Incorporating the strength of MIP modeling in schedule construction

C.A.J. Hurkens

Eindhoven University of Technology, Department of Mathematics and Computer Science,
P.O.Box 513, NL-5600 MB Eindhoven, The Netherlands
wscor@win.tue.nl

1 Introduction

In this paper we present an approach based on polyhedral techniques for solving a complex scheduling problem. The working example is the French Telecom technician assignment problem, posed as a challenge for the Roadef 2007 competition.

Linear programming techniques are found in the literature concerning sequencing and scheduling but not in great abundance. For instance they show up for tackling relatively *simple* problems such as minimizing maximum lateness for a preemptive schedule of jobs with release times on unrelated machines [1]. Also linear programming models with a combinatorial structure such as flows or matchings can be fruitful in some scheduling contexts. It is very tempting to use complete *integer linear models* for scheduling problems in practical applications. However, it turns out that more than often, an integer linear programming model will at best describe formally the underlying problem correctly, while it does not provide any solution when passed onto a solver. We have taken this opportunity to test how the strength of MIP modeling can be applied — in practice — to real-life scheduling challenges. We developed a schedule constructing algorithm with a lot of subroutines that make use of a CPLEX-library for solving LP and ILP problems. The principal characteristics and results for the twenty Challenge problems are given in the table 1 below.

In this table we give for each of the test instances the number of jobs, the number of technicians, the number of job requirements, and the number of immediate precedences between jobs. Moreover we present a lower bound on the objective function, based on a relaxation described in a later section. In the right half of the table we give the values of the best solution found. The value of each schedule is a certain weighted combination of 3 or 4 priority-makespans. Here $C^i_{\max}$ denotes the latest completion time of a job in the schedule belonging to priority class i. Most of these were obtained by running the generic implementation of our algorithm for a period of 40 minutes or less.

It can be seen from the table that the instances are varying a lot. The A-instances are relatively small and do not allow to reject (or outsource) jobs. The B-instances allow for outsourcing jobs (under restrictions) and some of them have a lot of precedences between jobs. Furthermore, in some B-instances the number of requirements may be rather high.

2 Solution strategy

To have some idea how the solution will look like we first compute a lower bound on the value of the schedule cost. The lower bound is based on a relaxation of the problem by allowing preemption, and by estimating the minimum number of technicians needed for a job. Next we construct a solution, partially based on the relaxed schedule. Since we have four priority classes, three of which having a non-zero weight, we always consider schedules with a fixed order for the makespans $C^1_{\max}$, $C^2_{\max}$, and $C^3_{\max}$. Note that it may be profitable to process first all jobs of say priority 2, and then the jobs of priority 1. This is particularly true if there are only few jobs of priority class 2. The table with solutions indeed shows very good solutions with makespans in other orders than 1,2,3,4.

2.1 Lower bound

For a given order of the makespans, say $C^{p1}_{\max} \leq C^{p2}_{\max} \leq C^{p3}_{\max} \leq C^4_{\max} \leq C^0_{\max}$, and a given set of jobs to schedule, a lower bound on $C^{p1}_{\max}$ is computed computing the total amount of man-hours (technician-units) MH needed to process all jobs that should be finished by time $C^{p1}_{\max}$, and

Table 1. Characteristics of French Telecom problems

Inst	nr jobs	nr techs	nr reqs	nr prec	LB	Cost	$C^1_{\max}$	$C^2_{\max}$	$C^3_{\max}$	$C^4_{\max}$
A1	5	5	6	0	2265	2340	60	15	90	-
A2	5	5	6	2	2055	4755	135	-	195	-
A3	20	7	6	0	11310	11880	300	120	360	-
A4	20	7	12	7	10629	13620	210	360	540	-
A5	50	10	6	13	26910	29355	855	240	300	-
A6	50	10	20	11	17625	20280	525	120	780	-
A7	100	20	20	31	28442	32520	600	780	960	-
A8	100	20	20	21	16191	18960	480	180	600	-
A9	100	20	20	22	25553	28320	720	240	960	-
A10	100	15	20	31	36399	40650	1020	435	1200	-
B1	200	20	16	47	32085	35460	420	1005	1755	2610
B2	300	30	15	143	14296	18300	450	165	615	930
B3	400	40	16	57	14610	16965	195	480	870	1305
B4	400	30	120	112	16635	27015	645	240	1005	1575
B5	500	50	28	427	45060	94200	1620	2310	3060	4260
B6	500	30	24	457	24180	30510	750	285	1035	1380
B7	500	100	50	387	27481	33060	720	480	1080	1860
B8	800	150	40	440	31950	32160	480	840	1230	2040
B9	120	60	25	55	27420	28080	720	360	480	960
B10	120	40	25	55	34830	35040	960	360	480	1200

computing the first time T at which this number of man-hours has been made available. If at each day m technicians are available $T = \frac{MH}{n}$ days. If the number of available technicians varies per day (which is the case here) a simple adjustment is needed. To estimate the number of technicians needed for a job the simplest way to go is to check the maximum number of required technicians per skill for a job. A more precise estimate can be made by use of an ILP formulation. This is what we do in a preprocessing phase.

Lower bounds on the other makespans are computed in a similar way.

In case it is possible to abandon jobs it is really fruitful to formulate the lower bound problem as an integer linear programming problem.

2.2 Constructing solutions

In short, the construction algorithm is based on a strategy of building a solution from scratch, starting at day 1, assigning a number of jobs in parallel to teams of technicians. These teams are being built on the fly. For each day, the technicians that are clustered to a team stay together. The basic ingredient of the algorithm is a model of matching a number of available jobs (those for which the predecessors have been scheduled) to a number of teams. Teams may also refer to a group of a single technician. Initially, at the beginning of each day, each team consists of one available technician. The set of jobs is either the complete set of available jobs, or the set of available jobs with a certain priority class. For the larger instances the set of candidate jobs had to be reduced to a manageable size. This was done by sorting the candidate jobs according to a certain *OrderingRule*, and then splitting the sorted sequences into almost equal size parts of approximately 50 jobs.

In the resulting matching problem a number of teams can be assigned to a job only if the combined expertise of the teams meets the requirements of the job. The expertise of a team can be written as the sum of the expertises of the technicians that are in the team. A team is assigned to at most one job (in each matching iteration) and a job may or may not be selected. The problem bears much ressemblance with facility location problems. Here the objective is to select a nicest combination of jobs. For measuring the quality of an assignment we consider several *Matching-Objectives*. They are based on job length, job load, job difficulty or job priority or a combination of these. The problem is formulated as a linear integer programming problem and passed to the solver (CPLEX).

A further characteristic that influences the outcome is found in the way the ILP-model of the matching problem is formulated and passed to the solver. One reason is that the solver is given only a limited amount of time. It may therefore have to break off its branch-and-bound search prematurely. A second reason is that there may be more than one optimal solution. The model has a sparse formulation which tends to solve more quickly, and a dense formulation which seems to have a tendency of producing slightly better solutions (in terms of complete day schedules and use of scarse resources). In the algorithm we may use different values for the *SparsityFactor* (between 0.0 and 1.0).

Obviously, for the instances in which part of the jobs can be abandoned, it makes quite a difference for the resulting schedule value, which set of jobs is abandoned. We consider different objectives for selecting jobs to be abandoned. Restrictions are that the cost of abandoning jobs must be within the budget, and secondly that if a job is being abandoned, then also its successors should be abandoned. The selection of abandoned jobs further influences the lower bound on the schedule cost (for the remaining jobs). Therefore we consider several *LowerBoundObjectives*. They reflect the bonus obtained by skipping certain jobs. Again this depends on length, load, priority, difficulty of jobs, or of combinations of these.

Finally, one can try to obtain a solution as close as possible to the computed lower bound. To this purpose it may be necessary to temporarily 'upgrade' some jobs: if $J_1 \rightarrow \ldots \rightarrow J_k$ is a chain of jobs all of priority 4, and the total length is higher than the difference of the relaxed makespans $C^4_{\max} - C^1_{\max}$, we should complete job J_1 earlier than $C^1_{\max}$. We may therefore change its priority class to 1, before we call the schedule construction algorithm. We consider two levels of *UpdatingPriorities* (yes/no).

2.3 Creating multiple solutions

In principle there are two ways to generate many solutions greedily according to the above scheme. One is to introduce randomness in the ordering and randomness in the matching objective coefficients. We have chosen not to do this but only vary over a limited set of parameter settings. By doing this we limit ourselves to the creation of at most $6 \cdot 4 \cdot 3 \cdot 3 \cdot 2 = 432$ schedules. Also one could consider adding a postprocessing stage to polish the obtained schedules. Most of our earliest solutions already show good quality.

References

1. Lawler, E.L. and Labetoulle, J.: On preemptive scheduling of unrelated parallel processors by linear programming. J. Assoc. Comput. Mach. 25, 612–619 (1978)