Resource load-based makespan estimation algorithm for resource-constrained project scheduling problem.

Dmitry Arkhipov[1], Olga Battaïa[1], Julien Cegarra[2]

[1] ISAE-SUPAERO, Université de Toulouse, 10 av. E. Belin - 31055 Toulouse Cedex 4 France.
miptrafter@gmail.com, olga.battaia@isae.fr

[2] Institut National Universitaire Champollion, Place de Verdun - 81 012 Albi Cedex
julien.cegarra@univ-jfc.fr

Mots-clés : *operational research ; project scheduling ; bounds estimation.*

1 Introduction

The resource constrained project scheduling problem (RCPSP) is a well-known combinatorial optimisation problem. It was shown by reduction from the 3-partition problem that the decision variant of the RCPSP is NP-complete in the strong sense even without precedence constraints and only one resource [1]. A library of instances PSPLIB was created to compare solution algorithms and lower bounds, the most known approaches were presented in [2], [3] and [4].

2 Problem formulation

The classical formulation of RCPSP with discrete times is considered. There is a set of jobs N to be processed and a set of renewable resources R. The capacity of resource $i \in R$ is defined by c_i. For any job $j \in N$, the following parameters are known : r_j – release time ; p_j – processing time ; a_{ji} $\forall i \in R$ is a non-negative amount of resource i required in each timeslot for job j processing. Precedence relations are defined by an acyclic direct graph $G(N, I)$. A schedule π for a set N is defined by an assignment of *start time* $S_j(\pi)$ and *completion time* $C_j(\pi)$, such as $C_j(\pi) - S_j(\pi) = p_j$, on any job $j \in N$. Schedule π called *feasible* if for any moment of time resource limits c_i are not violated and all jobs process without preemptions subject to precedence constraints. A set of feasible schedules is denoted as $\Pi(N, R)$. The classical objective is to find a feasible schedule with minimal makespan, i.e.

$$\min_{\pi \in \Pi(N,R)} \max_{j \in N} C_j(\pi).$$

3 Estimation Algorithm description

Estimation Algorithm can be divided into two parts : inner (AI) and master (AM).

Algorithm AI is executed on an ordered pair of resources $i,\ k \in R$ cycles timeslots starting at the initial one ($t = 0$). For each timeslot, Algorithm AI estimates the highest possible load of resource i subject to a limitation of resource k. There are two versions of Algorithm AI.

a) Time horizon T is given. Algorithm AIa terminates when the $t = T$ and returns a total estimated amount of resource i that can be used in the interval $[0, T)$.

b) Amount of resource i $Am_i \leq \sum_{j \in N} a_{ji}$ is given. Algorithm AIb terminates at a timeslot t' when total estimated amount of resource i becomes not lower than Am_i, and returns t'.

Algorithm AM cycles timeslots starting at the initial one ($t = 0$) and for any pair of resources $i,\ k \in R$ estimates makespan lower bound LB_{tik}. At the first step, we count total estimated amount $Am_{i|k}(t)$ of resource i used in the interval $[0, t+1)$ using Algorithm AIa. At the second step, a horizon required to use remaining amount of resource is estimated i $RAm_{i|k} = \sum\limits_{j \in N} a_{ji} - Am_{i|k}(t)$ by processing jobs from right to left. To find a horizon T we apply Algorithm AIb for a set of jobs N', which contains the same jobs as N but with the opposite directions of graph $G'(N', I)$ edges. Since the release times can't be used correctly for a set of jobs with inverted precedences, we set for each $j \in N'$ $r_j = \sum\limits_{l \in Cr_j(G')} p_l$, where $Cr_j(G')$ – incoming critical path to a vertex j in graph G' ($j \notin Cr_j(G')$). If $RAm_{i|k} = 0$, set $T = \min\limits_{j \in N' | a_{ji} \neq 0} r_j$ to take into account the jobs which do not require resource i.

Therefore, the makespan value cannot be lower than $LB_{tik} = t + T + 1$. AM repeats it for all timeslots $t = 0, \ldots$, applies the same technique to all ordered pairs i, k, and terminates when for each pair of resources $i,\ k \in R$ equality $RAm_{i|k} = 0$ holds. The returned value of the makespan lower bound is $LB = \max\limits_{t; i,\ k \in R} LB_{tik}$.

4 Numerical experiments

The proposed algorithm was implemented using C++ programming language and tested on the instances of PSPLIB library using processor Intel(R) Core(TM) i5-4670 3.40GHz and 8 GB of RAM. The lower bound on makespan was successfully found for each instance less than in 10 seconds. The results are presented in the table TAB. 1.

number of jobs	number of instances	bounds not worse than BKLB %	the worst ratio to BKLB %	bounds improved
30	445	45,2	66,3	0
60	450	62,0	75,2	0
90	445	72,4	81,0	2
120	600	49,0	80,7	12

TAB. 1 – PSPLIB numerical experiments results. BKLB – best known lower bound.

5 Conclusions and perspectives

A new algorithm for the lower bound on makespan was developed for RCPSP and tested on the PSPLIB data. The known lower bounds were improved for 14 instances. Future researches will be focused on algorithm improvement and on applying it to production planning problems.

Références

[1] M.R. Garey, D.S. Johnson Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*. 4 (4) : 397–411. doi :10.1137/0204035.

[2] Brucker P., Knust S. A linear programming and constraint propagation-based lower bound for the RCPSP *European Journal of Operational Research* Vol. 127 (2), 2000, 355–362.

[3] Schutt A., Feydy T., Stuckey P., Wallace M. Solving RCPSP/max by lazy clause generation. *Journal of Scheduling*, Vol. 16(3), June 2013, 273–289.

[4] T. Berthold, S. Heinz, M.E. Lubbecke, R.H. Mohring and J. Schulz. A Constraint Integer Programming Approach for Resource-Constrained Project Scheduling. Proceedings of CPAIOR 2010.