence we search the assignment of interventions to teams and the process order of interventions by local search and check the feasibility and determine the optimal start times at each step.

We propose two local search algorithms, which we call critical path and packing phases. In both, we use the swap neighborhood and the insertion neighborhood. A swap operation exchanges the assignment and the order of two interventions. An insert operation removes an intervention and inserts it into another position. In the local search, we consider only feasible moves, that is, the solution does not move if the operation violates a constraint.

5.1 Critical path phase

The aim of the critical path phase is to decrease each ending time of each priority (i.e., t_1, t_2, t_3 and t_4) without increasing the others. For a solution, we consider a critical path which is a sequence $(i_1, i_2, \ldots, i_l)$ of interventions such that intervention i_l gives the ending time of a priority and intervention i_{k+1} can not be scheduled unless intervention i_k is scheduled at earlier period. From the definition of a critical path, we have to schedule intervention i_1 at earlier period in order to decrease the ending time of the priority.

In the local search, for each priority, we find a critical path and search the neighborhood that intervention i_1 can be scheduled at earlier period.

5.2 Packing phase

In the packing path phase, we schedule interventions more efficiently without increasing the ending time of each priority.

We estimate the efficiency for a team t with interventions I which are assigned to it by a function

$$f(t, I) = (\text{the weighted summation of the wasted skill and time})(\text{cf. Fig. 3})$$

And we estimate the efficiency for a solution by the summation of $f(t)$ for all team. In this phase, our local search uses f as the estimation of solutions. Only a solution that does not increase the current ending time of priority can be accepted in the move of the local search.

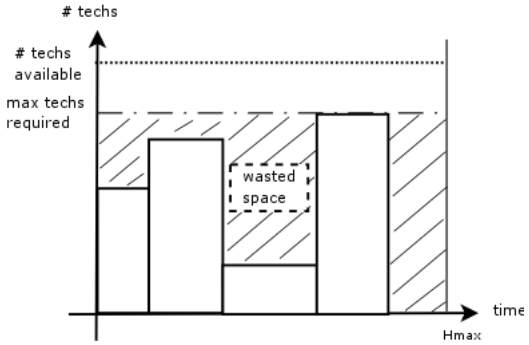

Fig. 3. Illustration of the wasted space for a domain and a level in a team

6 Computational results

For the experimental phase, we used a 3.2 GHz Pentium with 1Go of RAM memory and limited the process execution to 1200 seconds. The description of the data per column is the following:

- *instance*: The name of the instance.
- *int.*: The number of interventions.
- *tec.*: The number of technicians.
- *dom.*: The number of domains.
- *lev.*: The number of levels.
- *lb*: A lower bound of the objective value for the problem that does not contain the hired interventions.
- *obj.*: The best objective value found.
- *gap.*: The gap value to lower bound.

instance	int.	tec.	dom.	lev.	lb	obj.	gap
data1-setA	5	5	3	2	2265	2340	3,2
data2-setA	5	5	3	2	4215	4755	11,3
data3-setA	20	7	3	2	11310	11880	4,7
data4-setA	20	7	4	3	10995	13452	18,2
data5-setA	50	10	3	2	26055	28845	9,6
data6-setA	50	10	5	4	17775	18870	5,8
data7-setA	100	20	5	4	27405	30840	11,1
data8-setA	100	20	5	4	16166	17355	6,8
data9-setA	100	20	5	4	25618	27692	7,4
data10-setA	100	15	5	4	35405	40020	11,5
data1-setB	200	20	4	4	38385	43860	12,4
data2-setB	300	30	5	3	16605	20655	19,6
data3-setB	400	40	4	4	17460	20565	15
data4-setB	400	30	40	3	19035	26025	26,8
data5-setB	500	50	7	4	106290	120840	12
data6-setB	500	30	8	3	24450	34035	28,1
data7-setB	500	100	10	5	28470	35640	20,1
data8-setB	800	150	10	4	32820	33030	0,6
data9-setB	120	60	5	5	26310	29550	10,9
data10-setB	120	40	5	5	32790	34920	6

Table 1. Results obtained on benchmarks provided by France Telecom

References

1. H. R. Lourenço, O. C. Martin, T. Stützle, "Iterated Local Search," in Handbook of Metaheuristics, F. Glover, G. A. Kochenberger (eds), 321–353, Kluwer Academic Publishers, Boston, 2003.