

Time Symmetry of Project Scheduling with Time Windows and Take-give Resources

Zdeněk Hanzálek · Přemysl Šůcha

Abstract The paper studies a real production scheduling problem with special resources called take-give resources. Such resources are needed from the beginning of an activity to the completion of another activity of the production process. The difference to other works is that we consider sequence dependent changeover time on take-give resources. We formulate this problem by integer linear programming and we suggest a heuristic problem solution. In the second part of the paper we discuss how to construct a schedule in backward time orientation and we formulate a time symmetry mapping. Consequently it is shown that the mapping is bijective and involutive. Finally, performance of the heuristic algorithm with the time symmetry mapping is evaluated on a set of lacquer production benchmarks.

1 Introduction

The problem that we address in this paper is motivated by a real production scheduling specifically by a lacquer production [1] which is seen as the project scheduling problem with general temporal and resource constraints. Temporal constraints between activities of a project define minimal and maximal distance between activities, e.g. a laboratory checking must be performed within a prescribed interval related to the beginning of an activity of the lacquer production procedure. The resource constraints express limits of renewable resources such as machines (e.g. lacquer mixers) or manpower (e.g. laboratory technicians). In addition, there are special resources (mixing vessels) that are needed from the beginning of an activity to the completion of another activity. This is why we extend the classical resource constrained project scheduling problem by so called *take-give resources*. Moreover we observe that each instance of the problem can be transformed to another instance of the problem which is executed

Zdeněk Hanzálek
Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague
E-mail: hanzalek@fel.cvut.cz

Přemysl Šůcha
Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague
E-mail: suchap@fel.cvut.cz

in inverse time. Based on this observation we define a time symmetry mapping and we show that the mapping is bijective and involutive.

Various types of solutions of resource constrained project scheduling problems with positive and negative time-lags have been proposed in literature. Most of exact algorithms are based on branch and bound technique [3] but this approach is suitable for problems with less than 100 activities [6]. A heuristic algorithm by Cesta et al [5] is based on constraint satisfaction problem solving. The algorithm is based on the intuition that the most critical conflicts to be resolved first are those involving activities with large resource capacity requirements. Another heuristic algorithm proposed in [14] combines the benefits of the “squeaky wheel” optimization with an effective conflict resolution mechanism, called “bulldozing”. The possibility of improving on the squeaky wheel optimization by incorporating aspects of genetic algorithms is suggested in [15]. An overview of heuristic approaches is shown in [6] where the authors compare truncated branch and bound techniques, priority rule methods and schedule improvement procedures. A beam search heuristic is applied in [6] to scheduling of rolling ingots production. This problem cover renewable resources, changeover time and batching machines.

Up to our knowledge, there is no work dealing with take-give resources in project scheduling. Scheduling with *blocking operations* [11,4] can be seen as a subproblem of scheduling with take-give resources. Operations are blocking if they must stay on a machine after finishing when the next machine is occupied by another job. During this stay the machine is blocked for other jobs, i.e. blocking operations models the absence of storage capacity between machines. On the other hand, there is a more general framework called *reservoirs* or *storage resources* [10] usually used to model limited storage capacity or inventory limits. In this framework each activity can replenish or deplete certain amount of a resource but the resource assignment is not considered. Therefore this framework can not deal for example with changeover times on this resource type required in the lacquer production problem to model mixing vessels cleaning. A similar concept, introduced in [2], are *spatial resources*. But changeover times have not been studied in this concept yet.

Our work deals with project scheduling problems with take-give resources. We formulate this problem using Integer Linear Programming (ILP) and we suggest a priority-rule based heuristic with unscheduling step. Moreover, we study time symmetry of the problem. The motivation is to improve scheduling performance for certain types of scheduling problems occurring in production scheduling. The main contributions of this paper are: a) formulation of a project scheduling problem with take-give resources, b) a heuristic solution of the problem and c) definition of time symmetry mapping, proof of its properties and optimality.

This paper is organized as follows: Section 2 describes the notation and the scheduling problem. The ILP problem formulation is shown in Section 3. Section 4 presents a heuristic algorithm for the problem with take-give resources. Time symmetry of the problem is formalized in Section 5. Section 6 presents the heuristic algorithm performance evaluation and it shows a comparison with other approaches. Section 7 concludes the paper.

2 Problem Statement

Project scheduling problem with limited renewable resources and general temporal constraints is a well established model in the research community (see [7], [9], [13]). We assume that the project deals with a set of $n + 2$ non-preemptive activities $\mathcal{V} = \{0, 1, 2, \dots, n + 1\}$. Let $p_i \in \mathbb{R}_0^+$ be a processing time of activity i and $s_i \in \mathbb{R}_0^+$ be the start time of activity i . Activities 0 and $n + 1$ with $p_0 = p_{n+1} = 0$ denote “dummy” activities which represent the project beginning and the project termination, respectively. The activities correspond to nodes of oriented graph $G = (\mathcal{V}, \mathcal{E})$ where \mathcal{E} is a set of edges representing general temporal constraints between nodes. Each edge $e_{ij} \in \mathcal{E}$ from node i to node j is labeled by weight $\delta_{ij} \in \mathbb{R}$. The start times of activity i and activity j are subject to *temporal constraint* given by inequality

$$s_j - s_i \geq \delta_{ij} \quad \forall (i, j) \in \mathcal{V}^2; e_{ij} \in \mathcal{E}. \quad (1)$$

Let d_{ij} is the length of the longest path from node i to node j , and $d_{ij} = -\infty$ when there is no such a path or when $i = j$. Inequality $s_j - s_i \geq d_{ij}$ holds for each tuple $(i, j) \in \mathcal{V}^2$ when there is no positive cycle. When there is a positive cycle, then $d_{ii} > 0$ for all nodes i taking part in such a cycle and consequently there is no feasible schedule.

The activities are executed on a set of m (renewable) resources $\mathcal{R} = \{1, 2, \dots, m\}$. Resource $k \in \mathcal{R}$ has capacity of $R_k \in \mathbb{Z}^+$ units. Activity i requires $r_{ik} \in \mathbb{Z}_0^+$ units of resource k during its execution (i.e. from start time s_i to completion time $C_i = s_i + p_i$), such that $0 \leq r_{ik} \leq R_k$. Multiple resources may be required by one activity (sometimes called multiprocessor activity). Every activity i with $p_i = 0$ is executed on resource k with capacity $R_k = \infty$, such activity is assumed to be an event. Furthermore, we define assignment $z_{ivk} \in \{0, 1\}$, which is equal to 1 if activity i is assigned to unit v of resource k , and 0 otherwise. Consequently equation $\sum_{v=1}^{R_k} z_{ivk} = r_{ik}$ holds for each activity $i \in \mathcal{V}$ and each resource $k \in \mathcal{R}$.

In addition, we assume changeover time $o_{ij} \in \mathbb{R}_0^+$ (also called sequence dependent setup-up time) which satisfies triangular inequality. Occurrence of changeover in a given schedule is indicated by binary variable $change_{i,j,v,k}(s, z)$ which is equal to 1 if and only if $z_{ivk} = z_{jvk} = 1$ and activity i immediately precedes activity j on unit v of resource k . Consequently $s_j \geq s_i + p_i + o_{ij}$ ¹ holds when there exists unit v of resource k such that $change_{i,j,v,k}(s, z) = 1$.

For a given schedule (s, z) , let the set of activities in progress at time t be denoted as $\mathcal{A}(s, z, t) := \{i \in \mathcal{V}; s_i \leq t < s_i + p_i\} \cup \{i \in \mathcal{V}; \exists j \in \mathcal{V}, \exists k \in \mathcal{R}, \exists v \in \{1, \dots, R_k\}; change_{i,j,v,k}(s, z) = 1 \text{ and } s_i + p_i \leq t < s_i + p_i + o_{ij}\}$.

The *resource constraints* are defined as follows

$$\sum_{i \in \mathcal{A}(s, z, t)} r_{ik} \leq R_k \quad \forall t \in \langle 0, UB \rangle \text{ and } \forall k \in \mathcal{R}; R_k < \infty, \quad (2)$$

where UB denotes an upper bound on the schedule length [3].

The NP-hard problem of minimizing a schedule length $C_{max} = \max_{i \in \mathcal{V}} \{C_i\}$, being subject to the temporal and resource constraints, can be denoted by $PS | temp, o_{ij} | C_{max}$ in the tree-field notation of project scheduling problems (in order to avoid confusion with the start time, we use o_{ij} instead of s_{ij} proposed in [9]).

¹ The inequality holds for both immediate and non-immediate precedence, due to the satisfaction of triangular inequality, i.e. $o_{ij} \leq o_{il} + o_{lj} \quad \forall (i, l, j) \in \mathcal{V}^3$.

We extend this problem by a concept of *take-give resources* as follows. Let us assume a set of b take-give resources $\mathcal{Q} = \{1, 2, \dots, b\}$. Take-give resource $k \in \mathcal{Q}$ has capacity of $Q_k \in \mathbb{Z}^+$ units such that $Q_k < \infty$. Analogously to activity executed on resources, we introduce *occupation* executed on take-give resources. Occupation i requires $a_{ilk} \in \{0, 1\}$ units of take-give resource $k \in \mathcal{Q}$ during its execution. Occupation i starts its execution at s_i , start time of activity which takes a take-give resource, and finishes its execution at $C_l = s_l + p_l$, completion time of activity which gives back (i.e. releases) the take-give resource. Multiple take-give resources may be taken or given back by one activity, but there is at most one take-give resource taken by activity i and given back by activity l (i.e. $\sum_{k \in \mathcal{Q}} a_{ilk} \leq 1 \forall (i, l) \in \mathcal{V}^2$) and there is a path from i to l with $d_{il} > 0$. We define take-give assignment $\tilde{z}_{ivk} \in \{0, 1\}$, which is equal to 1 if occupation i is assigned to unit v of take-give resource k , and 0 otherwise. Consequently equation $\sum_{v=1}^{Q_k} \tilde{z}_{ivk} = a_{ilk}$ holds for each $(i, l) \in \mathcal{V}^2$ and each resource $k \in \mathcal{Q}$.

In the same way, as for resources, we assume changeover time $\tilde{o}_{ij} \in \mathbb{R}_0^+$ on take-give resources which also satisfies triangular inequality. Occurrence of changeover in a given schedule is indicated by binary variable $\tilde{change}_{i,j,v,k}(s, \tilde{z})$ which is equal to 1 if and only if $\tilde{z}_{ivk} = \tilde{z}_{jvk} = 1$ and occupation i immediately precedes occupation j on unit v of take-give resource k . Consequently $s_j \geq s_i + p_i + \tilde{o}_{ij}$ holds when there exists unit v of resource k such that $\tilde{change}_{i,j,v,k}(s, \tilde{z}) = 1$.

For a given schedule (s, z) , let the set of occupations in progress at time t is denoted as $\mathcal{O}(s, t, \tilde{z}) := \{(i, l) \in \mathcal{V}^2; \exists k \in \mathcal{Q}, a_{ilk} = 1, s_i \leq t < s_l + p_l\} \cup \{i \in \mathcal{V}; \exists j \in \mathcal{V}, \exists k \in \mathcal{Q}, \exists v \in \{1, \dots, Q_k\}, \tilde{change}_{i,j,v,k}(s, \tilde{z}) = 1 \text{ and } s_i + p_i \leq t < s_l + p_l + \tilde{o}_{ij}\}$. The *take-give resource constraints* are as follows:

$$\sum_{(i,l) \in \mathcal{O}(s,t)} a_{ilk} \leq Q_k \quad \forall t \in (0, UB) \text{ and } \forall k \in \mathcal{Q}. \quad (3)$$

A schedule $S = (s, z, \tilde{z})$ is feasible if it satisfies the temporal, resource and take-give resource constraints. A set of feasible schedules for given input data is denoted by \mathcal{S} . The problem of finding a feasible schedule with minimal C_{max} can be denoted by $PS|temp, o_{ij}, tg|C_{max}$.

An instance of problem $PS|temp, o_{ij}, tg|C_{max}$ is shown in Figure 1. The example considers 3 resources and 2 take-give resources with different capacities. There are 5 occupations, e.g. activity 12 takes resource 2 at its start time and at the completion time of activity 14 the resource is given-back. On the other hand, if we need to bind a resource taking with the completion time of an activity we could model it as it is shown in the figure on activities 2 and 3. The start time of dummy activity 3 is synchronized with the completion time of activity 2 by edges $e_{2,3}$ and $e_{3,2}$. Therefore occupation 3 can be seen as an occupation taken at the completion time of activity 2. In the same way, resource giving-back can be bind with start time of an activity as is shown in the figure on activities 10, 11. A schedule of this instance is shown in Figure 2.

3 Integer Linear Programming Formulation

In this part, we define problem $PS|temp, o_{ij}, tg|C_{max}$ by Integer Linear Programming. Let x_{ij} be a binary decision variable such that $x_{ij} = 1$ if and only if i is followed by j and $\hat{x}_{ij} = 0$ if and only if j is followed by i . Furthermore, let \hat{y}_{ij} be another binary decision variable such that if $\hat{y}_{ij} = 1$ the resource constraints are eliminated in effect, i.e. activities i and j can not share an unit.

subject to:	$\min s_{n+1}$	(4)
	$s_j - s_i \geq d_{ij},$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, d_{ij} > -\infty$ (5)
	$s_i - s_j + UB \cdot x_{ij} + UB \cdot y_{ij} \geq p_j + o_{ji},$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, \{i, j\} \in \mathcal{M}$ (6)
	$s_i - s_j + UB \cdot x_{ij} - UB \cdot y_{ij} \leq UB - p_i - o_{ij},$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, \{i, j\} \in \mathcal{M}$ (7)
	$-x_{ij} + y_{ij} \leq 0,$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, \{i, j\} \in \mathcal{M}$ (8)
	$z_{ivk} + z_{jvk} - 1 \leq 1 - y_{ij},$	$\forall (i, j) \in \mathcal{V}^2, \forall k \in \mathcal{R}, \forall v \in \{1, \dots, R_k\};$ $i \neq j, \{i, j\} \in \mathcal{M}, r_{ik} \cdot r_{jk} \geq 1$ (9)
	$\sum_{v=1}^{R_k} z_{ivk} = r_{ik},$	$\forall i \in \mathcal{V}, \forall k \in \mathcal{R}; r_{ik} \geq 1, R_k < \infty$ (10)
	$\tilde{p}_i = s_l + p_l - s_i,$	$\forall (i, l) \in \mathcal{V}^2; \sum_{k \in \mathcal{Q}} a_{ilk} = 1$ (11)
	$s_i - s_j + UB \cdot \tilde{x}_{ij} + UB \cdot \tilde{y}_{ij} \geq \tilde{p}_j + \tilde{o}_{ji},$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, \{i, j\} \in \mathcal{B}$ (12)
	$s_i - s_j + UB \cdot \tilde{x}_{ij} - UB \cdot \tilde{y}_{ij} \leq UB - \tilde{p}_i - \tilde{o}_{ij},$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, \{i, j\} \in \mathcal{B}$ (13)
	$-\tilde{x}_{ij} + \tilde{y}_{ij} \leq 0,$	$\forall (i, j) \in \mathcal{V}^2; i \neq j, \{i, j\} \in \mathcal{B}$ (14)
	$\tilde{z}_{ivk} + \tilde{z}_{jvk} - 1 \leq 1 - \tilde{y}_{ij},$	$\forall (i, j, l, h) \in \mathcal{V}^4, \forall k \in \mathcal{Q}, \forall v \in \{1, \dots, Q_k\};$ $i \neq j, \{i, j\} \in \mathcal{B}, a_{ilk} \cdot a_{jkh} = 1$ (15)
	$\sum_{v=1}^{Q_k} \tilde{z}_{ivk} = a_{ilk},$	$\forall (i, l) \in \mathcal{V}^2, \forall k \in \mathcal{Q}; a_{ilk} = 1$ (16)
domains of input variables are: $d_{ij} \in \mathbb{R}, p_i, o_{ij}, UB \in \mathbb{R}_0^+, r_{ik} \in \{1, \dots, R_k\}, a_{ilk} \in \{0, 1\}$		
domains of output variables are: $s_i \in (0, UB - p_i), z_{ivk}, \tilde{z}_{ivk} \in \{0, 1\}$		
domains of internal variables are: $\tilde{p}_i \in (0, UB), x_{ij}, \tilde{x}_{ij}, y_{ij}, \tilde{y}_{ij} \in \{0, 1\}$		

4. Combination $x_{ij} = 0$ and $y_{ij} = 1$ is not feasible due to constraint (8).

The number of units is limited using variable z_{ivk} in constraints (9) and (10). The constraint (10) satisfies that each activity i is assigned to the appropriate number of units r_{ik} for each resource $k \in \mathcal{R}$. From constraint (9) follows that when two activities i and j can overlap, i.e. $y_{ij} = 1$ then the activities can not be processed on the same unit v on resource k since $z_{ivk} + z_{jvk} - 1 \leq 0$.

The inequalities (11), (12), (13), (14), (15) and (16) stand for take-give resource constraints. The variables $\tilde{x}_{ij}, \tilde{y}_{ij}$ and \tilde{z}_{ivk} have the same meaning as x_{ij}, y_{ij} and z_{ivk} for resource constraints. The only difference is that \tilde{p}_i , processing time of occupation, is a variable while p_i , processing time of activity, is a constant. Processing time \tilde{p}_i , expressed in equation (11) is given by s_i , start time of activity i which takes take-give resource $k \in \mathcal{Q}$ and completion time of activity l which gives back the take-give resource. Finally, the objective function of the ILP model (4) minimizes the start time of dummy activity $n + 1$, i.e. the last activity of the schedule.

4 Iterative Resource Scheduling Algorithm

The presented heuristic algorithm is a priority-rule based method with unscheduling step where activities are scheduled successively according to the given priority rule. We got the idea from iterative modulo scheduling algorithm [12] designed for iterative loops scheduling, i.e. cyclic scheduling.

For given upper bound of schedule length $C \in \langle LB, UB \rangle$, where LB is the schedule makespan lower bound and UB is the schedule upper bound, the algorithm constructs a schedule according to the priority of activities (vector *priority*). Makespan

bounds LB and UB are initialized in the same way as is described in [3]. Function $\text{findSchedule}(C, \text{priority}, \text{budget})$ tries to find a feasible schedule with $C_{max} \leq C$ while the number of scheduling steps is limited by $\text{budget} = n \cdot \text{budgetRatio}$. The algorithm parameter BudgetRatio is the ratio of the maximum number of activity scheduling steps to the number of activities n .

If function findSchedule finds a feasible schedule S , activities are shifted to the left side in function $\text{shiftLeft}(S)$ so that constraints and order of activities in S is kept. In other words, this operation finds optimal schedule S_{left} for order of activities given by S using the longest paths evaluation. Furthermore, the improved schedule S_{left} is used to update upper bound $UB = C_{max}(S_{left}) - 1$. On the other hand, if function $\text{findSchedule}(C, \text{priority}, \text{budget})$ can not find a feasible schedule within the given limit of scheduling steps (i.e. budget) the schedule lower bound is increased $LB = C + 1$. The best solution S^{best} with respect to C_{max} is searched using interval bisection method over $C \in \langle LB, UB \rangle$.

```

input : an instance  $PS|temp, o_{ij}, tg|C_{max}, \text{budgetRatio}$ 
output: schedule  $S^{best}$ .
Calculate  $d_{i,j} \forall (i,j) \in \mathcal{V}^2$ ;
Calculate  $LB$  and  $UB$ ;
 $C = LB; S^{best} = \{\}$ ;
 $\text{budget} = \text{budgetRatio} \cdot n$ ;
 $\text{priority}(i) = d_{i,n+1}$ ;
while  $LB \leq UB$  do
     $S = \text{findSchedule}(C, \text{priority}, \text{budget})$ ;
    if  $S$  is feasible then
         $S_{left} = \text{shiftLeft}(S)$ ;
         $UB = C_{max}(S_{left}) - 1; S^{best} = S_{left}$ ;
    else
         $LB = C + 1$ ;
    end
     $C = \lceil (LB + UB)/2 \rceil$ ;
end
    
```

Algorithm 1: Iterative Resource Scheduling algorithm

Function $\text{findSchedule}(C, \text{priority}, \text{budget})$ constructs a schedule with $C_{max} \leq C$ using vector of priorities priority . As a priority we use the longest path from activity i to activity $n + 1$, i.e. $\text{priority}_i = d_{i,n+1}$. Until a feasible schedule is not found or upper bound of scheduling steps budget is not reached, the function choses such activity i with highest priority which was not scheduled yet. Subsequently, the algorithm determines the earliest and latest start time of the activity ES_i and LS_i respectively. Then function $\text{findTimeSlot}(i, ES_i, LS_i)$ finds the first $s_i \in \langle ES_i, LS_i \rangle$ such that there are no conflicts on resources \mathcal{R}, \mathcal{Q} . If there is no such s_i then s_i is determined according to whether activity i was scheduled once. If the activity is being scheduled for the first time, then $s_i = ES_i$ otherwise $s_i = s_i^{prev} + 1$ where s_i^{prev} is the previous start time. Finally,

function `scheduleActivity` schedule the activity at s_i . Conflicting activities, with respect to temporal or resource constraints, are unscheduled.

```

findSchedule( $C, priority, budget$ )
 $s_i = -\infty \forall i \in \{0, \dots, n + 1\}$ ;
 $scheduled = \{\}$ ;

while  $budget > 0$  and  $|scheduled| < n + 2$  do
     $i = \arg \max_{\forall i \in \mathcal{V}: i \notin scheduled} (priority_i)$ ;
     $ES_i = \max_{\forall j \in \mathcal{V}: i \in scheduled} (s_j + d_{ji})$ ;
     $LS_i = C - p_i$ ;
     $s_i = \text{findTimeSlot}(i, ES_i, LS_i)$ ;
     $scheduled = \text{scheduleActivity}(i, s_i, scheduled)$ ;
     $budget = budget - 1$ ;
end
    
```

Algorithm 2: Function `findSchedule`

Function `findTimeSlot` easily resolves conflicts on resource \mathcal{R} but a difficulty is with take-give resources \mathcal{Q} where processing time on \mathcal{Q} is given by two start times. In case, that either activity i (taking resource k) or activity j (giving resource k) is not scheduled yet, the start time of the other activity is estimated using longest path d_{ij} or d_{ji} respectively. The algorithm gives very good results (see Section 6) and is flexible with respect to change of resource constraints. On the other hand, modification to another objective function (e.g. weighted tardiness $\sum w_i T_i$) is relatively difficult.

In order to improve the algorithm performance we tried to use different priority rules (shortest distance from the start, the length of a negative cycle, ...) but a priority based on the longest path gives the best results in general. We also tested a graph preprocessing while using immediate selection [3] but the gain was very small with respect to preprocessing time.

5 Time Symmetry

Let us imagine a backward execution of a given schedule of $PS | temp, o_{ij}, tg | C_{max}$ problem illustrated in Figure 3 by a “backward” time axis which is drawn below a “forward” time axis. Basically there are two ways how to construct the schedule in backward oriented time while satisfying the temporal, resource and take-give resource constraints. The first way is to change the code, i.e. re-implement a scheduling algorithm. The second way is to transform the input data and to run the original scheduling algorithm. In the rest of this section we define such transformation as *time symmetry mapping* (*TSM*) and we derive its properties with respect to $PS | temp, o_{ij}, tg | C_{max}$ problem.

Definition 1 The time symmetry is a mapping

$(a, d, o, \bar{o}, p, r, UB, s, z, \bar{z}) \xrightarrow{TSM} (a', d', o', \bar{o}', p', r', UB', s', z', \bar{z}')$ such that

$$a'_{ilk} = a_{ilk}, \quad \forall (i, l) \in \mathcal{V}^2, \forall k \in \mathcal{R} \quad (17)$$

$$d'_{ji} = d_{ij} + p_j - p_i, \quad \forall (i, j) \in \mathcal{V}^2 \quad (18)$$

$$o'_{ji} = o_{ij}, \quad \forall (i, j) \in \mathcal{V}^2 \quad (19)$$

$$\tilde{o}'_{hl} = \tilde{o}_{ij}, \quad \forall (i, j, l, h) \in \mathcal{V}^4, \exists k \in \mathcal{Q}; a_{ilk} \cdot a_{jhk} = 1 \quad (20)$$

$$p'_i = p_i, \quad \forall i \in \mathcal{V} \quad (21)$$

$$r'_{ik} = r_{ik}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{R} \quad (22)$$

$$UB' = UB, \quad (23)$$

$$s'_i = UB - s_i - p_i, \quad \forall i \in \mathcal{V} \quad (24)$$

$$z'_{ivk} = z_{ivk}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{R}, \forall v \in \{1, \dots, R_k\} \quad (25)$$

$$\tilde{z}'_{ivk} = \tilde{z}_{ivk}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{R}, \forall v \in \{1, \dots, Q_k\}; a_{ilk} = 1 \quad (26)$$

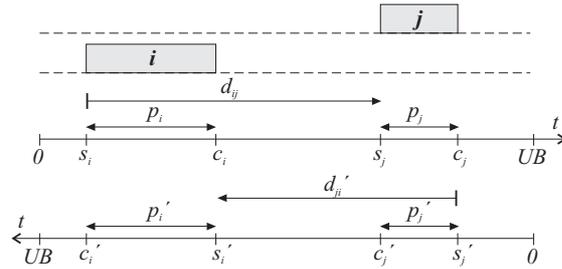


Fig. 3 Illustration of the time symmetry with forward and backward time axes

Let us assume activities i and j shown in Figure 3 where $d_{ij} > p_i$. We may observe that TSM introduces d'_{ji} with the opposite orientation (i.e. from activity j to activity i). In general variable d'_{ji} may even have a different sign than d_{ij} (for example when $d_{ij} = 1, p_j = 1, p_i = 3$) but it still preserves the same temporal constraint. Variable a in Figure 4 illustrates that the *take* operation of activity i in the forward sense is mapped on the *give* operation of activity i in the backward sense. By TSM, a matrix of change-over times is transposed and start times s'_i are given by distance from 0 on the backward time axis (i.e. UB on forward time axis).

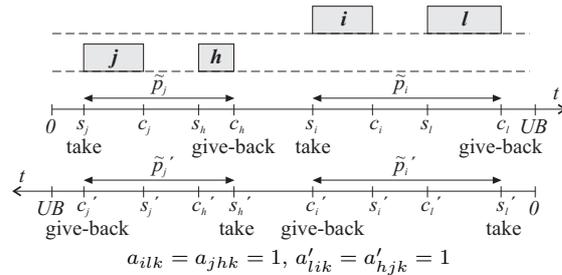


Fig. 4 Illustration of the time symmetry for take-give resources

In the following we assume that a mapping from X to Y is involutive if it holds for all $x \in X$ there exists $y \in Y$ such that $x \mapsto y \mapsto x$. For example let $X \equiv \mathbb{R}_0^+$ and $Y \equiv \mathbb{R}_0^-$, then a multiplication by -1 is involutive mapping, i.e. while applying the mapping

twice we get the identity. Let \mathbb{D} be a domain of variables of $PS|temp, o_{ij}, tg|C_{max}$ problem such that $a_{ilk}, z_{ivk} \in \{0, 1\}$, $d_{ij}, s_i, p_i, o_{ij}, UB \in \mathbb{R}$, $r_{ik} \in \mathbb{Z}$ and $d_{ij} < \infty$, $0 \leq s_i \leq UB - p_i$, $0 \leq p_i \leq UB$, $0 \leq o_{ij} \leq UB$, $0 \leq \tilde{o}_{ij} \leq UB$, $UB < \infty$, $0 \leq r_{ik} \leq R_k$. The time symmetry mapping of domain \mathbb{D} onto domain \mathbb{D} is linear, involutive and bijective. Let $(a, d, o, \tilde{o}, p, r, UB, s, z, \tilde{z})$ are variables of $PS|temp, o_{ij}, tg|C_{max}$ problem and let \mathbb{S} is the set of all feasible schedules for given n, m, b, R, Q , then time symmetry mapping of \mathbb{S} onto \mathbb{S} is involutive and bijective. Set \mathbb{S} of feasible schedules, is equivalent to the set of feasible solutions of the system (5)-(16). By (5')-(16') we denote the system obtained by TSM from (5)-(16).

Since the time symmetry mapping of \mathbb{S} onto \mathbb{S} is bijective, it holds that for every feasible schedule $(a, d, o, \tilde{o}, p, r, UB, s, z, \tilde{z}) \in \mathbb{S}$, there is exactly one feasible schedule $(a', d', o', \tilde{o}', p', r', UB', s', z', \tilde{z}') \in \mathbb{S}$ such that $(a, d, o, \tilde{o}, p, r, UB, s, z, \tilde{z}) \xrightarrow{TSM} (a', d', o', \tilde{o}', p', r', UB', s', z', \tilde{z}')$. Let $(a^F, d^F, o^F, \tilde{o}^F, p^F, r^F, UB^F)$ are input data of $PS|temp, o_{ij}, tg|C_{max}$ problem and let \mathcal{S}^F is the corresponding set of feasible schedules. Let $(a^B, d^B, o^B, \tilde{o}^B, p^B, r^B, UB^B)$ are input data of $PS|temp, o_{ij}, tg|C_{max}$ problem such that $(a^F, d^F, o^F, \tilde{o}^F, p^F, r^F, UB^F) \xrightarrow{TSM} (a^B, d^B, o^B, \tilde{o}^B, p^B, r^B, UB^B)$ and \mathcal{S}^B is the corresponding set of feasible schedules. Time symmetry mapping from \mathcal{S}^F to \mathcal{S}^B is involutive and bijective.

Consequences of TSM properties: We may obtain feasible schedule $S^F = (s^F, z^F, \tilde{z}^F)$ while performing this sequence:

- we map $(a^F, d^F, o^F, \tilde{o}^F, p^F, r^F, UB^F)$ to $(a^B, d^B, o^B, \tilde{o}^B, p^B, r^B, UB^B)$
- we solve the problem by an arbitrary algorithm while obtaining $S^B = (s^B, z^B, \tilde{z}^B)$
- we map (s^B, z^B, \tilde{z}^B) to (s^F, z^F, \tilde{z}^F) .

6 Experimental Results

In this section we show performance evaluations of Iterative Resource Scheduling algorithm. The experiments were performed on standard benchmarks UBOxxx for $PS|temp|C_{max}$ (i.e. RCPSP/max) problem (http://www.wior.uni-karlsruhe.de/LS_Neumann/Forschung/ProGenMax/) [6] and on instances of lacquers production [1, 8] where take-give resources are employed. All experiments were performed on an Intel Pentium 4 at 2.8 GHz. Algorithm Iterative Resource Scheduling was implemented in C# under Microsoft .NET framework. For all experiments with Iterative Resource Scheduling algorithm parameter was set to $budgetRatio = 2$ unless indicated otherwise.

Iterative Resource Scheduling algorithm was tested on standard benchmarks for RCPSP/max problem [6]. The results for benchmark UBO500 with $n = 500$ are summarized in Table 2 in Appendix where the first column denotes name of the benchmark instance. Column C_{max}^{best} denotes the best C_{max} obtained by algorithms: BB (Branch-and-bound algorithm), AM (Approximation method), FB (Filtered Beam Search), PR (Multi-Pass Priority-Rule Method), TS (Tabu Search) and GA (Genetic Algorithm) [6]. Name of the algorithm, which found the solution with C_{max}^{best} , is denoted in column *Algorithm*. Resulting C_{max} obtained by Iterative Resource Scheduling algorithm (C_{max}^{IRS}) is shown in the next column while the corresponding computation time is mentioned in column *CPU time*.

With respect to results obtained by [6] (considering 500 seconds time limit on 333MHz Pentium PC for each algorithm) our results are comparable or better. In addition, since Iterative Resource Scheduling algorithm solves the more general problem,

Instance	Orders			n	F <i>budgetRatio</i> = 2		F <i>budgetRatio</i> = 4		B <i>budgetRatio</i> = 2	
	met	bro	uni		<i>C_{max}</i>	CPU	<i>C_{max}</i>	CPU	<i>C_{max}</i>	CPU
					[-]	time [s]	[-]	time [s]	[-]	time [s]
lacquery-1-01.ins	7	8	7	170	73681	3.58	73681	9.66	71435	13.31
lacquery-1-02.ins	8	8	7	176	77449	3.92	77449	10.41	75203	15.56
lacquery-1-03.ins	8	7	8	175	68313	4.77	68683	13.16	73432	11.66
lacquery-1-04.ins	7	8	8	178	74455	3.88	74455	9.61	72231	13.91
lacquery-1-05.ins	8	8	8	184	78223	4.72	78223	12.09	75999	16.36
lacquery-1-06.ins	9	8	8	190	81991	5.59	81991	14.31	81169	17.08
lacquery-1-07.ins	8	9	8	193	77594	2.67	77594	6.39	84621	16.08
lacquery-1-08.ins	8	8	9	192	78688	5.36	78688	14.80	77180	12.99
lacquery-1-09.ins	9	9	8	199	85718	6.94	81362	9.55	88389	18.940
lacquery-1-10.ins	9	8	9	198	82456	6.30	82456	17.20	80948	15.61
lacquery-1-11.ins	8	9	9	201	81950	6.360	81950	21.70	81866	16.88
lacquery-1-12.ins	9	9	9	207	85718	7.58	85718	25.05	85634	19.38
lacquery-1-13.ins	10	9	9	213	89486	9.03	89486	30.4	89402	21.59
lacquery-1-14.ins	9	10	9	216	90539	8.95	86409	15.00	90671	35.69
lacquery-1-15.ins	9	9	10	215	87844	8.02	87844	25.25	89316	22.24
lacquery-1-16.ins	10	10	9	222	94307	10.56	90177	19.55	94439	40.25
lacquery-1-17.ins	10	9	10	221	91612	9.19	91612	24.39	93084	26.11
lacquery-1-18.ins	9	10	10	224	90798	8.11	90798	20.31	92462	25.49
lacquery-1-19.ins	10	10	10	230	94566	9.92	94566	24.86	96230	28.45

Table 1 Lacquer production benchmarks

CPU time of the algorithm is increased by “useless” code for constraints for changeover times and take-give resources. In 10% of the instances Iterative Resource Scheduling algorithm found a better solution and similar results were obtained for set of benchmarks UBO1000 with $n = 1000$.

To evaluate influence of time symmetry on Iterative Resource Scheduling algorithm we generated 19 benchmark instances by varying the number of production orders for different lacquers [1]. Table 1 shows the results that have been obtained on these benchmark instances. The first column denotes name of the benchmark instance while corresponding number of orders is shown in column *Orders*. Subcolumn *met* states for number of metallic lacquer batches, *bro* for bronze lacquer batches and *uni* for lacquer batches of type universal. Corresponding total number of activities is indicated in column *n*. Remaining three columns of the table show results (i.e., *C_{max}* and *CPU time*) for original problem (*F* forward) with *budgetRatio* = 2 and *budgetRatio* = 4 and the problem obtained by time symmetry mapping (*B* backward) with *budgetRatio* = 2.

The Influence of algorithm parameter *budgetRatio* is demonstrated in Figure 5. This experiment was performed on a set of 90 benchmark instances UBO100 with $n = 100$. Value *meanCPU time* denotes mean algorithm computation time. The number of feasible solutions found by Iterative Resource Scheduling algorithm is denoted by *#feasible*. Up to our experiences, a reasonable compromise between computation time and quality of the resulting schedule is usually achieved with *budgetRatio* = 2.

7 Conclusions

Inspired by practical case study we extended the project scheduling problem by take-give resources which allows to consider changeover times. Furthermore, this paper proposes a heuristic problem solution suitable for considerably large problem instances. Significant part of the paper is dedicated to the time symmetry mapping, i.e. scheduling in backward time orientation. We shows that this mapping, a simple and fast polynomial transformation, is bijective and involutive.

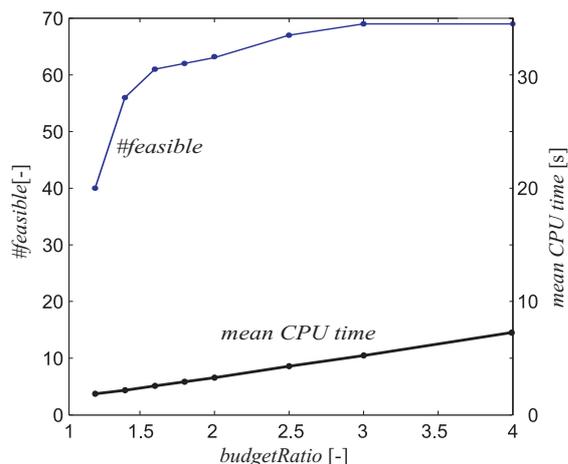


Fig. 5 Parameter *budgetRatio* influence

In order to improve the solution while using the same heuristic algorithm we used the time symmetry to find new feasible solution (in 4% of randomly generated benchmarks) and better C_{max} (about 10 % in half of the cases). A use of different priority rules and preprocessing techniques leads to considerably smaller improvements while having the same processing time.

In our current work we apply the approach described in this article to a production optimization on the assembly line. The structure of the problem is such that while using the time symmetry mapping, better C_{max} by factor of 10% is always found.

In our future work, we would like to exploit the time symmetry mapping inside heuristic algorithms while performing several iterations in forward and several in backward sense or while sharing information between two algorithms running in parallel, since the mapping is easy to extend to internal variables (like primal and dual formulation used in linear programming). In a similar manner, we would like to make use of this symmetry in Constraint Programming algorithms.

Acknowledgements The authors would like to thank Mirko Navara from Center for Machine Perception at Czech Technical University for valuable comments and inspirational suggestions.

This work was supported by the Ministry of Education of the Czech Republic under Research Programme MSM6840770038.

References

1. Gerd Behrmann, Ed Brinksma, Martijn Hendriks, and Angelika Mader. Production scheduling by reachability analysis - a case study. In *Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, page 140.1. IEEE Computer Society Press, 2005.
2. Ronald De Boer. *Resource-constrained multi-project management, a hierarchical decision support system*. PhD thesis, University of Twente, 1998.
3. Peter Brucker, Thomas Hilbig, and Johan Hurink. A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics*, 94(1-3):77–99, May 1999.
4. Peter Brucker and Thomas Kampmeyer. Cyclic job shop scheduling problems with blocking. *Annals of Operations Research*, 159(1):161–181, 2008.

5. Amedeo Cesta, Angelo Oddi, and Stephen F. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8(1):109–136, 2002.
6. Birger Franck, Klaus Neumann, and Christoph Schwindt. Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spektrum*, 23(3):297–324, August 2001.
7. Willy Herroelen, Bert De Reyck, and Erik Demeulemeester. Resource-constrained project scheduling : A survey of recent developments. *Computers and operations research*, 25(4):279–302, 1998. Elsevier.
8. Jan Kelbel and Zdeněk Hanzálek. A case study on earliness/tardiness scheduling by constraint programming. In *Twelfth International Conference on Principles and Practice of Constraint Programming - Doctoral Program*, pages 108–113. Laboratoire DInformatique de Nantes Atlantique, 2006.
9. Christoph Schwindt Klaus Neumann and Jrgen Zimmermann. *Project Scheduling with Time Windows and Scarce Resources*. Springer, 2003.
10. Philippe Laborie. Algorithms for propagating resource constraints in ai planning and scheduling: existing approaches and new results. *Artif. Intell.*, 143(2):151–188, 2003.
11. Alessandro Mascis and Dario Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498 – 517, 2002.
12. B. Ramakrishna Rau. Iterative modulo scheduling. PROGRESS 2000 Workshop on Embedded Systems, Utrecht, The Netherlands, 2000.
13. Christoph Schwindt. *Resource Allocation in Project Management*. Springer, 2005.
14. Tristan B. Smith. An effective algorithm for project scheduling with arbitrary temporal constraints. In *Proceedings of the 19 th National Conference on Artificial Intelligence. (2004)*, pages 544–549, San Jose, California, USA, 2004. AAAI Press, Menlo Park, California.
15. Justin Terada, Hoa Vo, and David Joslin. Combining genetic algorithms with squeaky-wheel optimization. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1329–1336, New York, NY, USA, 2006. ACM Press.

A List of Variables

a_{ilk}	binary constant indicating requirement of take-give resource k for occupation i i.e. $a_{ilk} = 1$ iff take-give resource k is taken at s_i and given (released) at C_l
b	number of take-give resources
C_i	completion time of activity i
C_{max}	schedule makespan
d_{ij}	length of the longest path from node i to node j
ES_i	the earliest start time of the activity i
G	graph
i, j, l, h	activity indices
I	instance, given by input parameters of the scheduling problem
k	resource index
LB	lower bound on schedule makespan
LS_i	the latest start time of the activity i
m	number of resources
n	number of activities
o_{ij}	changeover time (sequence dependent setup-up time) from activity i to activity j
\tilde{o}_{ij}	changeover time (sequence dependent setup-up time) from occupation i to occupation j
p_i	processing time of activity i
\tilde{p}_i	processing time of occupation i i.e. $\tilde{p}_i = C_l - s_i$ iff exists take-give resource k taken at s_i and given at C_l
Q_k	capacity of take-give resource k in number of units
r_{ik}	requirement of resource k for activity i
R_k	capacity of resource k in number of units
S	schedule
s_i	start time of activity i
UB	upper bound on schedule makespan
v	unit index
x_{ij}	binary variable indicating whether activity i precedes activity j i.e. $x_{ij} = 1$ iff $(s_i < s_j)$
\tilde{x}_{ij}	binary variable indicating whether occupation i precedes occupation j i.e. $\tilde{x}_{ij} = 1$ iff $(s_i < s_j)$
y_{ij}	binary variable indicating whether activity i and activity j must be assigned to different units of common resources i.e. if $y_{ij} = 1$ then $\forall k \in \{1, \dots, m\}, \forall v \in \{1, \dots, R_k\}; z_{ivk} \cdot z_{jvk} \neq 1$
\tilde{y}_{ij}	binary variable indicating whether occupation i and occupation j must be assigned to different units of common take-give resource i.e. if $\tilde{y}_{ij} = 1$ then $\forall k \in \{1, \dots, b\}, \forall v \in \{1, \dots, Q_k\}; \tilde{z}_{ivk} \cdot \tilde{z}_{jvk} \neq 1$
z_{ivk}	assignment binary variable i.e. $z_{ivk} = 1$ iff activity i is assigned to unit v of resource k
\tilde{z}_{ivk}	take-give assignment binary variable i.e. $\tilde{z}_{ivk} = 1$ iff occupation i is assigned to unit v of take-give resource k
δ_{ij}	length of edge e_{ij}
\mathcal{E}	set of edges e_{ij}
\mathcal{M}	set of resource conflicts between two activities i.e. unordered couple $\{i, j\} \in \mathcal{M}$ iff $\exists k \in \{1, \dots, m\}; r_{ik} \cdot r_{jk} \geq 1$ and $R_k < \infty$ and $(d_{ij} < (p_i + o_{ij})$ and $d_{ji} < (p_j + o_{ji}))$
\mathcal{Q}	set of take-give resources
\mathcal{R}	set of resources
\mathcal{S}	set of feasible schedules
\mathcal{V}	set of activities

B Results on Benchmarks

Instance	C_{max}^{best} [-]	Algorithm	C_{max}^{IRS} [-]	CPU time [s]	Instance	C_{max}^{best} [-]	Algorithm	C_{max}^{IRS} [-]	CPU time [s]
UBO500-01	inf ²	BB	inf	12,03	UBO500-46	821	AM	821	6,89
UBO500-02	2353	GA	inf	36,05	UBO500-47	1512	GA	1432	86,02
UBO500-03	2045	GA	inf	59,97	UBO500-48	1181	GA	inf	128,94
UBO500-04	2774	TS	2875	18,80	UBO500-49	1209	GA	inf	85,56
UBO500-05	inf	BB	inf	20,74	UBO500-50	1326	BB	1326	7,38
UBO500-06	2558	GA	inf	29,97	UBO500-51	1223	AM	1223	5,89
UBO500-07	inf	BB	inf	27,59	UBO500-52	1109	BB	1109	5,67
UBO500-08	2212	GA	inf	24,52	UBO500-53	1029	BB	1029	8,84
UBO500-09	inf	BB	inf	18,80	UBO500-54	825	BB	825	6,97
UBO500-10	2366	GA	inf	31,11	UBO500-55	1153	AM	1153	6,16
UBO500-11	589	AM	589	17,50	UBO500-56	976	BB	976	7,88
UBO500-12	1101	BB	1105	83,66	UBO500-57	1238	BB	1238	8,53
UBO500-13	1424	AM	1433	152,31	UBO500-58	1314	BB	1314	14,49
UBO500-14	1130	GA	inf	203,67	UBO500-59	1060	BB	1060	6,31
UBO500-15	683	GA	669	18,31	UBO500-60	1067	BB	1067	8,95
UBO500-16	982	GA	931	11,34	UBO500-61	2175	GA	2398	192,70
UBO500-17	1122	PR	1122	47,63	UBO500-62	2962	GA	3601	61,98
UBO500-18	978	TS	965	59,05	UBO500-63	inf	BB	inf	12,08
UBO500-19	1084	GA	inf	243,77	UBO500-64	2160	GA	2331	156,99
UBO500-20	1027	GA	inf	245,56	UBO500-65	inf	BB	inf	12,58
UBO500-21	717	BB	717	13,49	UBO500-66	3167	GA	inf	14,47
UBO500-22	983	BB	983	7,69	UBO500-67	2905	GA	inf	19,30
UBO500-23	848	AM	848	10,80	UBO500-68	2337	GA	2718	85,11
UBO500-24	1107	BB	1107	7,02	UBO500-69	2459	GA	2620	93,83
UBO500-25	1027	BB	1027	6,75	UBO500-70	2123	GA	inf	12,67
UBO500-26	804	BB	804	6,08	UBO500-71	1343	GA	1311	88,74
UBO500-27	749	BB	749	8,97	UBO500-72	1437	FB	1439	67,02
UBO500-28	913	BB	913	6,45	UBO500-73	1925	GA	inf	104,56
UBO500-29	893	BB	893	7,36	UBO500-74	2459	GA	inf	93,59
UBO500-30	792	BB	792	7,66	UBO500-75	976	BB	976	47,77
UBO500-31	inf	BB	inf	13,47	UBO500-76	2077	GA	2108	75,88
UBO500-32	inf	BB	inf	14,97	UBO500-77	1047	FB	1043	84,38
UBO500-33	2343	GA	inf	24,02	UBO500-78	2011	FB	2086	70,89
UBO500-34	inf	BB	inf	11,88	UBO500-79	1727	TS	1755	68,03
UBO500-35	inf	BB	inf	10,48	UBO500-80	1462	GA	1550	83,83
UBO500-36	2211	GA	2382	305,72	UBO500-81	1164	BB	1164	6,38
UBO500-37	2318	GA	inf	14,67	UBO500-82	1238	BB	1238	5,09
UBO500-38	2575	TS	inf	41,22	UBO500-83	1849	AM	1878	51,61
UBO500-39	inf	BB	inf	11,92	UBO500-84	936	BB	938	29,42
UBO500-40	2628	GA	inf	23,67	UBO500-85	1418	BB	1418	5,58
UBO500-41	1243	TS	1226	97,22	UBO500-86	1420	BB	1420	5,19
UBO500-42	1038	FB	1079	103,77	UBO500-87	1276	AM	1274	42,17
UBO500-43	1354	TS	1333	79,39	UBO500-88	1300	BB	1300	5,00
UBO500-44	801	AM	801	8,64	UBO500-89	1419	BB	1419	16,22
UBO500-45	1088	BB	1088	10,63	UBO500-90	1062	BB	1062	6,69

² Schedule was not found.

Table 2 Benchmark UBO500