2018 IX International Conference on Optimization and Applications (OPTIMA 2018)
ISBN: 978-1-60595-587-2

New Task Domain Propagators with Polynomial Complexity for Resource-Constrained Project Scheduling Problem

Dmitry ARKHIPOV[1,2,*], Olga BATTAÏA[1], Alexander LAZAREV[2,3,4,5], German TARASOV[2], and Ilia TARASOV[1,2]

[1] Department of Complex Systems Engineering, ISAE-SUPAERO, Université de Toulouse, 10 avenue Edouard Belin - BP 54032 - 31055 Toulouse Cedex 4, France
[2] V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia
[3] Lomonosov State University, Moscow, Russia
[4] National Research University Higher School of Economics, Moscow, Russia
[5] Moscow Physical and Technical Institute (State University), Dolgoprudnyi, Russia
[*] Corresponding author

Keywords: Project scheduling, Constraint programming, Scheduling, RCPSP, Task domain propagator

Abstract. We consider a classic Resource-Constrained Project Scheduling Problem (RCPSP) which is known to be NP-hard. For defined project deadline T, each task of the project can be associated with its temporal domain – a time interval in which this task can be processed. In this research, we adopt existing resource-based methods of task domain propagation to generalized statement with time-dependent resource capacity and show how to improve its propagation efficiency. We also present new polynomial-time algorithms (propagators) to tighten such temporal task domains in order to make the optimization problem easier to solve. Moreover, we show how these propagators can be used to calculate lower bound on project makespan.

Introduction

The Resource Constrained Project Scheduling Problem (RCPSP) is a well-known scheduling theory problem. This problem is strongly NP-hard [8]. This paper is focused on constraint programming resource-based methods to solve RCPSP.

Constraint programming approaches are widely used to solve scheduling problems including RCPSP. This approach is used in well-known solvers, i.e. IBM ILOG CP Optimizer, Choco, MiniZinc, Gecode, etc. Propagators can also be used as a powerful data pre-processing tool, to make task domains tighter and add new precedence relations.

This paper is organized as follows. In the section 2 the existing resource-based propagators for RCPSP are reviewed. Problem statement and data preprocessing procedures are presented in sections 3 and 4 respectively. In the section 5 we show how to adapt existing propagators to generalized RCPSP statement with time-dependent resource capacity. In sections 6 and 7 approaches to bound resource capacity function and project makespan are presented. By the section 8 we conclude.

State of the Art

There is a wide range of Constrained Programming techniques which can be applied to find an optimal/suboptimal solution of RCPSP or to make the problem easier to solve.

Some methods allow to add new precedence relations between tasks. In [18] and [5] the term of *disjunctive graph* was introduced and used to solve scheduling problems.

This research is focused on resource-based constraints. Let us briefly list them, the discussion of the most important for us will be given in section 5. A lot of constraints can be obtained by considering Cumulative Scheduling Problem (CuSP) – a single-resource version of RCPSP. Calculation of compulsory parts [12], time-tables [13], resource profile [7], resource histogram [6] allow to clarify resource usage by tasks under any feasible schedule. Time tabling sweep algorithms ([2], [3], [14]) can be used to adjust release times and tails of tasks. In [4] the application of rectangle placement problem to improve Time Tabling technique is presented. Filtering Time Table sweep algorithm with the best theoretical complexity was suggested in [9].

Edge-Finding algorithms ([16], [20], [11]) can be used to make task domains tighter and to avoid resource overloads on time intervals. This approach can be improved by using Extended-Edge-Finding algorithms presented by [15].

One of the advantages of constraint programming method is the possibility to combine different algorithms for more efficient propagation. The synergy of Edge-Finding, Extended Edge Finding and Time Tabling gave very good results obtained in [17].

For other results of using constraint programming to solve scheduling problems, we refer to [1].

Problem Statement

We consider a generalized statement of RCPSP. There is a project which should be finalized in the time horizon T. There is a set of project tasks N, and sets of resources R. In the classic statement the capacity of resource $r \in R$ equals to constant value c_r. In this paper we consider task domain propagation methods developed for the classic statement and expand them to the generalized statement of RCPSP, where the capacity of any resource $r \in R$ is defined by some non-negative capacity function $c_r(t)$. For any task $j \in N$ the following attributes are given:

- p_j – processing time;
- r_j – release time, the earliest possible time when the task j can start the processing;
- h_j – tail, which should separate task completion time and project completion time;
- a_{jr} – required amount of resource $r \in R$.

As usual the first and the last tasks of the project are dummy with zero processing times and without any resource requirements, i.e. $p_0 = p_{n+1} = 0$, $r_0 = r_{n+1} = 0$, $h_0 = h_{n+1} = 0$, $a_{0k} = a_{n+1k} = 0$ for any $k \in R$. Precedence relations with time lags are given by weighted directed acyclic graph $G = (N, E)$ where E – is the set of edges, defined bu triplets $\{i, j, e_{ij}\}$, where e_{ij} is the time lag between processing of tasks $i \in N$ and $j \in N$.

Schedule π defines start times $S_j(\pi)$ for each task $j \in N$. Completion time of any task $j \in N$ under schedule π can be calculated by the formula $C_j(\pi) = S_j(\pi) + p_j$. Schedule π is called feasible if the following constraints are satisfied.

1. Release times and tails should be satisfied, i.e. for any $j \in N$

$$[S_j(\pi), C_j(\pi)) \subseteq [r_j, T - h_j).$$

2. For any resource $k \in R$ capacity function is not violated for any t, i.e.

$$\sum_{j \in N | t \in [S_j(\pi), C_j(\pi))} a_{jk} \leq c_k(t).$$

3. Precedence relations with time lags should be satisfied. Therefore, any value $e_{ij} \in E$ implies that for start times of tasks i and j the inequality $S_i(\pi) + e_{ij} \leq S_j(\pi)$ holds. Note that values e_{ij} and e_{ji} can both belong to E.

The set of all feasible schedules is defined by $\Pi(N, R)$.

For any $j \in N$ we denote *the earliest starting time*, *the earliest completion time*, *the latest starting time* and *the latest completion time* by est_j, ect_j, lst_j and lct_j respectively. By the problem formulation we know that the task j should be processed in the time interval $[r_j, T - h_j)$, therefore $[est_j, lct_j) \subseteq [r_j, T - h_j)$. Introduced notations are more useful, since values r_j and h_j are given and est_j, ect_j, lst_j, lct_j can be dynamically changed during task domain propagation.

Data Preprocessing

In this part, we make a list of procedures which can improve the efficiency of most of the propagators. From the complexity point of view, it is more efficient to prepare data by applying procedures 1-5 before using propagators. Procedures 4 and 5 can also be used as other propagators: it is reasonable to execute these procedures if something changed to get more adjusted results.

1. For any resource $r \in R$ we can create set of tasks $N_r \subseteq N$ which require this resource, i.e. $N_r = \{j \in N | a_{jr} > 0\}$.
2. Longest paths calculation. This takes $O(n|E| + n^2 \log n)$ operations, where $|E|$ – number of edges in graph G, using algorithm presented in [10].
3. Find the earliest/latest starting/completion times using the formulae $est_j = r_j$, $ect_j = r_j + p_j$, $lst_j = T - h_j - p_j$, $lct_j = T - h_j$.
4. If $lst_j < ect_j$ for any task $j \in N$, interval $[lst_j, ect_j)$ sets a compulsory part of j. This term was firstly proposed by [12]. The complexity of this procedure is $O(1)$ for one task and $O(n)$ for all tasks of the set N. If all compulsory parts of tasks are calculated we can create a set of tasks $N^{CP} \subseteq N$ which compulsory parts are not empty, i.e. $N^{CP} = \{j \in N | lst_j < ect_j\}$. During the process of using domain propagators, est_j and lct_j can be changed and the set N^{CP} can be supplemented.
5. Calculate highest possible amount of resource $r \in R$ which can be used by non-compulsory parts of tasks $c_r^n(t)$. This function can be calculated by the formula

$$c_r^n(t) = c_r(t) - \sum_{j \in N : lst_j \leq t \leq ect_j} a_{jr}.$$

For this procedure algorithm presented by [7] can be applied adjusted for piecewise constant resource capacity function. The complexity of this procedure is $O(n_r + m)$ operations for single resource where n_r – number of tasks which require resource r during the processing and m – number of breakpoints of $c_r(t)$. If for any task from the set N_r compulsory part $[lst_j, ect_j)$ is updated, it is reasonable to recalculate function $c_r^n(t)$ to obtain more tight bound on available resource.

Adapting Existing Propagators to Piecewise-Constant Capacity Function

There are a lot of methods of task domain propagation which are based on resource overload. In this section, we present an adaptation of existing algorithms for the RCPSP generalization with piecewise-constant capacity function and show that in some cases it can improve the efficiency of existing techniques. Presented methods are focused on bounding the earliest starting time, but can be symmetrically used to propagate the latest completion time.

Edge-Finding

The idea of the Edge-Finding algorithm was firstly proposed in [16]. This method can be divided into two parts. In the first part, specific precedence relations $\Omega \lessdot j$ for any $\Omega \subset N$ and $j \notin \Omega$ are introduced. $\Omega \lessdot j$ means that under any feasible schedule $\pi \in \Pi(N, R)$ the completion time of j is not smaller than the completion time of all tasks of Ω, i.e.

$$C_j(\pi) \geq \max_{i \in \Omega} C_i(\pi).$$

Fast algorithm to detect all such precedences was presented by Vilim [20]. In the second part for each pair (Ω, j) the following idea is used to tighten the domain of j. Let us consider $[est_\Theta, lct_\Theta)$ – the domain of any $\Theta \subseteq \Omega$, where $est_\Theta = \min_{i \in \Theta} est_i$ and $lct_\Theta = \min_{i \in \Theta} lct_i$. $\Omega \lessdot j$ leads to the fact that all tasks of Θ have to be completed before $C_j(\pi)$ which implies that amount of resource a_{jr} can not be consumed by the tasks of Θ in the interval $[S_j(\pi), lct_\Theta)$. Let $A_{\Theta r} = \sum_{i \in \Theta} a_{ir}$ – total amount of resource r required to process Θ. Therefore (Fig. 1 a)

$$S_j(\pi) \geq t' = \max_{\Theta \subseteq \Omega} \{est_\Theta + \frac{A_{\Theta r} - (c_r - a_{jr})(lct_\Theta - est_\Theta)}{a_{jr}}\} \tag{1}$$

and the earliest starting time can be updated: $est_j := \max\{est_j, t'\}$.

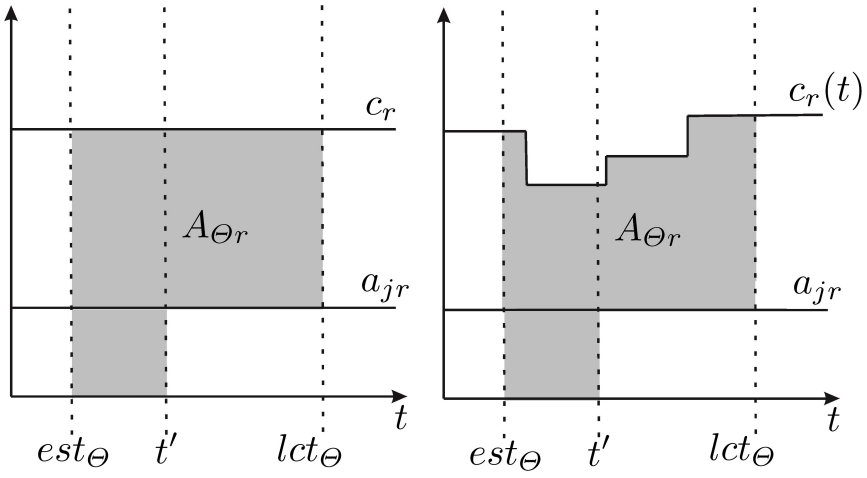

Figure 1. Edge-Finding for a) constant resource capacity, b) piecewise constant resource capacity.

For generalized statement with capacity function this approach can be used in the same way but with some changes in (1) (Fig. 1 b).

$$S_j(\pi) \geq t' = \max_{\Theta \subseteq \Omega} \{est_\Theta - \frac{A_{\Theta r} + a_{jr}(lct_\Theta - est_\Theta) + \int_{est_\Theta}^{lct_\Theta} c_r(t)dt}{a_{jr}}\}. \quad (2)$$

If capacity function is piecewise-constant value $\int_{est_\Theta}^{lct_\Theta} c_r(t)dt$ can be calculated in $O(m)$ operations, where m – number of breakpoints in the interval $[est_\Theta, lct_\Theta)$.

Extended Edge-Finding

Extended Edge-Finding rule was presented in [15]. It can be formulated as follows. Suppose that the task j starts at est_j, $[est_j, ect_j)$ overlaps the interval $[est_\Omega, lct_\Omega)$. Then if total amount of resource required to process j and Ω in the interval $[est_\Omega, lct_\Omega)$ is higher than $c_r(lct_\Omega - est_\Omega)$, then $\Omega \lessdot j$ holds, i.e. (Fig. 2 a)

$$(est_\Omega \in [est_j, ect_j)) \wedge (A_{\Omega r} + a_{jr}(ect_j - est_\Omega) > C_r(lct_\Omega - est_\Omega)) \Rightarrow \Omega \lessdot j. \quad (3)$$

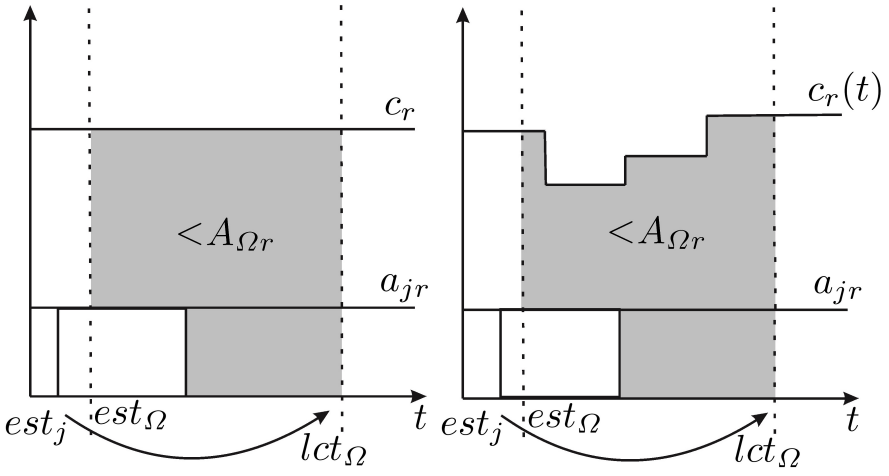

Figure 2. Extended-Edge-Finding for a) constant resource capacity, b) piecewise constant resource capacity.

For generalized statement with capacity function $c_r(t)$ this rule can be written as (Fig. 2 b)

$$(est_\Omega \in [est_j, ect_j)) \wedge (A_{\Omega r} + a_{jr}(ect_j - est_\Omega) > \int_{est_\Omega}^{lct_\Omega} c_r(t)dt) \Rightarrow \Omega \lessdot j. \quad (4)$$

Time Tabling

Time Tabling rule ([7]) bases on calculation *resource profile* – an aggregation of compulsory parts $[lst_j, ect_j)$ of tasks $j \in N$ which holds $lst_j < ect_j$

$$f_r(\Omega, t) = \sum_{j \in \Omega | t \in [lst_j, ect_j)} a_{jr}.$$

Then the *sweep technique* of [3] is used to check resource overloads (Fig. 3 a)

$$(ect_j > t) \wedge (c_r < a_{jr} + f_r(N \setminus j, t)) \Rightarrow est'_j > t.$$

The similar approach can be used to update lct_j. This method was later improved in [14] and [4].

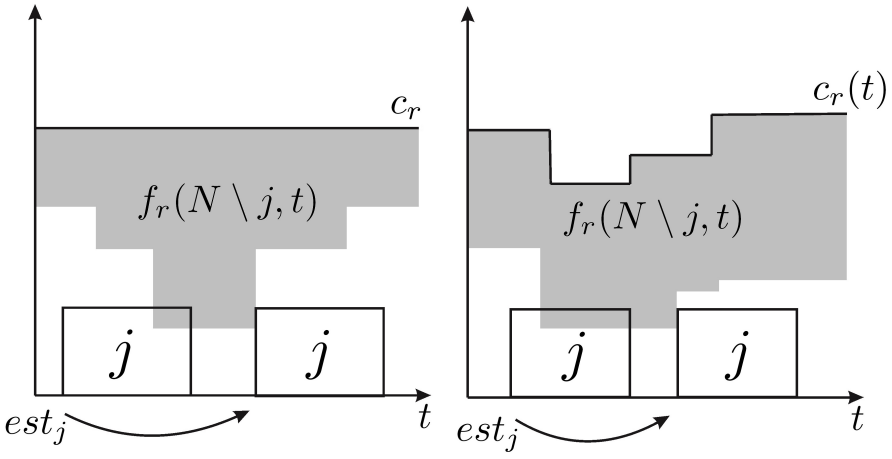

Figure 3. Time Tabling for a) constant resource capacity, b) piecewise constant resource capacity.

In terms of generalized statement this method can be written as (Fig. 3 b)

$$(ect_j > t) \wedge (c_r(t) < a_{jr} + f(N \setminus j, r, t)) \Rightarrow est'_j > t.$$

Note that if in the original algorithm we need to check overloads only in breakpoints of the function $f_r(N \setminus j, t)$, which belong to the interval $[est_j, ect_j)$. In this case, any breakpoint $t' \in [est_j, ect_j)$ of the function $C_r(t)$ is also have to be checked to get maximum domain propagation of $j \in N$.

Time Table Extended-Edge-Finding

Extended-Edge-Finding method was enhanced in [21] and [19] by combining it with Time Tabling. Let $A^f_{\Omega r}$ – amount of resource $r \in R$ required to process tasks of the set Ω plus amount of resource used by compulsory parts of tasks of the set $N \setminus \Omega$ over the interval $[est_\Omega, lct_\Omega)$, i.e.

$$A^f_{\Omega r} = A_{\Omega r} + \int_{est_\Omega}^{lct_\Omega} f(N \setminus \Omega, r, t)dt.$$

Substituting $A_{\Omega r}$ by $A^f_{\Omega r}$ in (1) and (3) gives Time-Table Extended-Edge-Finding rules.

This rules can be adopted for generalized statement by substituting $A_{\Omega r}$ by $A^f_{\Omega r}$ in (2) and (4).

Strengthening Time-Table Edge-Finding

The following two rules can be used to make Time-Table Edge-Finding algorithm more efficient.

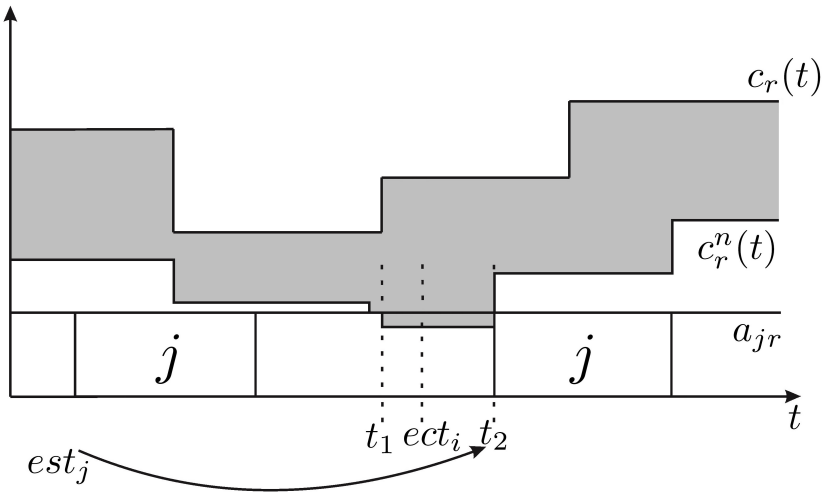

Figure 4. Strengthening Edge-Finding by considering $c_r^n(t)$ on the interval $[t_1, t_2)$ and a task $i \in \Omega$.

1. Let us consider $c_r^n(t) = c_r(t) - f_r(N, t)$ – the highest possible amount of resource which can be used by non-compulsory parts of tasks. In section 4 we showed how to calculate this function. Edge-Finding rule can be enhanced by adding the following rules for set $\Omega \subset N$ and task $j \in N$ such that $\Omega \lessdot j$.
   If for any $t \in [t_1, t_2)$ such that $[t_1, t_2) \in [est_\Omega, lct_\Omega) \setminus [ect_j, lst_j)$ holds $c_r^n(t) - a_{jr} < 0$ and there is a task $i \in \Omega$ such that $ect_i \geq t_1$, then the earliest starting time of j can be updated: $est'_j := t_2$ (Fig. 4).
2. Suppose there is a set $\Theta \subseteq \Omega$ and moment of time $t' \in [est_\Theta, lct_\Theta) \setminus [ect_j, lst_j)$, such that $c_r^n(t') < a_{jr}$ holds. Then if amount of resource required to process non-compulsory parts of all tasks of the set Θ is more then $\int_{est_\Theta}^{lct_\Theta} c_r(t) - a_{jr} dt$) and, then est_j can be updated. I.e.

$$A_{\Theta r} - \sum_{i \in \Theta} a_{ir}(|[lst_i, ect_i) \cap [est_\Theta, t')|) > \int_{est_\Theta}^{t'} (c_r(t) - a_{jr}) dt$$

   then $est'_j := t'$ (Fig 5).

This new rules can be combined with Time-Tabling Edge-Finding technique more powerful and give additional adjustments.

Capacity Function Adjustment

In the previous part, we showed how existing propagators can be adopted to generalized statement with resource capacity function and presented two new resource-based propagators. All mentioned propagators depend on the amount of resource available during the considered time interval. Therefore if we consider two instances of problem with the same sets of tasks $N^1 \equiv N^2$ with the same precedence relations and different resource capacities $c_r^1(t),\ c_r^2(t)$ such that $c_r^1(t) \geq c_r^2(t)$ for any $t \in [0, T)$, the efficiency of domain propagation in application to the second case will not be worse than to the first one, since task domains will be the same or more tight. In this section, an algorithm to clarify capacity function to improve the efficiency of existing resource-based

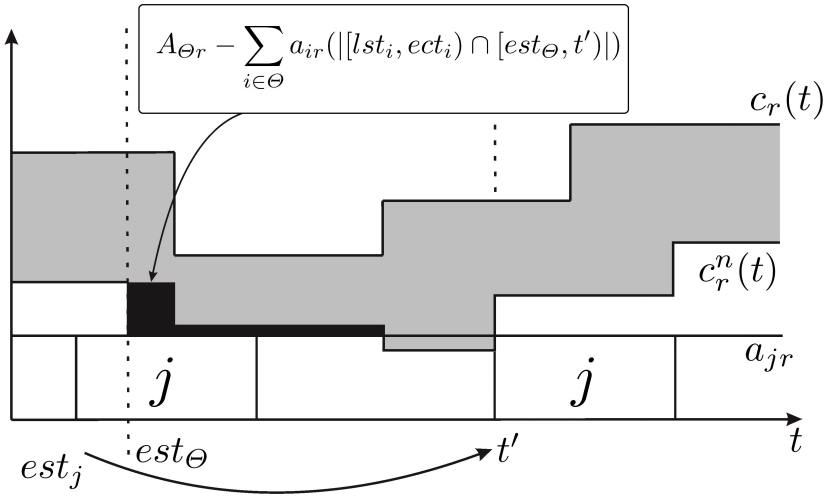

Figure 5. Strengthening Edge-Finding by considering $c_r^n(t)$ and resource usage in $[est_\Theta, t)$.

propagators is presented.

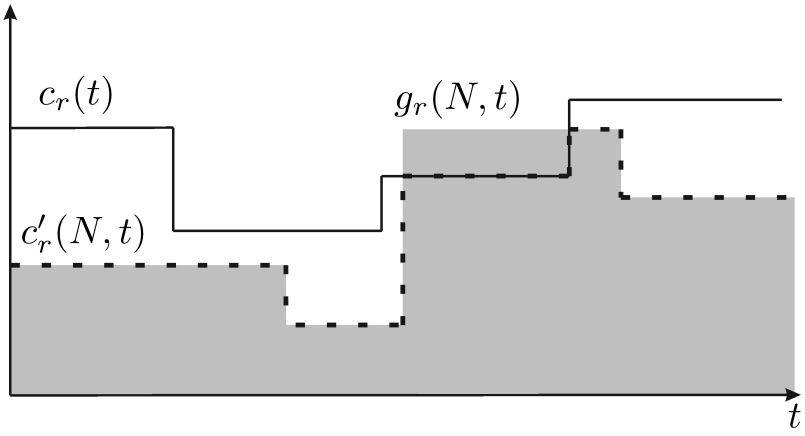

Figure 6. Capacity function adjustment.

Let us consider the following function

$$g_r(\Omega, t) = \sum_{j \in \Omega | t \in [est_j, lct_j)} a_{jr}. \tag{5}$$

$g_r(N,t)$ represents an upper bound on amount of resource, which can be consumed by the tasks of N at moment of time t. Therefore if for any t holds $g_r(N,t) < c_r(t)$, then capacity function $c_r(t)$ can be replaced by $c'_r(N,t)$ – the *highest possible consumption* of resource $r \in R$ by the tasks of the set N (Fig. 6). If $c_r(t)$ has a piecewise constant form with m breakpoints, the following algorithm allows to calculate $c'_r(N,t)$ in $O(n+m)$ operations.

Algorithm 1. Highest possible consumption bounding.

Input: Set of tasks N, $c_r(t)$ with the set of breakpoints BP_r.
Output: $c'_r(N,t)$.

$BP_t = \emptyset$;
t set $g_r(N,t) := 0$;
for all $j \in N$ **do**
    $\forall t \in [est_j, lct_j)$ increase $g_r(N,t) += a_{jr}$;
    add est_j and lct_j to BP_t;
end for
$prev_t = 0$;
$c'_r(N,0) = \min\{c_r(0), g_r(N,0)\}$;
for all $current_t \in BP_r \cup BP_t \setminus \{0\}$ in increasing order **do**
    $\forall t \in [prev_t, current_t)$ set $c'_r(N,t) := \min\{c_r(prev_t), g_r(N, prev_t)\}$;
    $prev_t := current_t$;
end for
Output $c'_r(N,t)$.

Note that this algorithm can be also applied to the classic statement with constant resource capacity. But as a result, the algorithm returns time-dependent piecewise constant highest possible consumption of resource. Therefore the existing methods are not able to be applied to the considered problem statement and generalized Edge-Finding, Extended-Edge-Finding and Time Tabling methods can be applied to propagate task domains.

Project Makespan Estimation

All presented methods can be used in obtaining destructive project makespan lower bound defined by the following Theorem.

Theorem 1. If for any time horizon T after application of task domain and capacity function propagators one of the following conditions holds, there is no feasible schedule with makespan smaller or equal to T.

1. If for any $j \in N$ task domain $[est_j, lct_j)$ will be not enough to process j, i.e. $lct_\Omega - est_\Omega < \sum_{j\in\Omega} p_j$.
2. Any set $\Omega \subseteq N$ does not have enough resource $r \in R$ to process all tasks of Ω in its domain $[est_\Omega, lct_\Omega)$ i.e. $\int_{est_\Omega}^{lct_\Omega} c_r(t)dt < lct_j - est_j < \sum_{j\in\Omega}(a_{jr}p_j)$.
3. If for any resource $r \in R$ and time moment $t \in [0,T)$ amount of resource $r \in R$ which can be used by non-compulsory parts of tasks is smaller than 0, i.e. $c_r^n(t) < 0$.

Using logarithmic search looking for the lowest possible time horizon T^* for which all three statements of the Theorem 1 are incorrect, makespan lower bound T^* can be obtained. Note that the search complexity and the quality of makespan lower bound depends on the considered sets Ω. If we consider $\Omega = i \in N$, the search speed will be polynomial.

Conclusion

In this paper existing resource-based methods of task domain propagation are considered and adopted to generalized statement of Resource-Constrained Project Scheduling Problem with time-dependent resource capacity. We presented two new approaches to tightening task domains and an

algorithm to clarify resource capacity function. These methods replenish the variety of constraint-programming techniques, which can be applied to RCPSP. We denote the function $c_r^n(t)$ which represents an upper bound on the amount of resource r which can be consumed by non-compulsory parts of tasks at time t, and presented an algorithm to calculate it in polynomial time. Well-known method of finding destructive makespan lower bound is replenished by the additional rule, based on $c_r^n(t)$ verification.

Acknowledgements

This research is supported by the Russian Foundation for Basic Research (grant 18-37-00295 mol_a)

References

[1] P. Baptiste, C. Le Pape, and Nuijten W. *Constraint-Based Scheduling.* Kluwer Academic Publishers, 1998.

[2] N. Beldiceanu and M. Carlsson. Sweep as a generic pruning technique applied to the non-overlapping rectangles constraint. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 377–391, 2001.

[3] N. Beldiceanu and M. Carlsson. A new multi-resource cumulatives constraint with negative heights. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 63–79, 2002.

[4] N. Beldiceanu, M. Carlsson, and E. Poder. New filtering for the cumulative constraint in the context of non-overlapping rectangles. In *Proceedings of the 5th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 21–35, 2008.

[5] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.

[6] Y. Caseau and F. Laburthe. Cumulative scheduling with task intervals. In *Proceedings of the Joint International Conference and Symposium on Lower bounds for Resource Constrained Project Scheduling Problem Logic Programming*, pages 363–377. The MIT Press, Cambridge, 1996.

[7] B. R. Fox. Non-chronological scheduling. In *Proceedings of AI, Simulation and Planning in High Autonomy Systems*, pages 72–77, 1990.

[8] M. Garey and D. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397—-411, 1975.

[9] S. Gay, R. Hartert, and P. Schaus. Simple and scalable time-table filtering for the cumulative constraint. In *Proceedings of the 21st International Principles and Practice of Constraint Programming CP*, pages 149–157, 2015.

[10] B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977.

[11] R. Kameugne, L. Fotso, and J. Scott. A quadratic edge-finding filtering algorithm for cumulative resource constraints. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming CP*, pages 478–492, 2011.

[12] A. Lahrichi. Ordonnancements. La notion de "parties obligatoires" et son application aux problemes cumulatifs. *RAIRO-Operations Research*, 16:241–262, 1982.

[13] C. Le Pape. *Des systemes d'ordonnancement flexibles et opportunistes*. PhD thesis, Universite Paris XI, 1988.

[14] A. Letort, N. Beldiceanu, and M. Carlsson. A scalable sweep algorithm for the cumulative constraint. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, pages 439–454, 2012.

[15] L. Mercier and P. Van Hentenryck. Edge finding for cumulative scheduling. *INFORMS Journal on Computing*, 20(1):143–153, 2008.

[16] W. Nuijten. *Time and resource constrained scheduling: A constraint satisfaction approach*. PhD thesis, Eindhoven University of Technology, 1994.

[17] P. Ouellet and C. Quimper. Time-table extended-edge-finding for the cumulative constraint. In *Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming CP*, pages 562–577, 2013.

[18] B. Roy and M. A. Sussman. Les problemes d'ordonnancement avec contraintes disjonctive. Tech. rep. Note DS 9, SEMA, Paris, 1964.

[19] A. Schutt, T. Feydy, and P.J. Stuckey. Explaining time-table-edge-finding propagation for the cumulative resource constraint. In *Proceedings of the 10th International Conference on Integration of AI and OR Techniques and Constraint Programming for Combinatorial Optimization Problems*, 2013.

[20] P. Vilim. Edge finding filtering algorithm for discrete cumulative resources in o(kn log n). In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, pages 802–816, 2009.

[21] P. Vilim. Timetable edge finding filtering algorithm for discrete cumulative resources. In *Proceedings of the 8th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 230–245, 2011.