Heuristic algorithm for problem of Technicians and Interventions Scheduling for Telecommunication
ROADEF 2007 Challenge Problem

Grzegorz Pawlak[1], Sławomir Bąk[2], Paweł Lichocki[3], et Wojciech Mruczkiewicz[4]

[1] Poznan University of Technology, ul. Piotrowo 2, 60-965 Poznan, Poland
grzegorz.pawlak@cs.put.poznan.pl
[2] Poznan University of Technology
sbak@skno.cs.put.poznan.pl
[3] Poznan University of Technology
plichocki@skno.cs.put.poznan.pl
[4] Poznan University of Technology
wmruczkiewicz@skno.cs.put.poznan.pl

1 Introduction

The subject of the ROADEF 2007 Challenge is a complex problem in the field of scheduling algorithms. The problem statement, available at the [1], is briefly described in Chapter 2 for this paper completeness. In Chapter 3 the algorithm developed is explained in details. In Chapter 4 results obtained and the description of the computation experiment are presented. In Chapter 5 final conclusions and future views are stated.

2 Problem description

There is given the set $\mathcal{I}$ of n interventions and the set $\mathcal{T}$ of m technicians and unlimited number of days. The technicians can group together and perform the interventions. Every technician has some skills. Skills are divided into domains and each domain is characterized by one parameter – level. The level is a non-negative integer value. To perform an intervention there is required a specific number of technicians with certain skill level in every domain. Thus there is a need to form groups of technicians. Technician can cover many domains at once. They satisfy the level requirements lower or equal to their level.

Groups of technicians are formed for the period of one day and cannot be destroyed during that time. Interventions have the duration parameter, the amount of time that it takes to intervene. Every day has a limited number of time units and interventions have to start and stop within single day. When particular group of technicians perform the intervention than it cannot perform any other intervention (interventions cannot overlap). Interventions can be performed only be one group and cannot be stopped. Every intervention has also the precedence constraints (the list of interventions that have to be performed before start of the intervention) and priority.

The goal is to schedule interventions in a such way that total cost of the schedule is minimized. The cost of the schedule is given by the formula

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

where t_1, t_2, t_3 and t_4 are appropriately ending times of the last scheduled interventions of priority 1, 2, 3 and the ending time of the last intervention regardless the priority. This means that an intervention with lower priority has a higher cost. Every intervention has also some cost and there is some amount of money that can be used to abandon interventions. Abandoned interventions are not included in the schedule.

3 Algorithm description

3.1 Jobs pool

Algorithm 1 The main loop.

```
while pool not empty do
  $intervention \leftarrow$ null
  if morning of the day then
    $intervention \leftarrow$ get next biggest intervention from pool
  else
    $intervention \leftarrow$ get next smallest intervention from pool
  end if
  if $intervention =$ null then
    add new date to schedule
    reset pool
  else
    add intervention to schedule
  end if
end while
```

Entire scheduling algorithm basis on the idea of the interventions pool. It is a container which keeps track on interventions that may be scheduled (those with fulfilled all precedence dependencies) and enables getting next biggest and smallest interventions in the sense of a specified metric. Pool remembers what intervention it has returned last time (both for get max intervention and get min intervention) and always returns the next one. When all available interventions were returned it returns nothing (null) and it is necessary to reset the pool. The basic comparator of the interventions (used by get next max intervention and get next min intervention) works according to the idea :

```
if real priority of $intervention_1 <$ real priority of $intervention_2$ then
  return $intervention_1$ is bigger
end if
if value of $intervention_1 >$ value of $intervention_2$ then
  return $intervention_1$ is bigger
end if
if number of all successors of $intervention_1 >$ number of all successors of $intervention_2$ then
  return $intervention_1$ is bigger
end if
return interventions are equal
```

The real priority is defined as the maximum of the priorities for all successors of the intervention and the intervention's priority. Furthermore, a successor is defined as both direct and indirect. Last, value of the intervention is its "size" which basically may be defined as sum of all its levels in all domains,

$$\sum_{i,n} R(I,i,n).$$

where $R(I,i,n)$ is required number of technicians of level n in competence domain i required to complete intervention I. Later minor changes were introduced to this scheme by using different metrics to compute intervention's value, for example using hyperbolic tangent function, which gave better results in some (but not all) cases :

$$\sum_{i,n} \tanh\left(R^2(I,i,n)\right).$$

However, the question what is the best way to calculate interventions' values remains unanswered.

3.2 The idea of the day's morning

At the begging (so-called morning) of the day it is empty (or almost empty) and all (or almost all) technicians are not working. This means it is relatively easy than to add a new intervention to the schedule. One may use it to his advantage by adding biggest (in the sense described above) interventions. When morning passes and it gets harder to add new interventions the smallest ones are added (hoping they are small enough to fit in large number). It is difficult to find a good definition of the morning (for example how many interventions must be scheduled in a day to say the morning is over - for different instances this value differs). For this reasons in the final algorithm the above main scheme is performed multiply times for different values of so-called border of the morning.

3.3 Adding new intervention to the day's schedule

Each day keeps track on what technicians are available (i.e. not yet disposed to a team). This is done in a similar way that global interventions pool works. Every day is defined as a set of pairs <interventions' sequence, technicians' set>, which will be called subdays. Adding new intervention to the day's schedule works as fallows :

```
if not simple_add intervention to the day then
  if not create_add intervention to the day then
    if not expand_add intervention to the day then
      return failed
    end if
  end if
end if
return success
```

Simple_add tries to add intervention to the one of the existing subdays without making any changes in its technicians' set. Create_add tries to create a new subday using technicians available at the moment. Expand_add tries to extend existing subday with additional available technicians (to make it possible to schedule the given intervention). Each procedure analyze existing subdays in the numeric order (first the first subday, then the second one, etc., the number of the subday is equivalent to the number of the team in the problem definition). So this mechanism prefers the non-invasion way of scheduling, then it prefers to create an entirely new team and only if it fails in both cases it will try to expand existing teams. However, it lacks the ability of choosing the best possible alternative ; perhaps sometimes its better to expand existing team than creating a new one. Implementing this feature is perhaps one of the more important tasks for the future (however it will require also a way to "judge" the teams). Of course the intervention is not added if its time dependencies cannot be fulfilled.

3.4 Adding new intervention to the existing subday or to newly created one

The simple_add is an obvious way of adding new interventions, since it is non-invasion. The other two methods require more detailed description. The way of expanding the subday and creating a new one is actually one and the same. Creating a new subday is like expanding it from a subday with an empty set of technicians and no interventions scheduled. We have distinguished both methods in order to underline the fact of preferring creating new teams over changing existing ones.

```
while not team can perform an intervention do
  temporarily add best technician from all available at the moment
end while
if team can perform an intervention then
  commit changes to the team
  delete all not needed technicians from the team
  return success
end if
return failed
```

Few words of commentary is needed. The best technician that will be chose is the one who will maximize the closeness of team and interventions to be performed by it. Closeness is defined as a relation of the team's value and intervention's value (where value is defined as stated earlier). Please note that it is possible to make similar metrics for technicians, teams and sequences of interventions. For technicians it is the same as for interventions, for a team one must first (before calculating the value) add all levels in all domains and for a sequence of interventions one must find a maximum level in all domains.

Overall this is a greedy approach, the best (at the moment) technician is chosen and than he expands the team. This action is repeated until the team can perform a intervention or there are no more technicians available. Let assume it was successful, than so-called teams squeezing may be performed. It means deleting technicians that are no longer required in the team (all interventions may be performed without them), which may happen in case of expanding an existing (not empty) subday. Please notice that in general the result of this is dependent on the order of technicians analysis. Currently it is done in a numerical order, but estimating the importance of each technician and deleting them from the team in that order may bring further improvements to the algorithm.

3.5 Solution improvement techniques

Given any correct schedule possible become search for time gaps (the time when technicians are not working) that are big enough to place intervention that is scheduled at a later time into that gap. This kind of move decreases intervention end time. The implementation iterates over every intervention and builds a list of every possible move that decreases intervention end time. The algorithm must be carefull to preserve precedence constraints and skill requirements. From selected possibilities the move that decreases the end time of the last intervention of the lowest priority is chosen. If there is no such intervention than any possible move is chosen. This technique is used on the final schedule as many times as it can make a move. The final schedules do not have many gaps needed to move interventions but there are cases when this technique decreases total cost of the schedule.

This approach can be generalized to swap intervention with another intervention, not just move it into a time gap. We considered a few variations of this technique.

The goal of the first approach was to decrease the end time of low priority interventions (they cost more to schedule than high priority interventions). In a single algorithm step two interventions were selected in a random way. These interventions were swapped together. The selection process was checking for problem rules not to be violated after the swap. This procedure was repeated until no more swaps were possible. Another improvement was to perform swaps in a such way that some of the technicians could be released and used to form new team or to support another team. To do this, in the selection process were also considered swaps between interventions of equal priorities. We used also modification of the procedure that was swapping any kind of interventions to introduce an artificial delay. The schedule modified in this way could be then “repaired” using one of techniques described above.

3.6 Interventions abandoning

The first version of choosing the interventions to abandon took the interventions from the end of the schedule. This approach turned out to give a little improvement because of the frequent situation where very expensive interventions were pushed to the end.

In response to this a new approach was devised. The main idea is to select the task for abandonment before the rest of the algorithm was run. The method selected the interventions with the greatest value of empirical measure defined as

$$\mu = \frac{\tanh\left(m^2\right)}{c}$$

where μ is the measure, m is the maximal level value in any domain in intervention skills requirements and c is cost. The interventions were abandoned until the cost of every remaining intervention was greater than the sum could be spared or there were no other interventions left.

4 Results

The short description of the results is presented on the figure 1. The computation was run on IBM-PC 1.8GHz machine with 1GB RAM. In the first four cases the test was not time limited and these algorithms finished before being stopped. Actually, the swapping algorithm could run

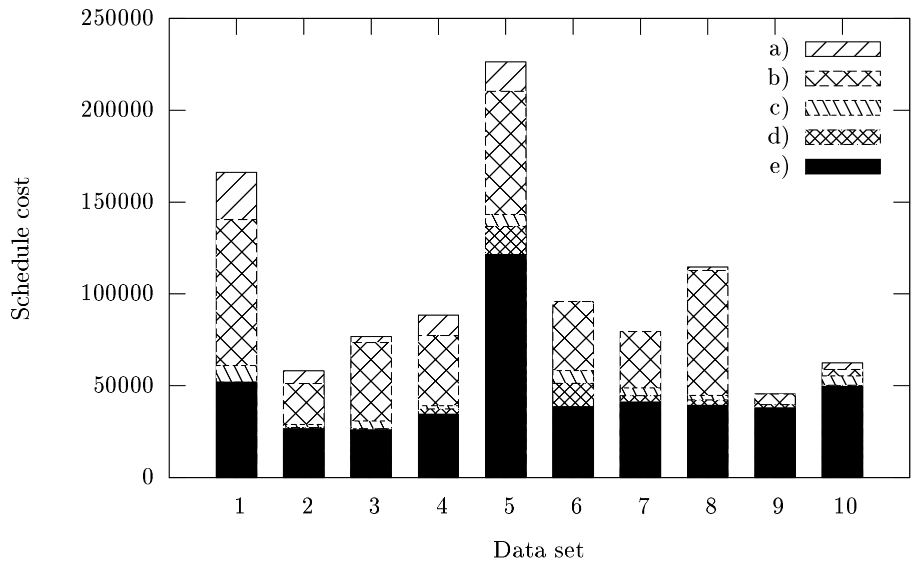

Fig. 1. Progress in schedule cost. a) no additional techniques, b) interventions abandonment enabled, c) swapping techniques enabled, d) initial solution algorithm starting many times with different parameters, e) final algorithm, all optimizations enabled.

indefinitely. The stopping condition was introduced and more than ten consecutive runs that do not improve schedule cost are not permitted. The tests where initial solution algorithm was starting many times with different parameters were limited by the fixed number of different parameter combinations allowed. The final results used time limited algorithm that tried to explore as many parameter combinations as possible. There are also additional stopping conditions applied when better solution is not expected. Thus only in test cases four through eight the algorithm was forced to stop because of the time limit. In most of cases there is a noticeable improvement in schedule cost after applying next techniques.

The results obtained for data set A and data set B is presented in tables 1 and 2

Instance	Cost
1	3540
2	4755
3	17040
4	15432
5	38700
6	25050
7	33960
8	24000
9	31680
10	48600

Tab. 1. Results for data set A.

5 Future views

Summarizing, the results were improved - in comparison of the first naive algorithm and its final version. The most visible uplifts of the schedule occurred after

1. taking into consideration not only interventions “size” but also priorities

Instance	Cost
1	51960
2	26625
3	25920
4	34500
5	121440
6	38655
7	41100
8	39360
9	37920
10	49680

Tab. 2. Results for data set B.

2. introducing "squeezing teams" technique
3. iterating over different algorithm's parameters (especially over "morning's border")
4. introducing new slightly changed interventions' comparators
5. buying interventions at the before finding the best schedule

However there is still much that can be done. More intelligent team's squeezing that takes into account the importance of the technician (this requires also finding the best definition of the importance) is a very promising direction to take. Also more complex swap interventions operators, which currently provide us with only small uplifts of the schedule should be investigated.

Other approaches may include changing the way teams are created (perhaps even finding a sort of dynamic way to do it) or totally different approaches :

1. using once created schedule to gather so-called hints, that will be used in the next iteration
2. find a way to create a best solution out of strictly defined permutation of intervention, which will allow perform local search in a different domain (not final schedules but only permutations). This will give the opportunity to use various different techniques and meta-heuristics, as for example genetic algorithms.

Références

1. Pierre-François Dutot, Alexandre Laugier and Anne-Marie Bustos : Technicians and Interventions Scheduling for Telecommunications, http ://gilco.inpg.fr/ChallengeROADEF2007/en/sujet/sujet2.pdf, France Telecom R&D (2006)
2. Ali Allahverdi, Tariq Aldowaisan : New heuristics to minimize total completion time in m-machine flowshops.
3. R. Conway, W. Maxwell and L. Miller : Theory of scheduling. Addison-Wesley (1967)
4. Ruben Ruiz, Concepcion Maroto : A comprehensive review and evaluation of permutation flowshop heuristics.
5. S. Xiang, G. Tang, T.C.E. Cheng : Solvable cases of permutation flowshop scheduling with dominating machines.