Roadef 2007 Challenge

Jean-François Cordeau, Gilbert Laporte, Federico Pasin, Stefan Ropke

HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

1 Introduction

This extended abstract describes our solution approach to the *Technicians and interventions scheduling for telecommunications problem* (TISTP) described in [1]. The document is structured as follows. Section 2 describes a construction algorithm while Section 3 describes an adaptive large neighborhood search algorithm for solving the problem. Section 4 contains computational results.

2 Construction algorithm

In order to describe the construction algorithm some definitions are necessary. We define the *heaviness* H_j of intervention j as follows :

$$H_j = \sum_{i \in D, n \in L} R(j,i,n)^{\alpha},$$

where D is the set of competence domains, L is the set of skill levels, $R(j,i,n)$ is the number of technicians of level n in domain i needed to perform task j, and α is a parameter.

The *distance* $d(j_1, j_2)$ between two interventions j_1 and j_2 is defined as

$$d(j_1,j_2) = \frac{\sum_{i \in D, n \in L} |R(j_1,i,n) - R(j_2,i,n)|}{\left(\sum_{i \in D, n \in L} (R(j_1,i,n) + R(j_2,i,n))\right)^{\beta}},$$

where β is a parameter. The distance measures how the two jobs are related in terms of required technician skills.

The *criticality* c_j of intervention j is defined as

$$c_j = w_{p_j} \cdot t_j + \sum_{k \in \sigma(j)} (w_{p_k} \cdot t_k),$$

where p_j is the priority of intervention j, t_j is the execution time of intervention j and w_p is the weight associated to priority p ($w_1 = 28$, $w_2 = 14$, $w_3 = 4$, $w_4 = 1$). Finally, $\sigma(j)$ is the set of tasks that must succeed task j (not necessarily direct successors).

The construction heuristic assembles teams and assigns tasks to teams one day at a time. In order to plan a day the heuristic uses a two-phase approach. In the first phase teams are constructed and a single task is assigned to each team. In the second phase further tasks are assigned to the created teams.

2.1 Phase 1

In phase 1 we use the procedure outlined in Figure 1 (recall that this procedure operates on a specific day).

The first phase creates a number of teams. Each team performs exactly one intervention. The first phase usually leaves a few technicians unassigned (depending on the parameter ρ). These technicians are used in the second phase to supplement existing teams.

2.2 Phase 2

Phase 2 assigns interventions to the teams created in phase 1. As in phase 1, the procedure operates on a specific day. The procedure is outlined in Figure 2.

The idea in step 2(a) is to assign interventions to teams that have approximately the right skills. Interventions with high criticality are prone to be inserted before interventions with low criticality due to the last term in the definition of $f(j,\tau)$.

1. Let U be the set of unassigned interventions. Let $A = \emptyset$ be the set of interventions served on this day. Let T be the set of of technicians available on this day.
2. Repeat until the number of unassigned technicians is less than $\rho|T|$ where $\rho \leq 1$ is a parameter.
   (a) Select the intervention $j \in U$ that maximizes
   $$\gamma_1 H_j + \gamma_2 c_j + \gamma_3 \sum_{k \in A} d(j,k),$$
   where γ_1, γ_2 and γ_3 are parameters.
   (b) Using the available technicians, create in a greedy fashion a team satisfying the requirements of intervention j. If this is not possible then remove j from U and repeat step 2.
   (c) Let j be served by the newly created team. $A = A \cup \{j\}$. $U = U \setminus \{j\}$

Fig. 1. Construction phase I

1. Let U be the set of unassigned interventions.
2. Repeat until all technicians have been assigned to a team.
   (a) Repeat until no more interventions can be assigned to teams.
      i. For each unassigned intervention $j \in U$ and each team τ on the current day, calculate
      $$f(j,\tau) = \delta_1 f_1(j,\tau) + \delta_2 f_2(j,\tau) - \delta_3 c_j.$$
      $f_1(j,\tau)$ calculates how many of the currently unassigned technicians must be added to team τ in order to perform intervention j, $f_2(j,\tau)$ calculates the skills that are wasted in team τ by assigning intervention j to team τ. δ_1, δ_2 and δ_3 are parameters.
      ii. Select intervention j and team τ such that $f(j,\tau)$ is minimized. Add extra technicians to team τ if necessary and let j be served by this team. $U = U \setminus \{j\}$.
   (b) If there are unassigned technicians then create an extra team on the day consisting of one of the unassigned technicians.

Fig. 2. Construction phase II

3 Adaptive large neighborhood search algorithm

This section presents an adaptive large neighborhood search (ALNS) metaheuristic for the TISTP. The ALNS heuristic was proposed in [3] and [2] and is an extension of the Large Neighborhood Search (LNS) heuristic proposed in [4,5]. The ALNS heuristic is outlined in Figure 3.

Key components in an ALNS heuristic are the *destroy* and *repair* methods. Destroy methods destroy part of the solution while the repair method repairs partially destroyed solutions. For the TISTP we let destroy methods unassign a number of interventions from the current solution. After unassigning interventions superfluous technicians are removed from their teams. If a team does not serve any intervention after the destroy operation, it is removed. The repair methods for the TISTP take as input a partial solution along with a set of unassigned interventions and for each day a set of unassigned technicians. The unassigned interventions are served by existing teams or by new teams created from unassigned technicians.

In an ALNS heuristic one defines several destroy and repair methods. The ALNS framework choses which pair of destroy/repair methods to use using a set of scores $\{\pi_j\}$ that are updated throughout the search. Destroy/repair methods that are performing well by finding new solutions get higher scores compared to methods that do not generate any improvements. In this way the algorithm is able to adapt dynamically to different instances. For details see [3,2].

In step 2.c the algorithm decides whether or not to accept the generated solution. A simple acceptance criterion is to accept only improving solutions, but past experience shows that a simulated-annealing criterion accepting deteriorating solutions with a certain probability often improves performance.

1. Construct a feasible solution x ; set $x^* = x$
2. Repeat
   (a) Choose a destroy method M^- and a repair method M^+ using roulette wheel selection based on previously obtained scores $\{\pi_j\}$
   (b) Generate a new solution x' from x using the chosen destroy and repair methods
   (c) If x' can be accepted then set $x = x'$
   (d) Update scores π_j of M^- and M^+
   (e) if $f(x') < f(x^*)$ then set $x^* = x'$
3. until stop criteria is met
4. return x^*

Fig. 3. Adaptive Large Neighborhood Search

3.1 Destroy methods

This section describes the destroy methods that we have implemented. The destroy methods have two common features :

1. If an intervention j with $\sigma(j) \neq \emptyset$ is removed, then so are all of the interventions from $\sigma(j)$. This simplifies the design of repair methods.
2. The number of interventions to remove is not fixed, but is chosen randomly in the interval $[l, u]$ each time a destroy method is invoked. l and u are dependent on the instance size.

Random destroy The random destroy method selects the interventions to remove at random.

Related destroy The related destroy method selects one intervention to remove at random initially. This intervention initially forms the set S of removed interventions. The method proceeds iteratively by choosing a random intervention from the set S and an assigned intervention j that is closely related to the chosen intervention. Intervention j is removed from the solution and added to the set S. We use the distance measure $d(j_1, j_2)$ defined in section 2 to express relatedness between interventions. The idea behind this method is that it should be easy to exchange interventions that are closely related.

Last-intervention destroy For this method a fraction of the interventions to be removed are chosen among the interventions that contribute to the objective (the last served interventions of priority 1 to 3 and the last served interventions in general). Two variants of the algorithm are implemented, depending on how the remaining interventions are selected. In the first variant the remaining interventions are selected at random, while in the second variant the remaining interventions are selected to be related to the objective-contributing interventions. The purpose of this method is to modify the part of the solution that really affects the obejctive function value by trying to serve some of the last interventions earlier in the schedule.

Whole-route destroy The whole-route destroy method selects a day at random and removes all interventions from two of the teams on this day. As long as the limit on the number of removed interventions has not been reached the algorithm repeats this process. This methods frees technicians on the chosen days and allows for the construction of new teams.

3.2 Repair heuristics

As repair heuristic we use the construction algorithm described in section 2. Two variants are considered : one that uses a fixed parameter setting and one that selects a random parameter setting by drawing a random value for each parameter within a predefined interval.

4 Computational results

Table 1 compares four heuristics for the problem. The construction heuristic described in section 2, a version of the construction heuristic that is run repeatedly with different parameter settings chosen at random, the ALNS algorithm with an acceptance criterion that only accepts improving solutions and an ALNS algorithm that use a simulated annealing acceptance criterion. The two first columns in the table report the instance name along with the number of interventions in the instance. The next column shows the best objective obtained during many experiments, the next four columns show the objective obtained with the four methods just described and the last four columns show how the solution obtained by the four heuristics compares to the best known solutions. The last row sums column 3 to 7 and averages column 8 to 11.

The construction heuristic spends less than one second on each instance while the three other heuristics are allowed to spend 10 minutes on each instance. The experiments were performed on an AMD Opteron 250 computer that according to our experiments is roughly twice as fast as the computer used in the competition.

The results show that using the ALNS method clearly is superior to repeatedly executing the construction heuristics with different paramter settings. It also shows that using a simulated annealing acceptance criterion results in better overall performance compared to accepting only improving solutions.

Table 2 shows the results obtained with the final version of the ALNS algorithm with further refinements. These results were also obtained by running the algorithm for 10 minutes on an Opteron 250 computer.

			Objective				Percentage above best objective			
name	n	Best	Constr. method	Rand. Constr.	ALNS steepest	ALNS SA	Constr. method	Rand. Constr.	ALNS steepest	ALNS SA
A_data1	5	2340	3690	2550	2340	2340	57.7	9.0	0.0	0.0
A_data2	5	4755	4755	4755	4755	4755	0.0	0.0	0.0	0.0
A_data3	20	11880	16950	14700	13710	11880	42.7	23.7	15.4	0.0
A_data4	20	13452	17520	15840	13620	13452	30.2	17.8	1.2	0.0
A_data5	50	33480	42495	36180	34740	33480	26.9	8.1	3.8	0.0
A_data6	50	18870	26775	23640	19635	19710	41.9	25.3	4.1	4.5
A_data7	100	30660	36630	33900	30960	30740	19.5	10.6	1.0	0.3
A_data8	100	19500	23280	22440	20040	19995	19.4	15.1	2.8	2.5
A_data9	100	28020	40290	34284	29700	28020	43.8	22.4	6.0	0.0
A_data10	100	38636	50640	45180	40860	38636	31.1	16.9	5.8	0.0
B_data1	200	48720	93120	68250	50880	50400	91.1	40.1	4.4	3.4
B_data2	300	21240	37425	26880	23910	21240	76.2	26.6	12.6	0.0
B_data3	400	24360	47250	36750	26340	24360	94.0	50.9	8.1	0.0
B_data4	400	36060	63705	45345	40425	37545	76.7	25.7	12.1	4.1
B_data5	500	119520	148620	124380	121440	122160	24.3	4.1	1.6	2.2
B_data6	500	34575	47850	42150	35205	35130	38.4	21.9	1.8	1.6
B_data7	500	38460	43680	38640	38940	41520	13.6	0.5	1.2	8.0
B_data8	800	34800	62160	52080	34800	40680	78.6	49.7	0.0	16.9
B_data9	120	29160	34620	32490	31680	29160	18.7	11.4	8.6	0.0
B_data10	120	39720	50820	45240	42960	39720	27.9	13.9	8.2	0.0
		628208	892275	745674	656940	644923	42.6	19.7	4.9	2.2

Tab. 1. Comparison of heuristics

Références

1. P.-F. Dutot, A. Laugier, and A.-M. Bustos. *Technicians and Interventions Scheduling for Telecommunications*. France Telecom R&D, August 2006.
2. D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 2007. Forthcoming.

Data set A		Data set B	
Name	Objective	Name	Objective
A_data1	2340	B_data1	38925
A_data2	4755	B_data2	17700
A_data3	11880	B_data3	17190
A_data4	13452	B_data4	27480
A_data5	33480	B_data5	107280
A_data6	19635	B_data6	31440
A_data7	30540	B_data7	34620
A_data8	20100	B_data8	33360
A_data9	28020	B_data9	31680
A_data10	38580	B_data10	38040

Tab. 2. Final results

3. S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4) :455–472, 2006.
4. P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, APES Group, Department of Computer Science, University of Strathclyde, 26 Richmond Street, Glasgow, Scotland, July 1997.
5. P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431, 1998.