# A GRASP-Based Approach for Technicians and Interventions Scheduling for Telecommunications.

Hideki Hashimoto, Sylvain Boussier, Michel Vasquez, Christophe Wilbaut

# A GRASP-Based Approach for Technicians and Interventions Scheduling for Telecommunications

Hideki Hashimoto[1], Sylvain Boussier, Michel Vasquez and Christophe Wilbaut[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,Christophe.Wilbaut}@ema.fr*

**Abstract.** The Technicians and Interventions Scheduling Problem for Telecommunications embeds the scheduling of interventions, the assignment of teams to interventions and the assignment of technicians to teams. Every intervention is characterized, among others attributes, by a priority. The objective of this problem is to schedule interventions such that the interventions with the highest priority are scheduled at the earliest time possible while satisfying a set of constraints like the precedence between some interventions and the number of technicians with the required skill level by domain. To solve this problem, we propose a GRASP algorithm based on the dynamic update of the weights assigned to the interventions combined with a local search procedure. We also compute lower bounds and present experimental results that validate the effectiveness of this approach.

**Keywords:** Technicians and Intervention Scheduling; GRASP; Metaheuristics

## 1. Introduction

In this paper we describe a heuristic approach for solving a Technicians and Interventions Scheduling Problem for Telecommunications which we abbreviate as **TIST**.

The subject [3] was proposed by *France Telecom*[1] for the 5th challenge of the French Society of Operations Research and Decision Analysis[2]. At *France Telecom*, supervisors have to decide for each day which technicians will work together for the day and which interventions they will have to perform. With the growth of interventions due to the expansion of the new services associated with Internet such as VoIP or television broadcasts –and to maintain their competitiveness with restricting the numbers of technicians– the supervisors have to

[1] French telecommunications company
[2] http://www.g-scop.fr/ChallengeROADEF2007/ or http://www.roadef.org/

tackle more and more complicated schedules. The aim of the TIST is to provide efficient schedules that can help the supervisors in their job.

The interventions are characterized by criteria such as priority and length of time. Some interventions are linked to other interventions that must be completed first. That constitutes a set of preceding constraints to be satisfied by any scheduling. The interventions are also composed of different types of tasks which require a given number of technicians with a certain skill level in a given domain. The technicians are specialized in different domains with different skill levels and they have a list of non-working days. Therefore it is necessary to assign interventions to teams of technicians with the required skill levels and who are available the scheduled day. In addition, any intervention has a given cost if an external company is hired to do it. The total cost of the subcontracted interventions cannot exceed a given budget. Note that interventions can be subcontracted only if their successors, in the precedence constraint, are subcontracted too. The objective of the TIST is to schedule interventions such that the interventions with the highest priority are scheduled at the earliest time possible. This problem has two related combinatorial aspects: the scheduling of the interventions (depending on the precedence constraint) and the assignment of the technicians to the teams (depending on the scheduled day).

The core of our approach to solve the TIST is constituted by a GRASP-based algorithm. The selection criteria of intervention to be scheduled by the greedy algorithm are updated during the search: new solutions are generated using the information extracted from previous ones. Then we attempt to improve these solutions with local search. Globally, our approach is divided in three phases: first, a preprocessing phase which selects the interventions to be subcontracted and delete them from the problem; second, we identify initial coefficients for the greedy algorithm in the insertion order phase that keeps the two *best* orders to insert the interventions; third, the GRASP phase which is composed of the greedy phase that generates initial solutions followed by the local search phase that tries to improve them. The remainder of the paper is organized as follows: in section 2, we describe the problem and introduce the notations; section 3 is devoted to the presentation of the different components of our approach; then, in section 4, we expose a lower bound computation, implemented to evaluate the quality of the obtained results. The whole computational experiments are given in section 5, and finally we conclude and give some further prospects to enhance the present work in section 6.

## 2. Problem Description

In this section, we first describe the problem informally. Then we present the different notations used in the paper to refer the data of the problem, and we conclude with a mathematic formulation of the TIST.

### 2.1. Global description

The problem deals with interventions that have to be assigned to teams of technicians. Technicians are described by their available days and skills, and interventions are characterized by their priority, execution time, predecessors (interventions which have to be completed before) and required number of technicians of each skill level in each domain. The aim is to build teams of technicians for each day and assign interventions to those teams while verifying all the constraints of the schedule and minimizing the objective function

$$28t_1 + 14t_2 + 4t_3 + t_4$$

where $t_k$ is the ending time of the last intervention of priority $k$ for $k = 1, 2, 3$ and $t_4$ is the ending time of the whole schedule.

A schedule has to satisfy a list of constraints for the assignment of technicians and for the assignment of interventions. We consider that each working day is in the interval $[0, H_{\max}]$ and that it is not possible to exceed this limit. Consequently, an intervention cannot be performed before time 0 or after time $H_{\max}$ and cannot be done in several days. An intervention has to be done by only one team at one time, several teams cannot share the same intervention. There are days off for the technicians and obviously, no intervention can be assigned to a technician who is not working during the current day. A strong constraint is that the teams cannot change during one day, so a technician belongs to only one team each day. This constraint is due to the limited number of available cars and to the time it would take to get several teams back to a central point to mix the teams.

A team has to satisfy the requirements to perform an intervention. Thus, for each intervention we have to assign enough qualified technicians to satisfy all the requirements. For example, an intervention requiring one technician of level 2 in the domain d1 can be done by one technician of level 2, 3 or 4 in domain d1, but cannot be done by two technicians of level 1 in domain d1. The required number of technicians at a given level for an intervention is cumulative since a technician of a given level is also qualified for all the smaller levels of the same competence domain. For example, if a technician has a skill level of 3 in a given domain then he can work on interventions requiring only a skill level of two in this domain.

Finally, it is possible to subcontract interventions to an external company. Each intervention has a specific cost to hire an external company to do it and the total cost of the subcontracted interventions cannot exceed a total budget. Let us note that the mathematical model we expose in section 2.3 does not consider subcontracted interventions. Indeed, this part of the problem is tackled with a preprocessing heuristics which is described in section 3.1.

## 2.2. Notations

In this section we introduce a set of notations used throughout the paper. First we define the constants for the problem:

- $H_{\max}$ is the length of time of each day ($H_{\max} = 120$ in the subject).
- $T(I)$ is the execution time of intervention $I$.
- $cost(I)$ is the cost of intervention $I$.
- $A$ is the total budget allowed for the subcontracted interventions.
- $P(t, j)$ is equal to 1 if technician $t$ is working on day $j$, 0 otherwise.
- $C(t, i)$ is the skill level of technician $t$ in domain $i$.
- $R(I, i, n)$ is the required number of technicians of level $n$ in domain $i$ to complete the intervention $I$.
- $Pred(I)$ is the list of interventions which have to be completed before starting intervention $I$.

We also use some variables listed below:

- $s(I)$ is the starting time of intervention $I$.
- $e(t, j)$ is the team number of technician $t$ for the day $j$. Team number 0 is a special team composed of the non-working technicians.
- $d(I)$ is the day when intervention $I$ is scheduled.

From the previous description and the previous notations, an intervention requiring for domain $i$ at least one technician of level three and one technician of level two will have its requirements noted: $R(I, i, 1) = 2$, $R(I, i, 2) = 2$, $R(I, i, 3) = 1$, $R(I, i, 4) = 0$.

Here are the constants used for the mathematical model:

- $Pr(k, I)$ is equal to 1 if the priority of intervention $I$ is $k$ and 0 otherwise.

- $\mathcal{P}(I_1, I_2)$ is equal to 1 if intervention $I_1$ is a predecessor of intervention $I_2$ and 0 otherwise.

Finally we also use the following variables in the mathematical model:

- $x(I, j, h, \epsilon)$ is equal to 1 if team $\epsilon$ works on intervention $I$ on day $j$ at the starting time $h$ and 0 otherwise.

- $y(j, \epsilon, t) = 1$ if technician $t$ is in team $\epsilon$ on day $j$ and 0 otherwise.

- $t_k$, $k = 1, 2, 3$ is the ending time of the last scheduled intervention of priority $k$.

- $t_4$ is the ending time of the whole schedule.

## 2.3. MATHEMATICAL MODEL

As we explained in section 2.1, the following mathematical model does not consider the subcontracted interventions. Let us call this problem

$(P')$, then it can be stated as follows:

$$\begin{aligned}
&\text{Minimize} \quad 28t_1 + 14t_2 + 4t_3 + t_4 \\
&\text{subject to} \sum_{j,h,\epsilon} x(I,j,h,\epsilon) = 1 \qquad \forall I && (1)\\
&y(j,0,t) = 1 - P(t,j) \qquad \forall j,t && (2)\\
&\sum_{\epsilon} y(j,\epsilon,t) = 1 \qquad \forall j,t && (3)\\
&x(I,j,h,0) = 0 \qquad \forall I,j,h && (4)\\
&\sum_{h_1=\max(h_2-T(I_1)+1,0)}^{\min(h_2+T(I_2)-1,H_{\max})} x(I_1,j,h_1,\epsilon) + x(I_2,j,h_2,\epsilon) \leq 1 \quad \forall I_1,I_2,h_2,j,\epsilon && (5)\\
&\sum_{j,h,\epsilon} (jH_{\max}+h)\,(x(I_1,j,h,\epsilon) - x(I_2,j,h,\epsilon)) + T(I_1)x(I_1,j,h,\epsilon) \leq 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad \forall I_1,I_2 \mid \mathcal{P}(I_1,I_2) = 1 && (6)\\
&x(I,j,h,\epsilon) = 0 \qquad \forall I,j,h,\epsilon \mid h + T(I) > H_{\max} && (7)\\
&\sum_{h} R(I,i,n)x(I,j,h,\epsilon) \leq \sum_{t \mid C(t,i) \geq n} y(j,\epsilon,t) \qquad \forall I,i,n,\epsilon,j && (8)\\
&\sum_{j,h,\epsilon} (jH_{\max}+h+T(I))\,Pr(k,I)x(I,j,h,\epsilon) \leq t_k \quad \forall I, k = 1,2,3 && (9)\\
&\sum_{j,h,\epsilon} (jH_{\max}+h+T(I))\,x(I,j,h,\epsilon) \leq t_4 \qquad \forall I && (10)
\end{aligned}$$

Constraint (1) ensures that each intervention is made by one team only on one day at one time. Constraint (2) says that if a technician $t$ is not working on day $j$, he is in team 0. Constraint (3) specifies that a technician belongs to only one team each day. Constraint (4) guarantees that no interventions are made by the non-working team. Constraint (5) ensures that two interventions assigned on the same day to the same team are done at different times. Constraint (6) says that all the predecessors of a given intervention have to be completed before starting this intervention. Constraint (7) ensures that the working days are strictly limited to $H_{\max}$, maximum number of time slices per day. Constraint (8) specifies that a team working on intervention $I$ has to meet the requirements concerning levels of competence for $I$. Finally, constraint (9) specifies that $t_k$ is the ending time of the last scheduled intervention of priority $k$, $k = 1,2,3$ and constraint (10) specifies that $t_4$ is the ending time of the whole schedule.

## 3. General Approach

The proposed algorithm is centered on a GRASP method. In this section, we explain the function and the principles of each part of the approach.

A first part consists in dealing with the problem of selecting which interventions will be subcontracted or not according to the available budget. As this problem is not so trivial, it is tackled with a preprocessing heuristics detailed in section 3.1. Once the interventions to be subcontracted are selected, they are deleted once for all from the problem, and the GRASP method described in section 3.2 is then applied on the problem $(P')$. One of the main part of the GRASP method is the greedy algorithm, which is used to generate feasible solutions of the problem. As described in section 3.2.1, it is also used to update dynamically the weights of the interventions during the search. The initialization of the memory is described in section 3.2.2. The GRASP method is also combined with a local search phase, which is described in detail in section 3.2.3. We conclude this section with an overview of our approach in section 3.3.

### 3.1. Choosing the interventions to be subcontracted

The subcontracted interventions problem is tackled by using a preprocessing heuristics which selects interventions to be excluded. These interventions are cleaned from the problem once for all: the heuristics used is based on the minimum number of technicians required for each intervention ($mintec(I)$) and the duration of the interventions ($T(I)$). The first phase consists in computing a weight $\omega_I$ for each intervention $I$ so that $\omega_I = mintec(I) \times T(I)$. Let $\Omega_t$ be the set of indexes of technicians and $x \in \{0,1\}^{|\Omega_t|}$ a vector of decision variables. 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 associated with $I$:

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

Let $\Omega_I$ be the set of indexes of interventions and $x \in \{0,1\}^{|\Omega_I|}$ be the decision variable vector such that $x_I = 1$ if $I$ is subcontracted and 0 otherwise. In the second phase, we have to find a subset of interventions $\Gamma$ to be subcontracted so that $\sum_{x_I \in \Gamma} w_I x_I$ is maximum and the total cost does not exceed the total budget $A$. The problem of finding this subset is a precedence constrained knapsack problem [7]. Let (PCKP) be this problem and $S(x_I, x_{I'}) = 1$ if intervention $I'$ must start after the completion of intervention $I$ and 0 otherwise. The (PCKP) can be stated as follows:

$$PCKP \begin{cases} \text{Maximize } \sum_{I \in \Omega_I} w_I x_I \text{ subject to,} \\ \sum_{I \in \Omega_I} cost(I) \cdot x_I \leq A, \\ x_I \leq x_{I'} \quad \forall I, I' \in \Omega_I \mid S(x_I, x_{I'}) = 1 \\ x_I \in \{0,1\} \quad I \in \Omega_I \end{cases}$$

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 subcontracted, while the total cost does not exceed the maximal available budget $A$.

## 3.2. GRASP-based algorithm

When the subcontracted interventions have been deleted from the original problem, the subproblem composed by the remaining interventions is solved by a GRASP algorithm [4].

The terminology GRASP refers to a multi-start metaheuristics for combinatorial optimization. To be more precise, it refers to a class of procedures in which randomized greedy heuristics and local search techniques are employed. GRASP has been applied to a wide range of combinatorial optimization problems such as scheduling [12], routing [1], graph theory [9], assignment problems [6], etc. The reader can refer for instance to [10] or [5] for complete annotated bibliographies.

A classical GRASP implementation generally repeats the following scheme in three steps until a stopping condition is satisfied:

– Generate a feasible solution with a greedy randomized algorithm.

– Apply a local search to the previous solution.

– Update the best solution of the search.

The stopping condition can be a maximum number of iterations or a limited CPU time for instance.

GRASP has already been used for solving a problem in a similar context in [12]. In this paper, the authors developed a greedy algorithm, a local search method and a GRASP algorithm for solving a field technician scheduling problem. They showed that the GRASP method led to the best results despite an important increase of the execution time. However they implemented a parallel version of their algorithm to offset this drawback.

In the next section we first present the greedy algorithm used to generate a feasible solution. We also explain how the memory is updated dynamically during the search.

### 3.2.1. *Using GRASP to find a feasible solution*

The greedy algorithm is an important part of the GRASP implementation since it provides the method with many feasible solutions. In this section we give a general description of the mechanism of our greedy approach. Let us recall that the greedy algorithm only considers the non-subcontracted interventions that are called the *candidates*.

*Selecting a candidate*

Initially, the weight of a candidate is fixed to the coefficient of its priority in the objective function for interventions of priority 1, 2, 3 and we arbitrarily fix a coefficient of 1 for the interventions of priority 4. Thus, candidates of priority 1 (respectively 2, 3) have a weight of 28 (resp. 14, 4) and candidates of priority 4 have a weight of 1. The criterion of the greedy algorithm consists in selecting the candidate with the larger weight. The use of random allows the algorithm to decide which candidate to choose when two or more candidates have the same weight. Note that a given candidate cannot be scheduled if one of its predecessors at least is not scheduled.

Then, for a given candidate, the greedy algorithm attempts to assign it according to the following three criteria: (1) the earliest day possible; (2a) the team which requires the less additional technicians to perform the intervention; (2b) the minimum starting time possible. The process is repeated until all the candidates are scheduled. We next explain how the algorithm respects these three criteria.

*Computing the earliest day possible*

When trying to insert the current candidate $I$ in the solution, the condition (1) looks for adding it as soon as possible to limit the increase of the last day in the schedule. The first step consists in computing the minimum starting date $(d(I), s(I))$ of $I$. This is achieved by searching for the maximum day ($d_{max}$) among all its predecessors, then taking the maximum ending time $s(I_{max})+T(I_{max})$ such that $I_{max} \in Pred(I)$ and $I_{max}$ is scheduled on day $d_{max}$. If $s(I_{max})+T(I_{max})+T(I) > H_{\max}$ then $d(I) = d_{max} + 1$ otherwise $d(I) = d_{max}$.

*Computing the minimum required number of technicians for inserting a candidate*

Criteria (2a) and (2b) have to take into account the available technicians on $d(I)$ and the existing teams of technicians on $d(I)$. To respect the criterion (2a), a first phase consists in computing the number of technicians needed to construct a new team for $I$.

The second phase of the method consists in considering all the existing teams, and in computing for each of them the minimum starting date for $I$. The algorithm checks if the skills required by the intervention $I$ are satisfied by a given team $\epsilon$. If it is the case, there is no need to add a technician to $\epsilon$. On the contrary, the minimum number of technicians to add to $\epsilon$ can be determined by scanning the list of the available technicians for $d(I)$ and by comparing their skills with the missing requirements. Let $techs^{\epsilon}(I)$ be the required number of technicians needed for assigning $I$ to team $\epsilon$ ($\epsilon = 0$ if a new team has to be created).

*Computing the minimum starting time*
The next step of the algorithm consists in computing the value of $s^{\epsilon}(I)$ if $I$ is affected to a given team $\epsilon$. This is possible by scanning the list of the current interventions of $\epsilon$ and checking when it is possible to insert $I$ without *enjambment* with the interventions already scheduled.

*Defining the priority between (2a) and (2b)*
The order of the criteria (2a) and (2b) depends of the value of $d(I)$ obtained previously. It also depends on the ending time of the last scheduled intervention with the same priority as $I$ for priorities 1, 2, 3 and to the ending time of the whole schedule for priority 4 (i.e. $t_i$, where $i$ designate the priority of the candidate $I$). If $d(I) \times H_{\max} + s(I) + T(I) < t_i$ then the condition (2a) is considered before the condition (2b). Otherwise the condition (2b) has the priority. That can be justified by the fact that if the schedule of $I$ leads to an increase of the ending time of the same priority as $I$ or an increase of the whole schedule, then it is better to try to minimize the deterioration by looking for the minimum starting time possible.

Hence to favor condition (2a) the algorithm chooses the team $\epsilon$ with the minimum value of $techs^{\epsilon}(I)$. If several teams have the same value, the chosen team is the one for which the value of $s^{\epsilon}(I)$ is minimum. On the contrary, if it is the condition (2b) that is favored, then the algorithm chooses the team $\epsilon$ with the minimum value of $s_I^{\epsilon}$, and the one with the minimum value of $techs^{\epsilon}(I)$ if there are at least two teams with the same value $s_I^{\epsilon}$.

*Using a permutation of the weights in the greedy algorithm*
Some preliminary experiments underlined that the choice of the weight of an intervention is not so trivial. Let us note $w(I)$ as the weight of intervention $I$. In the previous description, we suppose that $w(I)$ is equal to the coefficient of the priority of $I$ in the objective function.

That corresponds to apply the highest priority first order. The Figure 1 presents a solution generated by the greedy algorithm where the weights of the interventions are 28 for interventions of priority 1, 14 for priority 2, 4 for priority 3 and 1 for priority 4. This solution was obtained for one of the instances used in our computational experiments. In this figure, interventions of priority 1 are represented by red boxes, those of priority 2 are represented by green boxes, those of priority 3 are represented by yellow boxes and there is no intervention of priority 4. Each line represents a technician, with the first technician represented by the top line and the last technician represented 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. The objective value of this solution is 17820.

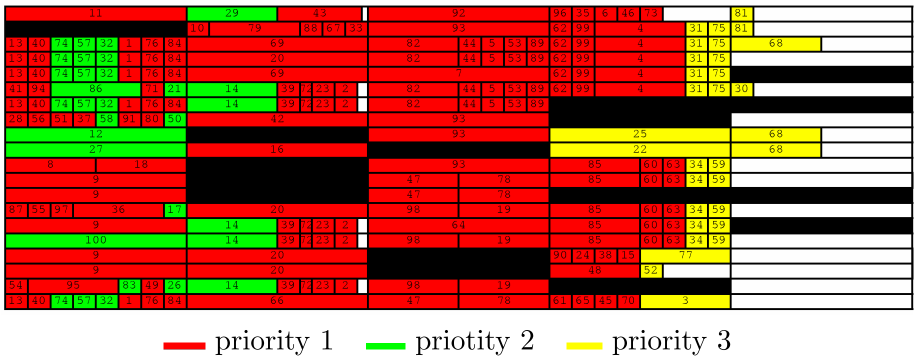

*Figure 1.* Solution with objective 17820 for instance data8 of instances set A

However it is possible to affect the weights of the interventions in a different way. Suppose that interventions of priority 4 have a weight of 28, those of priority 3 have a weight of 14, those of priority 1 have a weight of 4 and those of priority 2 have a weight of 1. That corresponds to use the permutation (4,3,1,2) of the weights. The Figure 2 gives the solution generated by the greedy algorithm with these weights. The objective value of this solution is 17355. Note that the weights of the interventions are obviously not used to evaluate a solution but only to guide the greedy algorithm.

This example clearly shows that for this instance, it should be better to fix a greater weight to the interventions of priority 3 and thus to use the permutation or priority order (4,3,1,2).

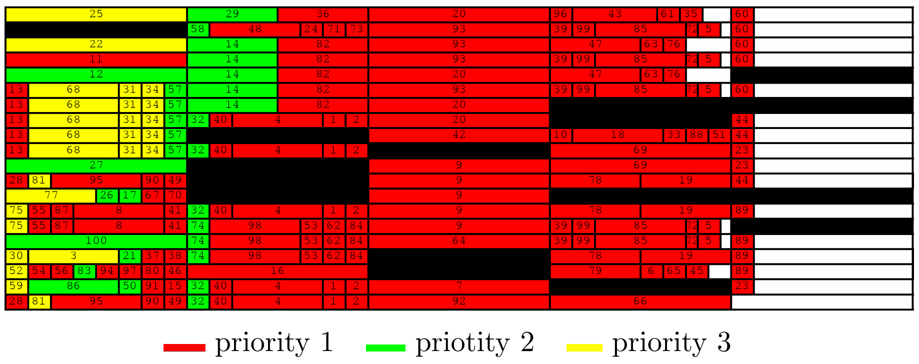

*Figure 2.* Solution with objective 17355 for instance data8 of instances set A

*Updating the weights*
Let us note as $w_p(I)$ the weight associated with intervention $I$ according to the priority order $p$. For example, let us suppose that $p = (3, 2, 1, 4)$, then $w_p(I) = 28$ if the priority of $I$ is 3, $w_p(I) = 14$ if the priority of $I$ is 2, etc. At the end of the greedy algorithm, the weights of the interventions are updated from the pieces of knowledge [11] associated with the solution generated. This update consists in adding the value $w_p(I)$ to the last interventions of each priority and to all their predecessors. Consequently, the greedy algorithm will attempt to schedule those interventions earlier at the next iteration. The objective si to decrease the ending time of each priority. The algorithm 1 illustrates this update procedure.

**Algorithm 1** update phase

```
Require: The list of interventions. A priority order $p$.
1: for each priority $prio$ do
2:     $I$ := last scheduled intervention of priority $prio$;
3:     $w(I) = w(I) + w_p(I)$;
4:     for each intervention $J \in Pred(I)$ do
5:         $w(J) = w(J) + w_p(I)$;
6:     end for
7: end for
```

Since it is difficult to determine a priori the *best* permutation of the weights, we define a first phase described in the next section that consists in initializing the memory by evaluating each of the 24 possible permutations.

3.2.2. *Initializing the memory*
The aim of this phase is to identify the best priority order that leads to the best behavior of the greedy algorithm. This is achieved heuristically by applying the greedy algorithm described above several times for each possible permutation of the weights (28,14,4,1) as follows:

- apply several times the greedy algorithm for each of the 24 possible permutations of the weights associated with the priorities.
- repeat the same process only for the 12 permutations that lead to the best solutions.
- repeat the same process only for the 6 permutations that lead to the best solutions.

When this phase is completed, we keep the *best* 2 permutations that obtain the best solutions and the GRASP method uses these 2 permutations. Consequently, when applying the greedy algorithm in GRASP, the weights of the interventions depend on the permutation used and are dynamically adapted subsequently.

3.2.3. *Improving Solutions by Local Search*
In this section, we describe our local search for improving solutions. In this problem, after fixing the assignment of interventions to the teams and the process order of interventions of each team, we can determine the feasibility of the schedule. Moreover, the optimal start times of interventions can be determined easily since the graph which represents the process order of interventions and the precedence constraints is a weighted directed acyclic graph [2]. 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.

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. In both phases, we consider only feasible moves, that is, if a neighborhood solution violates a constraint, the solution does not accepted. With respect to the assignment of technicians to each team, we maintain the minimal technicians for the assigned interventions. Hence available technicians of a day belong to either an empty team where no intervention is assigned or the teams whose technicians are minimal for the assigned interventions.

A swap operation exchanges the assignment and the order of two interventions. In this operation, reassignment of technicians is not considered since it is not a trivial problem. On the other hand, if the neighborhood solution is accepted in the move, technicians may be rearranged to preserve the minimality of teams by moving technicians to empty teams.

An insert operation removes an intervention and inserts it into another position. In this operation, after removing the intervention, it may remove technicians from the team where the intervention was assigned to maintain the minimality, and the team where the intervention is being inserted is merged with the empty team in order to make it easier to satisfy the constraints.

*Critical path phase*

The aim of the critical path phase is to decrease the ending times of each priority and that of the whole schedule (i.e., $t_1$, $t_2$, $t_3$ and $t_4$) simultaneously.

We define a critical path for a priority as a maximal sequence $(I_1, I_2, \ldots, I_l)$ of interventions such that intervention $I_l$ gives the ending time of the priority and each consecutive interventions $I_k$ and $I_{k+1}$ $(k = 1, \ldots, l-1)$ satisfy

$$d(I_k) = d(I_{k+1}) \text{ and } s(I_k) + T(I_k) = s(I_{k+1})$$

or

$$d(I_k) + 1 = d(I_{k+1}),\ \ s(I_{k+1}) = 0 \text{ and } s(I_k) + T(I_k) + T(I_{k+1}) > H_{\max}.$$

Intervention $I_{k+1}$ cannot be scheduled unless intervention $I_k$ is scheduled at earlier period. From the definition of a critical path, intervention $I_1$ has to be scheduled at earlier period in order to decrease the ending time of the priority. The local search finds a critical path for each priority and tries to schedule intervention $I_1$ at earlier period by searching the neighborhood.

*Packing phase*

In the packing phase, the algorithm schedules interventions more efficiently without increasing the ending time of each priority.

We consider a measure of the efficiency for team $\epsilon$ of day $j$. Let $J = \{I_1, I_2, \ldots, I_l\}$ be the interventions which are assigned to team $\epsilon$. Let $S(\epsilon, i, n)$ be the available number of technicians for level $n$ and domain $i$ for team $\epsilon$ (i.e., $S(\epsilon, i, n) = \sum_{t|e(t,j)=\epsilon} \left\lfloor \frac{C(t,i)}{n} \right\rfloor$). Let $N(J, i, n)$ be the required number of technicians for level $n$ and domain $i$ for a team to execute the interventions $J$ (i.e., $N(J, i, n) = \max_{I \in J} R(I, i, n)$ ). Let

$$W_{\text{skill}}(\epsilon, J) = \sum_{I \in J} \sum_{i,n} (N(J, i, n) - R(I, i, n)) T(I)$$

and

$$W_{\text{time}}(J) = H_{\max} - \sum_{I \in J} T(I),$$

and we estimate the wasted skill (resp., the wasted time) for team $\epsilon$ and interventions $J$ by $W_{\text{skill}}(\epsilon, J)$ (resp., $W_{\text{time}}(J)$). Then, we estimate the efficiency for assigning interventions $J$ to team $\epsilon$ by the function

$$f(\epsilon, J) = W_{\text{skill}}(\epsilon, J) + \alpha W_{\text{time}}(\epsilon),$$

where $\alpha$ is set to a very large value (i.e., the efficiency is estimated by a lexicographic order of time and skill).

In this phase, the local search estimates a solution by the summation of $f(\epsilon, J)$ for all team of all day, and it accepts a neighborhood solution in the move of the local search if the solution is feasible and it does not increase the current ending time of priority.

### 3.3. Overview of the approach

We summarize our approach in the Algorithm 2. This algorithm describe the three phases we exposed in the previous sections. The preprocessing heuristics that selects the interventions to be subcontracted is represented by the function *Greedy_Hired* on line 2. This function returns a sub-problem in which a part of the variables have been fixed (i.e. with several interventions deleted). Then, the second phase that consists in determining the 2 best permutations of the weights associated with the priorities is represented by the function *Initialize_Memory* called on line 3. This procedure provides $perm_1$ and $perm_2$, which correspond to the 2 permutations used in the GRASP, which is described between the lines 5 and 10. The GRASP method consists in repeating the application of the greedy algorithm with the 2 selected permutations, then updating the memory and finally applying the local search algorithm when the best solution is improved. The process stops when the allowed CPU time is passed.

## 4. Lower Bound

In this section, in order to evaluate the performance of the proposed algorithm, we consider a lower bound of the problem $P'$ where subcontracted interventions have already determined and have been excluded. To compute a lower bound for $P'$, we consider 8 relaxed problems with restricted interventions and compute a lower bound on the ending time of the schedule (which is often called a makespan) for each problem. Finally, the lower bound for $P'$ is computed by using them.

We consider the following problems

**Algorithm 2** Solve_TIST($PB$,MAX_CPU_Allowed)

```
Require: An instance $PB$ of TIST to solve; The total allowed CPU time.
 1: $Best\_Solution = \oslash$
 2: $SPB$ = Greedy_Hired($PB$);
 3: ($perm_1$, $perm_2$) = Initialize_Memory($SPB$);
 4: while MAX_CPU_Allowed is not reached do
 5:     $Solution_1$ = Greedy_Randomized_Construction($SPB$, $perm_1$);
 6:     $Solution_2$ = Greedy_Randomized_Construction($SPB$, $perm_2$);
 7:     Update the memory
 8:     Improve = Update_Best_Solution($Solution_1$,$Solution_2$,$Best\_Solution$);
 9:     if Improve = True then
10:         $Best\_Solution$ = Local_Search($SPB$, $Best\_Solution$);
11:     end if
12: end while
13: return $Best\_Solution$;
```

- $MSP(1)$ which has only priority 1 interventions and their predecessors for $P'$
- $MSP(2)$ which has only priority 2 interventions and their predecessors for $P'$
- $MSP(3)$ which has only priority 3 interventions and their predecessors for $P'$
- $MSP(1,2)$ which has priority 1 and 2 interventions and their predecessors for $P'$
- $MSP(2,3)$ which has priority 2 and 3 interventions and their predecessors for $P'$
- $MSP(3,1)$ which has priority 3 and 1 interventions and their predecessors for $P'$
- $MSP(1,2,3)$ which has priority 1, 2 and 3 interventions and their predecessors for $P'$
- $MSP(1,2,3,4)$ which has all interventions for $P'$.

Let $T_1$, $T_2$, $T_3$, $T_{1,2}$, $T_{2,3}$, $T_{3,1}$, $T_{1,2,3}$ and $T_{1,2,3,4}$ be lower bounds of their makespan respectively. Then the following problem gives a lower bound for $P'$:

$$\begin{array}{ll}\text{minimize} & 28t_1 + 14t_2 + 4t_3 + t_4 \\ \text{subject to} & T_1 \leq t_1,\ T_2 \leq t_2,\ T_3 \leq t_3 \\ & T_{1,2} \leq \max\{t_1, t_2\},\ T_{2,3} \leq \max\{t_2, t_3\},\ T_{3,1} \leq \max\{t_3, t_1\} \\ & T_{1,2,3} \leq \max\{t_1, t_2, t_3\} \\ & T_{1,2,3,4} \leq t_4,\end{array}$$

where $t_1$, $t_2$ and $t_3$ are the ending time for priority 1,2 and 3, respectively, and $t_4$ is the ending time of whole schedule. It is easy to see that any feasible solution (i.e., $t_1$, $t_2$, $t_3$ and $t_4$) must satisfy each constraint.

We propose three lower bounds for the makespan problem; The box lower bound is computed in a combinatorial way. The assignment lower bound is obtained by a linear programming problem which is a relaxation of $P'$. The trivial lower bound is derived from trivial conditions. We take the best lower bound from them and strengthen it by a post processing.

## 4.1. The Box Lower Bound

The box lower bound, which is a lower bound on the days (not time), is computed for each domain $i$ and level $n$, and the largest among them is adopted. The same method was proposed by Lodi, Martello and Vigo [8] for the two-dimensional level packing problem.

For each domain $i$ and level $n$, we consider a rectangle whose height is $R(I, i, n)$ and width is $T(I)$ for each intervention $I$. The area of a rectangle is the multiplication of the needed amount of skill and time for the intervention. In order to compute the box lower bound, the area of a rectangle may be partitioned. Let $Ad(j, i, n)$ be the number of technicians who can work at day $j$ and has a skill level $n$ in domain $i$. We arrange all rectangles by the height and split them by every $H_{\max}$ from the left and take the minimum rectangles which contain each $H_{\max}$ blocks. Figure 3 (a) shows such a situation and the dotted part is the empty space of the minimum rectangles. Each minimum rectangle means the work for the interventions which may be contained partially is executed by a team whose number of technicians is the height of the minimum rectangle. Next we build the minimum rectangles up and compute $\mu^* = \min\{\mu \in Z \mid H \leq \sum_{j=1}^{\mu} Ad(j, i, n)\}$ for the total height $H$. The $\mu^*$ is a lower bound on the days which are needed for the schedule. Figure 3 (b) shows the situation where $\mu^* = 2$.

Since this is a lower bound of days and we do not know the time, the gap can be at most $H_{\max}$ (1 day) in the sense of time. Hence we split a day into halves and apply the procedure assuming $\frac{H_{\max}}{2}$ is a

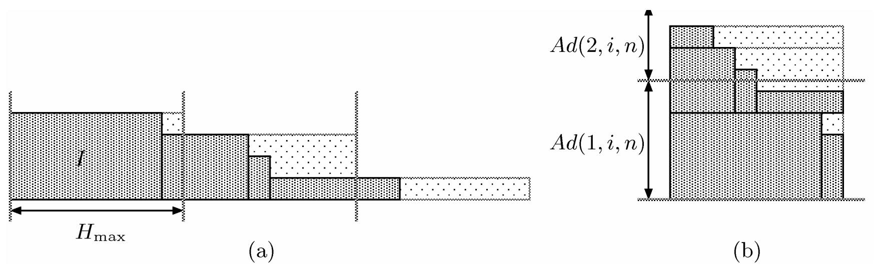

*Figure 3.* An example of the procedure for the box lower bound

day. We continue the process with $\frac{H_{\max}}{3}$, $\frac{H_{\max}}{4}$, ... , 1, and take the best lower bound among them.

## 4.2. The Assignment Lower Bound

In order to compute the assignment lower bound, we repeat guessing the number of days $\mu$ where the assignment lower bound lies and solving a linear programming problem corresponding to $\mu$ until the guess is a hit. We can know whether the guess is a hit or not from the optimal solution of the linear programming. In the case that the guess is not a hit, we can also know whether the guess is large or small.

For the number of days $\mu$ which we guess (i.e., we guess the makespan $M$ is in $[\mu H_{\max}, (\mu+1)H_{\max}]$), the available working time $U_\mu(t,M)$ until time $M$ for each technician $t$ can be computed easily. Note that $U_\mu(t,M)$ is represented as $M - lH_{\max}$ for some $l \leq \mu$ or 0. We consider the following linear programming ALP($\mu$):

$$\text{minimize} \quad M \tag{11}$$

$$\text{subject to} \quad \sum_{t \mid C(t,i) \geq n} x_{I,t} \geq R(I,i,n), \qquad \forall I, \forall i, \forall n \tag{12}$$

$$\sum_{I} T(I) x_{I,t} \leq U_\mu(t,M) \qquad \forall t \tag{13}$$

$$0 \leq x_{I,t} \leq 1 \qquad \forall I, \forall t, \tag{14}$$

where $x_{I,t}$ represents the assignment of $I$ to $t$. The feasible region of ALP($\mu$) is included by that of ALP($\mu+1$) and, hence, the optimal objective value becomes smaller for larger $\mu$. If the problem is infeasible or the optimal value $M^*$ of ALP($\mu$) is larger than $(\mu+1)H_{\max}$, we can see the assignment lower bound is larger. If the optimal value $M^*$ of ALP($\mu$) is smaller than $\mu H_{\max}$, we can see the assignment lower bound is smaller. The process is repeated until the guess hits and the last $M^*$ turns out to be the assignment lower bound.

### 4.3. The Trivial Lower Bound

The trivial lower bound is derived from the following necessary conditions. (1) The maximum execution time for all interventions is a lower bound. (2) The sum of execution time for a sequence of the interventions where each successive interventions has precedence relation is a lower bound. The maximum value of all such sequences can be computed in linear time.

### 4.4. Postprocess

A lower bound can be strengthened, because a makespan must be a values which consists of combination of $T(I)$ and $H_{\max}$. We compute all possible makespan by the dynamic programming, and take the least value larger than or equal to the given lower bound as a strengthened lower bound.

## 5. Experimental Results

Our algorithm was tested on data sets provided by *France Telecom* for the $5^{th}$ challenge of the French Society of Operations Research and Decision Analysis. There are three data sets available, each data set contains 10 instances with a different number of interventions, technicians, domains and levels. The first data set called data-setA does not consider the problem of subcontracted interventions. It contains instances from 5 to 100 interventions, from 5 to 20 technicians, from 3 to 5 domains and from 2 to 4 levels. The instances of the data-setB are much harder to solve and they does consider the problem of subcontracted interventions. This data-set contains instances from 120 to 800 interventions, from 30 to 150 technicians, from 4 to 40 domains and from 3 to 5 levels. Finally, the data-setX is the data-set on which the evaluation for the challenge ranking was based. It contains instances from 100 to 800 interventions, from 20 to 100 technicians, from 6 to 20 domains and from 3 to 7 levels.

We provide in the Table I the official results which were published on the website of the challenge. The computer used contains an AMD Processor of 1.8 GHz and 1 GB of DDR-RAM. The execution time was limited to 1200 seconds. The description of the data per column is the following: *inst.*: 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. The column GRASP has three values: *value*: the objective value of the GRASP-based algorithm, *LB*: the lower bound value without the subcontracted interventions and

Table I. Results obtained on the benchmarks provided by France Telecom

| | | | | | GRASP | | | Best objective | |
|---|---|---|---|---|---|---|---|---|---|
| inst. | int. | tec. | dom. | lev. | value | LB | gap. | value | gap |
| 1-setA | 5 | 5 | 3 | 2 | 2340 | 2265 | 3.2 | 2340 | 0 |
| 2-setA | 5 | 5 | 3 | 2 | 4755 | 4215 | 11.35 | 4755 | 0 |
| 3-setA | 20 | 7 | 3 | 2 | 11880 | 11310 | 4.79 | 11880 | 0 |
| 4-setA | 20 | 7 | 4 | 3 | 13452 | 10995 | 18.26 | 13452 | 0 |
| 5-setA | 50 | 10 | 3 | 2 | 28845 | 26055 | 9.67 | 28845 | 0 |
| 6-setA | 50 | 10 | 5 | 4 | 18870 | 17775 | 5.8 | 18795 | 0.39 |
| 7-setA | 100 | 20 | 5 | 4 | 30840 | 27405 | 11.13 | 30540 | 0.97 |
| 8-setA | 100 | 20 | 5 | 4 | 17355 | 16166 | 6.85 | 16920 | 2.50 |
| 9-setA | 100 | 20 | 5 | 4 | 27692 | 25618 | 7.48 | 27692 | 0 |
| 10-setA | 100 | 15 | 5 | 4 | 40020 | 35405 | 11.53 | 38296 | 4.3 |
| | | | | | | **Average** | **9.01** | | **0.81** |
| 1-setB | 200 | 20 | 4 | 4 | 43860 | 38385 | 12.48 | 34395 | 21.58 |
| 2-setB | 300 | 30 | 5 | 3 | 20655 | 16605 | 19.6 | 15870 | 23.16 |
| 3-setB | 400 | 40 | 4 | 4 | 20565 | 17460 | 15.09 | 16020 | 22.1 |
| 4-setB | 400 | 30 | 40 | 3 | 26025 | 19035 | 26.85 | 25305 | 2.76 |
| 5-setB | 500 | 50 | 7 | 4 | 120840 | 106290 | 12.04 | 89700 | 25.76 |
| 6-setB | 500 | 30 | 8 | 3 | 34215 | 24450 | 28.54 | 27615 | 19.28 |
| 7-setB | 500 | 100 | 10 | 5 | 35640 | 28470 | 20.11 | 33300 | 6.56 |
| 8-setB | 800 | 150 | 10 | 4 | 33030 | 32820 | 0.63 | 33030 | 0 |
| 9-setB | 120 | 60 | 5 | 5 | 29550 | 26310 | 10.96 | 28200 | 4.56 |
| 10-setB | 120 | 40 | 5 | 5 | 34920 | 32790 | 6.09 | 34680 | 0.68 |
| | | | | | | **Average** | **15.24** | | **12.64** |
| 1-setX | 600 | 60 | 15 | 4 | 181575 | 140025 | 22.88 | 151140 | 16.76 |
| 2-setX | 800 | 100 | 6 | 6 | 7260 | 6840 | 5.78 | 7260 | 0 |
| 3-setX | 300 | 50 | 20 | 3 | 52680 | 49650 | 5.75 | 50040 | 5.01 |
| 4-setX | 800 | 70 | 15 | 7 | 72860 | 59560 | 18.25 | 65400 | 10.23 |
| 5-setX | 600 | 60 | 15 | 4 | 172500 | 126465 | 26.68 | 147000 | 14.78 |
| 6-setX | 200 | 20 | 6 | 6 | 9480 | 6180 | 34.81 | 9480 | 0 |
| 7-setX | 300 | 50 | 20 | 3 | 46680 | 45000 | 3.59 | 33240 | 28.79 |
| 8-setX | 100 | 30 | 15 | 7 | 29070 | 20590 | 29.17 | 23640 | 18.67 |
| 9-setX | 500 | 50 | 15 | 4 | 168420 | 101985 | 39.44 | 134760 | 19.98 |
| 10-setX | 500 | 40 | 15 | 4 | 178560 | 99705 | 44.16 | 137040 | 23.25 |
| | | | | | | **Average** | **23.05** | | **13.74** |

*gap*: the gap in percentage between the lower bound and the objective value. The column Best objective has two values: *value*: The best objective value found among all the challengers solutions and *gap*: the gap in percentage between this best objective and our objective value. *Average*: The average of the gaps.

There was a total of 17 teams participating to the final stage of the challenge and our algorithm was ranked to the fourth position. The evaluation was based only on the data-setX. The ranking was made on the average value of the gaps between our solution and the best solution among all the challengers for each instance.

Table I shows that the gap between the upper bound provided by the GRASP and the lower bound is quite constant for all the instances except for some instances of the data-set X. The experimentations we have made later showed us that our algorithm never reached the improving phase embedding the local search for some of these instances. This might be a good reason why the gap in this case is bigger than the one for the others data sets.

A point that can be extracted from those experiments is that the choice of the subcontracted interventions is not optimum. Indeed, some values of our lower bounds of the remaining problem are under the best solutions among all the challengers (upper bounds of the whole problem). We made some new experiments which confirm this point of view: for the instance 9 of the data-set B, we selected a different set of subcontracted interventions and then we executed the same algorithm. It appeared that it founds a lower bound of 25695 instead of 26310 and an objective value of 27960 instead of 29550 whereas the best solution found among all the challengers is 28200.

## 6. Conclusion

In this paper, we presented a Technician and Intervention Scheduling Problem for Telecommunications and gave a mathematical formulation.

We proposed a heuristic algorithm based on three main stages: (1) the fixing of some variables which is done by the "knapsack ratio heuristic" for subcontracted interventions, (2) the initializing of the memory (weights allocated to the interventions) which is carried out by the search of the best orders to insert interventions and (3) the greedy randomized adaptive search procedure and the local search embedded which seek to improve the initial solutions by updating the initial weights of interventions.

We gave lower bounds which confirm the effectiveness of our approach. The experimentations have clearly showed that the choice of the "knapsack ratio heuristic" is the main weak spot of this approach. Nevertheless, as we pointed out with the instance 9 of the data-set B, a different fixing heuristics can provide better solutions.

Finally, the greedy adaptive memory algorithm shows to be a promising tool for solving this problem especially if we improve the first step of our global approach. That is the aim of further work we are planning to conduct.

## References

1. J. B. Atkinson. A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operational Research Society*, 49:700–708, 1998.
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* The MIT Press, 2 edition, 2001.
3. P.-F. Dutot and A. Laugier. Technicians and interventions scheduling for telecommunications(ROADEF challenge subject). Technical report, France Telecom R&D, 2005.
4. T. A. Feo and M. G. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.
5. P. Festa and M. G. C. Resende. GRASP: An annotated bibliography. In C. C. Ribeiro and P. Hansen, editors, *Essays and surveys in metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.
6. C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11:198–204, 1999.
7. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.
8. A. Lodi, S. Martello, and D. Vigo. Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization*, 8:363–379, 2004.
9. M. G. C. Resende and C. C. Ribeiro. A GRASP for graph planarization. *Networks*, 29:173–189, 1997.
10. M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.
11. É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135:1–16, 2001.
12. J. Xu and S. Y. Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7:495–509, 2001.