A Min-Flow Algorithm for Minimal Critical Set Detection in Resource Constrained Project Scheduling

Michele Lombardi, Michela Milano
DISI, University of Bologna
{michele.lombardi2,michela.milano}@unibo.it

Abstract

This is a summary of (Lombardi and Milano 2012), where we propose a novel method for Minimal Critical Set identification, to be used for the solution of scheduling problems via Precedence Constraint Posting. The method is based on a minimum-flow problem and a heuristic minimization step. The proposed approach is much more scalable than enumeration-based MCS detection, faster and easier to implement than enveloped-based sampling, more versatile than earliest-start-schedule sampling. As a second major contribution, the paper contains a thorough comparison (on the PSPLIB) of MCS detection method, which outlines their individual strengths and weakness. Additionally, the experimentation provides novel insight in the effectiveness of a widely employed MCS ranking heuristic.

Summary of the Journal Paper

Resource Constrained Project Scheduling: The classical Resource Constrained Project Scheduling Problem (RCPSP) consists in ordering a set A of activities a_i, connected by a set E of precedence constraints (a_i, a_j). The two sets form the so-called *Project Graph*. Each activity has a fixed duration d_i and requires some amount $rq_{i,k}$ of a set R of renewable resources. Each resource is referred to as r_k and has finite capacity c_k. The objective is to minimize the schedule makespan. The problem is NP-hard and extremely challenging. The RCPSP has many practical applications and has received enormous attention in Operations Research and in Artificial Intelligence.

Precedence Constraint Posting: *Precedence Constraint Posting* (PCP, (Policella et al. 2007)), is technique to solve RCPSPs that operates by repeatedly identifying and resolving possible resource conflicts. Conflicts are resolved by introducing additional precedence constraints. The process stop when the graph is conflict-free. The output is a resource-feasible *augmented graph*, from which an schedule can be obtained by instantiating the start times of all activities so that all the precedence constraints are satisfied. PCP approaches have the ability to tackle scheduling problemS with uncertain or decision-dependent durations and are particularly well suited to build mixed off-line/on-line systems (Lombardi, Milano, and Benini 2013).

A possible conflict on a resource r_k is encoded as *Minimal Critical Set* (MCS), i.e. a set S of activities such that: 1) all a_i in S can overlap. 2) The activities cause a resource over-usage in case of overlapping execution, i.e. $\sum_{a_i \in S} rq_{i,k} > c_k$. 3) The set is minimal, i.e. by removing any activity from the set, no over-usage occurs. An MCS corresponds to an Independent Set (IS) in the graph, i.e. to a set of nodes connected by no precedence relation. An MCS represents a minimal-size possible conflict, which can be avoided by posting a single precedence constraint between any pair of activities in the set. Many PCP approaches operate by selecting and resolving MCS. This makes the conflict identification step particularly important for the method effectiveness.

Current MCS Selection Methods: There are currently three main MCS selection approaches.

1) MCS Enumeration: in (Laborie 2005), MCS identification for r_k is performed by enumerating the ISs in the Project Graph. The activities are ranked by decreasing $rq_{i,k}$, so that the enumeration is restricted to minimal size IS. The approach has exponential worst-case complexity.

2) Envelope-based Sampling: In (Policella et al. 2004) an MCS identification method that relies on a maximum resource usage envelope is introduced. The envelope is obtained by solving a sequence of (polynomial complexity) maximum-flow problems on transformed graphs. A *peak* is a point in the envelope where the usage of r_k exceeds the capacity c_k. A set of MCS is sampled from each peak via enumeration, with exponential worst case complexity.

3) Earliest-Start Schedule Sampling: A further method from (Policella et al. 2004) is based on sampling an Earliest Start Schedule, i.e. a schedule is obtained by starting each activity at the earliest possible time allowed by the precedence constraints. This method is faster than the previous one, but it does not work for variable or uncertain durations. The sampling step has still exponential time complexity.

The three approaches select for resolution the best MCS (among the examined ones) according to a ranking heuristic. A popular ranking criterion is the preserved space (from (Laborie 2005)), which estimates the portion of the schedulability windows which stays feasible after the addition of a precedence constraint.

The Novel MCS Selection Method: Our method relies on the fact that finding the maximum-weight Independent Set in a transitive graph is a polynomial problem. By using the requirements $rq_{i,k}$ as weights and finding such set, we obtain the maximum possible usage for r_k. If this is lower than c_k, the graph is conflict free. Otherwise, we have a Conflict Set, not necessarily satisfying the minimality property. An MCS is then obtained by iteratively removing the activity a_i causing the smallest reduction in heuristic rank.

Both steps have polynomial time complexity, making our approach very scalable. The maximum weight IS is analogous to the highest peak in the usage envelope: however, the former is obtained by solving a minimum flow problem (almost) directly on the Project Graph, rather than on a more complex transformed graph (such as in (Policella et al. 2004)). Therefore, our approach requires little or no additional data structure, it is easier to implement and very well suited for incremental propagation. As a main disadvantage, our method visits a small number of sets, with the risk to miss good MCSs according to the ranking heuristic.

Experimental Setup and Comparison: We have performed and extensive comparison of MCS detection procedures on the PSPLIB. We focused on the enumeration and the envelope based ones (referred to as ENUM and PEAKS), plus our approach (MINFLOW). We look for an optimal schedule via Depth First Search, branching by iterative selecting an MCS and a precedence constraint to resolve it. The preserved space heuristic is used to rank both MCS and precedence relations. Constraint propagation is employed to improve the search performance. We have a cutoff time of 600 seconds.

Table 1 reports part of the results for the MINFLOW approach. Instances are grouped by Resource Factor (RF — the average number of required resources, averaged over the number of activities) and Resource Strength (RS – the ratio between a resource capacity and its average requirement). We report the number of optimally solved and timed-out instances (OP/TO). For the optimally solved problems, we also report the average solution time and number of fails. Table 2 and 3 contain some data about the comparison with ENUM and PEAK. Here, the "time" and "fail" rows report the gap between the MINFLOW results and the considered approach, for the instances that both methods were able to solve. Negative values mean that MINFLOW has a faster solution time/smaller search tree.

Conclusion: The MINFLOW approach appears to work pretty well, being more scalable than both ENUM (by far) and PEAKS (to a lesser, but still considerable, extent). The ENUM approach often leads to larger search trees than MINFLOW, which is quite surprising, since the enumeration is able to find the best MCS according to the preserved space heuristic. This results points out the importance to balance the MCS selection effort with the reliability of the ranking criterion. In this context, the PEAKS approach seems to provide a nice compromise, reporting overall the smallest number of time-outs. However, the method does not work well for large RF values and it often slower than MINFLOW when both approaches are able to prove optimality.

	RF	**0.25**	**0.25**	**0.25**	**0.25**	**0.50**	**0.50**	**0.50**	**0.50**
	RS	**0.20**	**0.50**	**0.70**	**1.00**	**0.20**	**0.50**	**0.70**	**1.00**
J60	OP/TO	25/5	30/0	30/0	30/0	0/30	28/2	30/0	30/0
	time	10.54	0.05	0.04	0.03	—	15.05	0.12	0.11
	fail	27372	98	68	46	—	21869	124	92
J90	OP/TO	11/19	29/1	30/0	30/0	0/30	22/8	30/0	30/0
	time	70.90	0.22	0.16	0.13	—	6.70	0.66	0.70
	fail	84016	244	131	107	—	7316	245	201
	RF	**0.75**	**0.75**	**0.75**	**0.75**	**1.00**	**1.00**	**1.00**	**1.00**
	RS	**0.20**	**0.50**	**0.70**	**1.00**	**0.20**	**0.50**	**0.70**	**1.00**
J60	OP/TO	0/30	18/12	30/0	30/0	0/30	8/22	30/0	30/0
	time	—	8.38	0.28	0.22	—	15.03	0.62	0.37
	fail	—	8892	197	106	—	13601	303	114
J90	OP/TO	0/30	17/13	30/0	30/0	0/30	12/18	30/0	30/0
	time	—	50.15	1.69	1.62	—	6.79	2.76	2.70
	fail	—	43734	343	240	—	1762	353	252

Table 1: Results for the MINFLOW approach

	RF	0.25	0.25	0.25	0.25	0.50	0.50	0.50	0.50
	RS	0.20	0.50	0.70	1.00	0.20	0.50	0.70	1.00
J60	OP/TO	25/5	30/0	30/0	30/0	0/30	24/6	30/0	30/0
	time	**-6.2%**	**-40%**	**-42.2%**	**-47.2%**	—	**-66.9%**	**-89.8%**	**-91.8%**
	fail	0.9%	2.6%	**-4%**	**-6.2%**	—	**-0.4%**	**-4.9%**	**-6.6%**
J90	OP/TO	7/23	30/0	29/1	30/0	0/30	15/15	22/8	17/13
	time	9.4%	**-70.6%**	**-80.3%**	**-87.2%**	—	**-98.7%**	**-99.2%**	**-99.2%**
	fail	16.1%	**-3.4%**	**-11%**	**-3%**	—	**-13.1%**	**-12.7%**	**-5.3%**
	RF	0.75	0.75	0.75	0.75	1.00	1.00	1.00	1.00
	RS	0.20	0.50	0.70	1.00	0.20	0.50	0.70	1.00
J60	OP/TO	0/30	15/15	30/0	27/3	0/30	2/28	16/14	17/13
	time	—	**-97.7%**	**-98%**	**-98.8%**	—	**-98.5%**	**-99.3%**	**-99.7%**
	fail	—	**-40.9%**	**-11.6%**	**-13.6%**	—	25.8%	**-19.3%**	**-23.2%**
J90	OP/TO 0/30	0/30	5/25	0/30	0/30	0/30	0/30	0/30	0/30
	time	—	—	**-99.8%**	—	—	—	—	—
	fail	—	—	**-12.4%**	—	—	—	—	—

Table 2: Results for the comparison with ENUM

References

Laborie, P. 2005. Complete MCS-Based Search: Application to Resource Constrained Project Scheduling. In *Proc. of IJCAI*, 181–186. Professional Book Center.

Lombardi, M., and Milano, M. 2012. A min-flow algorithm for minimal critical set detection in resource constrained project scheduling. *Artif. Intell.* 182-183:58–67.

Lombardi, M.; Milano, M.; and Benini, L. 2013. Robust scheduling of task graphs under execution time uncertainty. *IEEE Trans. Computers* 62(1):98–111.

Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004. Generating Robust Schedules through Temporal Flexibility. In *Proc. of ICAPS*, 209–218.

Policella, N.; Cesta, A.; Oddi, A.; and Smith, S. F. 2007. From precedence constraint posting to partial order schedules: A CSP approach to Robust Scheduling. *AI Communications* 20(3):163–180.

	RF	**0.25**	**0.25**	**0.25**	**0.25**	**0.50**	**0.50**	**0.50**	**0.50**
	RS	**0.20**	**0.50**	**0.70**	**1.00**	**0.20**	**0.50**	**0.70**	**1.00**
J60	OP/TO	30/0	30/0	30/0	30/0	4/26	29/1	30/0	30/0
	time	-28%	-76.6%	-80.2%	-79.4%	—	-56.6%	-95.2%	-95.4%
	fail	13.4%	10.8%	6.5%	6%	—	23.7%	15%	10.2%
J90	OP/TO	22/8	30/0	30/0	30/0	0/30	23/7	28/2	24/6
	time	-0.2%	-87.5%	-91.2%	-92.1%	—	-89.8%	-98.7%	-98.8%
	fail	54.1%	11.6%	6.7%	8.8%	—	25.3%	8.6%	9.9%
	RF	**0.75**	**0.75**	**0.75**	**0.75**	**1.00**	**1.00**	**1.00**	**1.00**
	RS	**0.20**	**0.50**	**0.70**	**1.00**	**0.20**	**0.50**	**0.70**	**1.00**
J60	OP/TO	0/30	17/13	30/0	30/0	0/30	7/23	29/1	28/2
	time	—	-85.4%	-98%	-98.1%	—	-63.8%	-98.6%	-99.1%
	fail	—	11.5%	12%	10.3%	—	24.1%	13%	7.8%
J90	OP/TO	0/30	4/26	14/16	12/18	0/30	1/29	7/23	9/21
	time	—	-99.5%	-99.3%	-99.4%	—	-99.6%	-99.7%	-99.6%
	fail	—	-18.9%	16.6%	11.4%	—	16.1%	4.6%	17.9%

Table 3: Results for the comparison with PEAKS