Total Setup Time Minimisation in Production Scheduling with Alternatives

Zdeněk Hanzálek[1 2], Roman Čapek[1], and Přemysl Šůcha[1]

[1] FEE, Czech Technical University in Prague
Karlovo náměstí. 13, 121 35 Prague 2, Czech Republic
[2] CIIRC, Czech Technical University in Prague,
Jugoslávských partyzánů 1580/3, 160 00 Prague 6, Czech Republic

Abstract. The research presented in this paper is focused on the scheduling problem with alternative process plans where the goal is to minimise the sum of all the performed setup times in the schedule. The setup times play an important role in scheduling problems, yet they are, in most cases, considered only as an additional constraint, not as a part of the objective function. We propose a model, based on the resource constrained project scheduling problem with alternative process plans, release times and deadlines, that includes the setup times in the scheduling criterion. Both the exact mathematical model and the new heuristic algorithm are proposed to solve the problem. The effectiveness of the proposed two-phase heuristic algorithm, designed with the intention to solve the large instances of the problem, is evaluated on a wide set of instances.

1 Introduction

This article is dedicated to the *resource constrained project scheduling problem* with *alternative process plans* while the *total setup time* is minimised. Up to our knowledge, there is no existing solution approach for such a problem and therefore, a new model and a new heuristic algorithm is proposed for the considered problem with the intention to solve large instances with up to 1000 activities.

Sequence dependent setup times (also called *changeovers*) are crucial for the problems where the resources are very expensive in terms of wasting their time by unnecessary setups. Setup times represent the time necessary to reconfigure the resource or to change its functionality. During this time period, no work on the resources can be performed, which can cause the entire process flow to be inefficient. The problem in minimisation of the total setup time is a part of many manufacturing processes (we "sell the machinery time") as well as it is often a crucial constraint in the optimisation of algorithms for the field-programmable gate arrays (FPGAs) where the reconfiguration of the available resources is very time consuming. In other words, the minimisation of the time and the costs related to setting up the resources is a natural demand that can be applied in many different optimisation problems. Yet the setup times are almost always considered only as a problem constraint, not as a part of the criterion. One of the main goals of this research is to fill the gap in this area, i.e. to propose a generic approach to deal with the minimisation of the total setup time.

In this article, we consider shared resources and precedence constraints among activities. The classification of the resource constraint project scheduling problem (RCPSP) is used for the problem representation. Furthermore, alternative process plans are considered in the scheduling model to cover the flexibility of the studied processes. The alternative process plans allow one to define more possible ways how to finish the process, differing in the required resources, time constraints or even in the number of activities and precedence relations among them. As a result, not all of the given activities will be present in the final schedule. The considered problem can be classified as the resource constrained project scheduling problem with alternative process plans (RCPSP-APP) where the goal is to minimise the total setup time (TST), equal to the sum of the overall performed setup time (TST) in the schedule.

2 Literature review

The resource constrained project scheduling problem (RCPSP), which is used in this article, is a well-known problem with many applications. According to [18], RCPSP can be defined as a set of activities with specific requirements that have to be processed on a particular work centre with limited capacity. [5] and [7] proposed a formal notation and categorisation for the RCPSP problems as well as for their extensions. Other reviews of the models and the solution methods can be found e.g. in [13], [8], [9], [6], [16], [21], [20], [25] and [15]. The multi-mode resource constrained project scheduling problem (MRCPSP), which is an extension of RCPSP with more execution modes for each activity, has been studied in [12], [26], [25], [28] and [29]. Apart from the non-renewable resources, each MRCPSP problem can be represented as the RCPSP-APP problem: each activity with multiple modes is to be transformed into the appropriate number of single-mode activities while only one is always selected in the schedule.

Therefore, the formalism of the alternative process plans is a generalisation of the multi-mode behaviour of activities in the MRCPS problem. It allows us to model how to complete projects more than one way, while not only the resource demands, but also the number of activities, the precedence relations, etc. can differ among the alternative process plans. [2, 3] defined a structure called *Nested temporal network with alternatives* (NTNA) to model alternative process plans. [4] formulated a constraint-based representation of the alternative activities. [11] dealt with the RCPSP extended by the alternative process plans and the sequence dependent setup times. The authors presented a mixed integer linear programming (MILP) model for the exact solution of small instances and a heuristic called *iterative resource scheduling with alternatives* (IRSA) for larger ones. [19] proposed three algorithms for the jobshop problem with processing alternatives. [22] and [23] focused on RCPSP with alternatives that is close to the jobshop problem and proposed agent based metaheuristic algorithms to minimise the makespan. [27] presented an integration model of process planning and scheduling problems which are carried out simultaneously. The authors developed a genetic algorithm to minimise the schedule length.

Allahverdi et al. [1] dealt with the setup times in general and published a survey in which many different problems related to the setup times are summarised. The authors also reported on solution approaches and proposed a notation for all of these problems. [31] published a study for a metal casting company concerning the minimisation of the total setup costs in which the authors demonstrate the importance of setup times by calculating the savings to the company. [14] dealt with the general shop problem with the sequence dependent setup times. The authors proposed a two phase Pareto heuristic to minimise the makespan and the total setup costs. In the first phase, the makespan is minimised and, in the second phase, the total setup costs are minimised, while the makespan is not allowed to get worse. [30] focused on a single machine earliness tardiness problem with sequence dependent setup times. The objective function is to minimise the total setup time, earliness and tardiness. [24] proposed a hybrid simulated annealing algorithm for the single machine problem with sequence dependent setup times. The objective function is given by the sum of the setup costs, delay costs and holding costs.

3 Paper contribution and outline

The main contribution of this paper is the formulation of novel problem, incorporating the alternative process plans and a criterion based on the performed setup times into the area of the resource constrained project scheduling problems. The strength of the proposed model, formulated using the mixed integer linear programming, is in the combination of the well known RCPSP formalism with the additional flexibility gained by the alternative process plans and the total setup time minimisation. Such a problem has not been studied before in this range. There were only a few attempts to deal with the scheduling problems where the criterion reflects the setup times. The closest problem that can be found in the literature, when compared to the approach studied in this article, was published by [14] who focused on the job shop problem with the alternative machines while the makespan and the total setup time is minimised. Compared to the problem studied in [14], the model proposed in this paper is developed for more general problems, namely for non-unary resources, deadlines of activities and more complex precedence rules including alternative process plans.

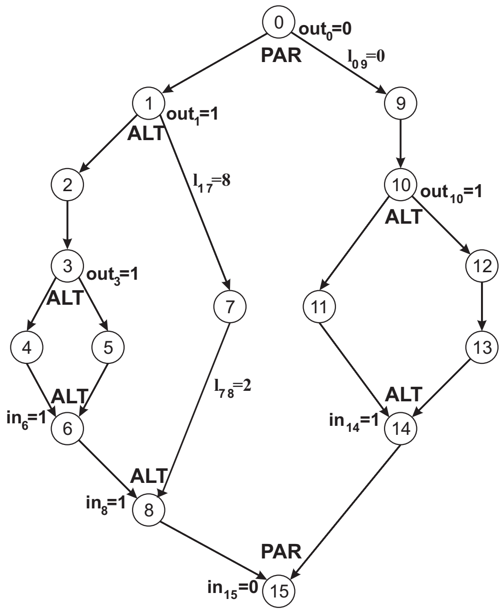
The second contribution lies in the newly developed algorithm able to solve the instances of the RCPSP-APP problem with up to 1000 activities. The effectiveness of the algorithm is evaluated using the datasets published in [10] while the proposed algorithm outperforms the results presented in [14]. Moreover, the algorithm presented in this paper is able to solve instances with 1000 activities within dozens of seconds.

The rest of the paper is organised as follows: Section 4 provides a definition of the considered problem for which the mathematical model is presented in Section 5. A new heuristic algorithm is proposed in Section 6. Section 7 presents the results of the performance evaluation of the developed algorithm and Section 8 concludes the work.

4 Problem statement

The problem considered in this paper is defined by a set of activities, a set of resources, a set of constraints and an optimality criterion. Let $\mathcal{A} = \{1 \dots n\}$ be the set of n activities representing the project to be scheduled. Furthermore, let $\mathcal{A}_{\mathcal{E}} = \mathcal{A} \cup \{0, n+1\}$ be the extended set of activities, where dummy activities 0 and $n+1$ with zero processing time restrict the whole project. Activity 0 represents the start and activity $n+1$ the end of the project. There are m resource types $\mathcal{R} = \{R_1 \dots R_m\}$ where each resource type $R_q \in \mathcal{R}$ has a discrete capacity $\theta_q \geq 1$, i.e. there are θ_q resource units available for resource type R_q. Each activity $i \in \mathcal{A}_{\mathcal{E}}$ has the following parameters: processing time $p_i \geq 0$, release time $r_i \geq 0$, deadline $\tilde{d}_i \geq 0$ and the resource demand $\mathrm{R}_i^k > 0$ for one resource type $R_k \in \mathcal{R}$. In this article, only mono-resource activities are considered, meaning that each activity demands for only one resource type. Additional constraints of the problem are defined by the alternative process plans, the non-negative start to start time-lags and the sequence dependent setup times.

The alternative process plans are defined using the nested temporal network with alternatives (NTNA) presented by [3]. NTNA is a directed acyclic graph $G = (V, E)$ where each node $i \in V$ corresponds to activity $i \in \mathcal{A}_{\mathcal{E}}$ and each edge $e = (i, j) \in E$ represents one temporal constraint in the form of a non-negative start to start time-lag $s_i + l_{ij} \leq s_j$ (where $l_{ij} \in \mathbb{R}_0^+$), i.e a minimal amount of time between the start times of activities i and j. Furthermore, each node i of the graph has an input label $in_i \in \{0, 1\}$ and an output label $out_i \in \{0, 1\}$, denoting the type of input and output branching, which can be either parallel or alternative. Based on the NTNA instance, some of the activities, called the



p=[0, 3, 2, 1, 3, 2, 1, 7, 2, 3, 2, 3, 2, 3, 2, 0]

r=[0, 0, 0, 6, 6, 6, 6, 5, 0, 0, 2, 5, 5, 5, 0, 0]

$\tilde{d}$=[5, 9, 9, 9, 15, 15, 20, 9, 25, 9, 11, 15, 13, 16, 20, 30]

res=[4, 2, 1, 2, 2, 3, 1, 3, 3, 1, 2, 3, 3, 3, 2, 4]

R_i^k=[0, 2, 2, 1, 2, 3, 2, 1, 1, 2, 2, 3, 2, 2, 1, 0]

R_1	2	6	9
2	0	3	1
6	3	0	3
9	1	3	0

R_2	1	3	4	10	14
1	0	3	3	4	2
3	2	0	2	4	2
4	3	3	0	2	3
10	4	4	2	0	2
14	3	2	3	2	0

R_3	5	7	8	11	12	13
5	0	2	2	0	0	3
7	2	0	0	2	3	3
8	2	0	0	2	3	3
11	0	3	3	0	0	3
12	0	2	2	0	0	3
13	3	3	3	3	3	0

Fig. 1. Nested temporal network with alternatives - example

selected activities, will be present in the schedule and the rest, called the *rejected* activities, will not be. When there is a parallel branching at the input/output of the selected activity i ($in_i = 0/out_i = 0$), all its direct predecessors/successors have to be selected. If activity i is rejected, all its direct predecessors/successors have to be rejected as well. On the contrary, when there is an alternative branching at the input/output of the selected activity i ($in_i = 1/out_i = 1$), exactly one of its direct predecessors/successors has to be selected. If activity i is rejected, all its direct predecessors/successors have to be rejected.

Both parallel and alternative branchings can be further nested one in another. An example of the NTNA instance is shown in Figure 1 where the parallel branchings are denoted as PAR and the alternative branchings are denoted as ALT. Several time-lags are used to demonstrate how the temporal constraints are defined, see e.g. time-lag $l_{17} = 8$ that forces activity 7 to start at least 8 time units after the start time of activity 1. All the parameters related to the activities are also included; res determines the resource type required by each activity. The setup times are depicted for each resource separately.

The sequence dependent setup times $st_{ij} \geq 0$ are given for all pairs of the activities assigned to the same resource type, i.e. for all $(i,j) \in \mathcal{A}^2 : \left(\exists k : \mathrm{R}_i^k > 0 \wedge \mathrm{R}_j^k > 0\right)$. The term setup time (in our case sequence dependent setup time) st_{ij} denotes the minimal time between the completion time of activity i and the start time of activity j, if activities i and j are scheduled subsequently on the same resource type and they share at least one resource unit. The setup time can be different for each pair of the activities and therefore the actual values are determined based on the sequence of the activities. For more details, the reader is referred to [6].

The goal of the scheduling process is to select one process plan and to schedule the corresponding activities to the available resources with respect to both the temporal and the resource constraints. A process plan is a subset of all activities such that the constraints for the selection defined for the corresponding NTNA instance are satisfied. The objective function is the minimisation of the total setup time (TST), given by the sum of all setup times performed in the schedule. To represent a schedule, the following variables are used: $s_i \in \mathbb{R}_0^+$, $v_i \in \{0,1\}$ and $z_{ivk} \in \{0,1\}$. Variable s_i denotes the start time of activity $i \in \mathcal{A}_{\mathcal{E}}$, v_i determines whether activity i is selected ($v_i = 1$) or rejected ($v_i = 0$). Finally, if $z_{ivk} = 1$ then activity i is scheduled on resource unit v of resource type k; $z_{ivk} = 0$ otherwise. For the purpose of the objective function evaluation, variable $f_{ij} \in \{0,1\}$ is defined as follows: If activities i and j are scheduled subsequently on the same resource type and they share at least one unit of its resource capacity, then $f_{ij} = 1$; $f_{ij} = 0$ otherwise. The objective function is then formulated as $TST = \sum\limits_{\forall i \in \mathcal{A}} \sum\limits_{\forall j \in \mathcal{A}} f_{ij} \cdot st_{ij}$.

The setup time from activity i to activity j is always considered only once in the objective function, regardless the actual number of the resource units which are shared by both activities. Lets assume that activity i requires three units of a certain resource type and activity j also requires three units of the same resource types. Furthermore, lets assume that activity i is assigned to resource units $\{1,2,4\}$ and activity j is assigned to resource units $\{2,3,4\}$. Although the activities share two resource units, the setup time from i to j will be added to the value of the objective function only once.

Our problem can be classified as $PS|nestedAlt, l_{ij}^{min}, ST_{SD}, r_j, \tilde{d}_j|TST$ using the extended notation of [7] or as $m1|nestedAlt, min, ST_{SD}, r_j, \tilde{d}_j|TST$ using the extended notation proposed by [17]. Both notations are extended by terms $nestedAlt$ to denote the alternative process plans (see [11]), ST_{SD} to denote the sequence dependent setup times and TST to define the total setup time as the objective function according to [1]. The term PS stands for the project scheduling, $m1$ for m renewable resources, l_{ij}^{min} and min for the minimal start to start time-lags, r_j for the release times and finally $\tilde{d}_j$ for the deadlines.

5 Mathematical formulation

The mathematical formulation using the mixed integer linear programming (MILP) for the problem defined in the previous section is formulated below. For a higher efficiency of the model, variable z_{ivk} is substituted by variable z_{iu}, i.e. only one index u is used to reference the assigned resource units of a certain resource type. The mutual conversion between (v,k) and u is given as follows: $u = \sum\limits_{q=1}^{k-1} \theta_q + v$ and $k = arg \min\limits_{k} \left\{ \sum\limits_{q=1}^{k} \theta_q \geq u \right\}$; $v = u - \sum\limits_{q=1}^{k-1} \theta_q$.

In addition to variables s_i, v_i, f_{ij} and z_{ivk} (z_{iu}) defined in the previous section, auxiliary binary variables g_{ijk}, x_{ijk} and y_{ijk} are used. Variable g_{ijk} determines whether activities i and j are selected and assigned to the same resource unit k such that i is a direct predecessor of j on such resource unit. Similarly, variable x_{ijk} determines whether activities i and j are selected and assigned to the same resource unit k such that i is an arbitrary (direct or propagated) predecessor of j on such resource unit. Finally, variable y_{ijk} determines whether both activities i and j are assigned to resource unit k.

There are three types of constraints in the model - constraints for the selection of activities, temporal constraints and resource constraints. The goal is to minimise the sum of all the performed setup times in the schedule, i.e. the total setup time.

First, the constraints for the selection of the activities are stated. Equations (1) and (2) define the rules for the selection of activities in alternative branchings, Equation (3) defines the rule for the selection of the activities in parallel branchings and Equation (4) forces the schedule to have at least one selected activity (empty schedule has no relevant significance).

Second, the temporal constraints are given in three formulas. The start time of each activity is constrained by the release time and the deadline - (5) and (6). Both constraints are applied for the selected activities only. The non-negative start to start time-lags are defined in Formula (7).

The rest of the formulas then serve to define the resource constraints, including the determination of the performed setup times in the schedule. Formulas (8) and (9) prevent more activities (from overlapping) on one resource unit in one moment. Equation (10) ensures that the number of the assigned resource units is equal to the resource demand for each activity. Equations (11) and (12) are used to assign dummy activities 0 and $n + 1$ to each resource unit of each resource type, which then ease the definition of the constraints related to the setup times. Formulas (13) and (14) constrain the value of variable y_{ijk} - if both activities are scheduled on the same resource unit, then y_{ijk} is equal to 1; 0 otherwise. Formula (15) determines the value of variable x_{ijk} - if both activities i and j are assigned to the same resource unit k, they must be scheduled sequentially. Equation (16) forces each activity to have only one direct successor on each assigned resource unit. Similarly, Equation (17) forces each activity to have only one direct predecessor on each resource unit. Formula (18) prevents the cycles in values of variable g_{ijk} for each resource unit. Finally, Formula (19) determines whether a particular setup time has to be taken into consideration in the objective function, i.e. whether activities i and j are scheduled subsequently on the same resource unit.

$$\min \sum_{\forall i \in \mathcal{A}} \sum_{\forall j \in \mathcal{A}} f_{ij} \cdot st_{ij}$$

subject to:

$$v_i = \sum_{\forall j:(i,j)\in E} v_j \qquad \forall i \in \mathcal{A}_{\mathcal{E}} : out_i = 1 \tag{1}$$

$$v_i = \sum_{\forall j:(j,i)\in E} v_j \qquad \forall i \in \mathcal{A}_{\mathcal{E}} : in_i = 1 \tag{2}$$

$$v_i = v_j \qquad \forall\,(i,j) \in E : out_i = 0 \wedge in_j = 0 \tag{3}$$

$$\sum_{i \in \mathcal{A}_{\mathcal{E}}} v_i \geq 1 \tag{4}$$

$$s_i \geq r_i - (1 - v_i) \cdot UB \qquad \forall i \in \mathcal{A}_{\mathcal{E}} \tag{5}$$

$$s_i + p_i \leq \tilde{d}_i + (1 - v_i) \cdot UB \qquad \forall i \in \mathcal{A}_{\mathcal{E}} \tag{6}$$

$$s_i + l_{ij} \leq s_j + UB \cdot (2 - v_i - v_j) \qquad \forall\,(i,j) \in E \tag{7}$$

$$s_j + p_j + st_{ji} \leq s_i + UB \cdot (x_{iju} + 1 - y_{iju}) + UB \cdot (2 - v_i - v_j) \qquad \forall (i,j) \in \mathcal{A}^2 : i \neq j; \forall u \in \{1 \ldots K\} \tag{8}$$

$$s_i + p_i + st_{ij} \leq s_j + UB \cdot (2 - x_{iju} - y_{iju}) + UB \cdot (2 - v_i - v_j) \qquad \forall (i,j) \in \mathcal{A}^2 : i \neq j; \forall u \in \{1 \ldots K\} \tag{9}$$

$$\sum_{u=C+1}^{C+\theta_q} z_{iu} = \mathrm{R}_i^q \cdot v_i \qquad \forall i \in \mathcal{A}; \forall q \in \{1 \ldots m\}; C = \sum_{j=1}^{q-1} \theta_j \tag{10}$$

$$z_{0u} = 1 \qquad \forall u \in \{1 \ldots K\} \tag{11}$$

$$z_{n+1\,u} = 1 \qquad \forall u \in \{1 \ldots K\} \tag{12}$$

$$y_{iju} \geq z_{iu} + z_{ju} - 1 \qquad \forall (i,j) \in \mathcal{A}_{\mathcal{E}}^2 : i \neq j; \forall u \in \{1 \ldots K\} \tag{13}$$

$$y_{iju} \leq z_{iu} \qquad \forall (i,j) \in \mathcal{A}_{\mathcal{E}}^2 : i \neq j; \forall u \in \{1 \ldots K\} \tag{14}$$

$$x_{iju} \leq y_{iju} \qquad \forall (i,j) \in \mathcal{A}_{\mathcal{E}}^2 : i \neq j; \forall u \in \{1 \ldots K\} \tag{15}$$

$$\sum_{j=1}^{n+1} g_{iju} = z_{iu} \qquad \forall i \in \mathcal{A}; \forall u \in \{1 \ldots K\} \tag{16}$$

$$\sum_{i=0}^{n} g_{iju} = z_{ju} \qquad \forall j \in \mathcal{A}; \forall u \in \{1 \ldots K\} \tag{17}$$

$$g_{iju} \leq x_{iju} \qquad \forall (i,j) \in \mathcal{A}_{\mathcal{E}}^2; \forall u \in \{1 \ldots K\} \tag{18}$$

$$f_{ij} \cdot UB \geq \sum_{\forall u \in \{1 \ldots K\}} g_{iju} \qquad \forall (i,j) \in \mathcal{A}_{\mathcal{E}}^2 \tag{19}$$

where:

$$s_i \in \mathbb{R}_0^+;\; v_i, f_{ij}, z_{iu}, g_{iju}, x_{iju}, y_{iju} \in \{0,1\}; \tag{20}$$

$$K = \sum_{q=1}^{m} \theta_q; UB > \sum_{\forall i \in \mathcal{A}_{\mathcal{E}}} \max\left(p_i + \max_{\forall j \in \mathcal{A}_{\mathcal{E}}} st_{ij}, \max_{\forall j \in \mathcal{A}_{\mathcal{E}}} l_{ij}\right)$$

6 Heuristic algorithm

This section is dedicated to the description of the heuristic algorithm designed to solve the large instances of the problem defined in Section 4. The goal is to find a schedule determined by the selection of activities (variable v_i), their start times (variable s_i) and their assignment to resources (variable z_{ivk}) such that all the constraints are satisfied and the total setup time (TST) value is minimised.

The basic scheme of the proposed heuristic algorithm, called *STOAL* (Setup Time Optimization ALgorithm), consists of two phases - the initial phase to find any feasible solution and the local search for the improvement of the objective value. The initial phase is inspired by the IRSA algorithm published in [11] and the local search, based on a time separation technique, is inspired by the work of [14]. If a feasible solution is not found (due to the presence of deadlines) in the initial phase, the local search is not started at all and the algorithm is terminated. Detailed description of the STOAL algorithm is available from the authors upon request.

7 Performance evaluation

Two sources of instances have been used for the performance evaluation of the algorithm proposed in Section 6, designed to solve the problems with alternative process plans. First, the STOAL algorithm is evaluated on randomly generated instances and compared with the IRSA algorithm proposed by [11]. Second, the standard benchmarks of [10] are used and the results of the STOAL algorithm are compared with the results reported in [14]. Furthermore, various settings of the STOAL algorithm are discussed and tested on large instances of the problem (up to 1000 activities). The STOAL algorithm was implemented in the C# language and the experiments were performed on a PC with an Intel Core 2 Quad CPU at 2.83GHz with 8GB of RAM.

7.1 Comparison with IRSA algorithm on random instances

Random instances of the problem defined in Section 4 are generated to compare the STOAL algorithm with the existing IRSA algorithm, designed for the minimisation of the schedule length for the RCPSP with alternative process plans and positive and negative time-lags. As reported in [11], the IRSA algorithm was originally implemented in the Matlab environment, but for the purpose of this article, we have re-implemented the algorithm in the C# language to get a fair comparison. Since IRSA does not consider resources with non-unary capacities, all the instances contain only unary resources and all activities have resource demand equal to 1. There are three different sets of generated instances: *loose*, *medium* and *tight* which differ in the specification of release times and deadlines. Each set further contains 100 instances for each of 20, 50, 100 and 200 activities per instance.

The instances were generated with the following settings: the parameters for each activity i were randomly selected from the intervals $p_i \in \langle 2, 10 \rangle$, $r_i \in \langle 0, k_1 \cdot n \rangle$, $\tilde{d_i} \in \langle k_1 \cdot \frac{n}{2}, k_2 \cdot n \rangle$ where n is the number of activities in a particular instance and k_1 and k_2 are constants depending on the type of the instance (*loose*/*medium*/*tight*); namely $k_1 = 5$ and $k_2 = 15$ for the *loose* instances, $k_1 = 7$ and $k_2 = 13$ for the *medium* instances and $k_1 = 10$ and $k_2 = 10$ for the *tight* instances. For each instance, the release times and deadlines are sorted in non-decreasing order and assigned to the activities based on the precedence relations from activity 0 towards activity $n + 1$. Each activity has the resource demand equal to one, i.e. $\mathrm{R}_i^q = 1$, for one resource type q. The number of resource types m is randomly chosen from interval $\langle 1, 2 \rangle$ for 20 and 50 activities per instance and from interval $\langle 1, 5 \rangle$ for 100 and 200 activities per instance. The setup times st_{ij} are generated in the interval $\langle 5, 10 \rangle$ and the non-negative start to start time-lags l_{ij} in the interval $\langle 0, 20 \rangle$. The structural properties of the generated NTNA instances are as follows: If node i starts the parallel branching, the number of successive nodes lies in interval $\langle 5, 10 \rangle$. Similarly, if node i starts the alternative branching, the number of direct successors lies in interval $\langle 2, 4 \rangle$.

Table 1 shows the comparison of the results obtained by the IRSA algorithm and by the STOAL algorithm. Column $feas$ determines the percentage ratio of feasible solutions found by each algorithm. Column TST contains an arithmetic average value of the objective function for instances that were successfully solved by both algorithms. Column *time* determines the average computational time (in milliseconds) to solve a single instance regardless of whether a solution was found or not. Finally, column TST^{impr} states the improvement of the STOAL algorithm over the IRSA algorithm in terms of the TST value.

The number of feasible solutions found is almost the same for both tested algorithms, but the STOAL algorithm outperforms the IRSA algorithm in both the TST value and the solution time. The fact that the success rate in finding feasible solutions is equal proves that the STOAL algorithm is very effective for

		IRSA			STOAL			
n	$type$	$feas$ [%]	TST	$time$ [ms]	$feas$ [%]	TST	$time$ [ms]	TST^{impr} [%]
20	loose	100	102	5	100	76	3	25.50
50	loose	100	254	36	100	215	12	15.35
100	loose	100	494	112	100	427	77	13.56
200	loose	94	942	322	100	824	141	12.53
20	medium	62	77	4	69	76	2	1.01
50	medium	58	226	29	60	221	14	2.10
100	medium	69	386	98	64	371	57	3.92
200	medium	72	707	293	68	662	112	6.37
20	tight	44	65	4	41	63	2	1.03
50	tight	31	183	25	32	183	15	0.00
100	tight	26	302	86	33	295	48	2.30
200	tight	37	597	266	42	592	119	0.92

Table 1. Comparison with IRSA algorithm using new random instances

the considered temporal constraints, since the IRSA algorithm was developed with the main aim to find any feasible solution. The most significant difference in terms of the objective value can be observed for the *loose* instances where the flexibility of the activities is higher and, therefore, the optimisation can be performed in a wider scope.

7.2 Comparison with algorithm of Focacci [14]

For a further evaluation of the STOAL algorithm, the instances of the general job shop problem proposed by [10] are used. As a reference, the results for such instances reported in [14] are considered. The problem studied in [10] is a sub-problem of the problem defined in Section 4 since there are no release times or deadlines, no alternative process plans and the resources are considered to be unary. The objective function reported in [14] is twofold, first the makespan in minimised and then the total setup time is being minimised without a deterioration of the makespan value.

Focacci Table 2 shows the comparison of the STOAL algorithm with the one published by [14]. Compared with the algorithm described by [14], the STOAL algorithm improved the value of the total setup time by more than 16% in average. The price for the better value of the TST is the higher value of the makespan, by almost 19% in average. Such a trade-off between the makespan and the total setup time shows the good efficiency of the STOAL algorithm proposed in terms of the total setup time criterion.

The big trade-off between TST and makespan is probably incurred by the alternative process plans. The two criteria should be more linked in the classical problems without alternatives. The makespan criterion probably makes more sense, since it includes the setup time as well. On the other hand the sole TST criterion may be useful when the setup is costly (e.g. including the waste of the material).

	Focacci		STOAL			
Set	TST	C_{max}	TST	C_{max}	TST^{impr} [%]	C_{max}^{det} [%]
t2-ps12	1 530	1 445	1 010	1 920	33.99	32.87
t2-ps13	1 430	1 658	1 330	1 872	7.00	18.93
t2-pss12	1 220	1 362	950	1 599	22.13	17.4
t2-pss13	1 140	1 522	1 140	1 610	0	5.78
average	1 330	1 497	1 110	1 825	16.54	18.74

Table 2. Comparison with [14] using instances of [10]

8 Conclusion

This paper fills the gap in the literature, where only very few pieces of work have been dedicated to scheduling problems with setup times as a part of the criterion. The setup times are usually considered only as a constraint. The proposed innovative model combines the RCPSP problem with the alternative process plans and the criterion to minimise the total setup time in the schedule. Furthermore, the model includes the release time and deadline for each activity and the non-negative start to start time-lags for precedence constrained activities. For such a model, of the studied problem, the mathematical formulation, using the mixed integer linear programming (MILP), is proposed.

The two-phase heuristic algorithm is then developed to solve the large instances of the considered problem. The goal of the algorithm first phase is to find any feasible solution and the second phase, based on the time separation of the schedule, is dedicated to improve the existing schedule in terms of the total setup time. The STOAL algorithm is compared with two reference algorithms. The experiments show a very good performance of the STOAL algorithm in both the quality of the solutions and the running time.

In the future research, we want to concentrate on situations where tasks are owned by agents representing, e.g. departments of a company. In this case, the resources are shared by the agents, and the problem becomes a multiobjective optimization problem. This extension requires a realistic definition of a fair use of resources with respect to the objective of the individual agents.

Acknowledgments

This work was supported by the Grant Agency of the Czech Republic under the Project FOREST GACR P103-16-23509S.

References

1. Allahverdi, A., Ng, C., Cheng, T., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research 187(3), 985–1032 (2008)
2. Barták, R., Čepek, O.: Temporal networks with alternatives: Complexity and model. In: Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference (FLAIRS), Florida, USA. pp. 641–646. AAAI Press (2007)
3. Barták, R., Čepek, O.: Nested temporal networks with alternatives: recognition and tractability. In: Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Ceara, Brazil. pp. 156–157. ACM (2008)
4. Beck, J.C., Fox, M.S.: Constraint-directed techniques for scheduling alternative activities. Artificial Intelligence 121(1), 211–250 (2000)
5. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Scheduling Computer and Manufacturing Processes. Springer-Verlag New York, Inc. (1996)
6. Brucker, P.: Scheduling Algorithms. Springer-Verlag New York, Inc. (2007)
7. Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112(1), 3–41 (1999)
8. Brucker, P., Knust, S.: Complexity results for single-machine problems with positive finish-start time-lags. Computing 63(4), 219–316 (1998)
9. Brucker, P., Kunst, S.: Complex Scheduling. Springer-Verlag New York, Inc. (2006)
10. Brucker, P., Thiele, O.: A branch & bound method for the general-shop problem with sequence dependent setup-times. Operations Research Spectrum 18(3), 145–161 (1996)
11. Čapek, R., Šůcha, P., Hanzálek, Z.: Production scheduling with alternative process plans. European Journal of Operational Research 217(2), 300–311 (2012)
12. De Reyck, B., Herroelen, W.: The multi-mode resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research 119(2), 538–556 (1999)

13. Demeulemeester, E., Herroelen, W.: A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science 38(12), 1803–1818 (1992)
14. Focacci, F., Laborie, P., Nuijten, W.: Solving scheduling problems with setup times and alternative resources. In: Artificial Intelligence Planning Systems 2000 Proceedings (AIPS). pp. 1–10. AIPS (2000)
15. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207(1), 1–14 (2010)
16. Herroelen, W., De Reyck, B., Demeulemeester, E.: Resource-constrained project scheduling: A survey of recent developments. Computers & Operations Research 25(4), 279–302 (1998)
17. Herroelen, W., De Reyck, B., Demeulemeester, E.: A classification scheme for project scheduling. In: Weglarz, J. (ed.) Handbook of Recent Advances in Project Scheduling, pp. 1–26. Kluwer Academic Publishers, Dordrecht (1999)
18. Kadrou, Y., Najid, N.M.: A new heuristic to solve rcpsp with multiple execution modes and multi-skilled labor. In: Proceedings of the IMACS Multiconference on Computational Engineering in Systems Applications (CESA). pp. 1–8. IEEE (2006)
19. Kis, T.: Job-shop scheduling with processing alternatives. European Journal of Operational Research 151(2), 307–322 (2003)
20. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174(1), 23–37 (2006)
21. Kolisch, R., Padman, R.: An integrated survey of deterministic project scheduling. Omega The International Journal of Management Science 29(3), 249–272 (2001)
22. Leung, C.W., Wong, T.N., Maka, K.L., Fung, R.Y.K.: Integrated process planning and scheduling by an agent-based ant colony optimization. Computers & Industrial Engineering 59(1), 166–180 (2010)
23. Li, X., Zhang, C., Gao, L., Li, W., Shao, X.: An agent-based approach for integrated process planning and scheduling. Expert Systems with Applications 37(2), 1256–1264 (2010)
24. Mirabi, M.: A hybrid simulated annealing for the single-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times and costs and dynamic release of jobs. The International Journal of Advanced Manufacturing Technology 54(9-12), 795–808 (2010)
25. Neumann, K., Schwindt, C., Zimmermann, J.: Project scheduling with time windows and scarce resources. Springer-Verlag Berlin Heidelberg (2003)
26. Salewski, F., Schirmer, A., Drexl, A.: Project scheduling under resource and mode identity constraints: Model, complexity, methods and application. European Journal of Operational Research 102(1), 88–110 (1997)

27. Shao, X., Li, X., Gao, L., Zhang, C.: Integration of process planning and scheduling - a modified genetic algorithm-based approach. Computers & Operations Research 36(6), 2082–2096 (2009)
28. Van Peteghem, V., Vanhoucke, M.: An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. Tech. rep., Faculty of Economics and Business Adminstration (Ghent University) (2011)
29. Van Peteghem, V., Vanhoucke, M.: Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. Journal of Heuristics 17(6), 705–728 (2011)
30. Wang, L., Wang, M.: A hybrid algorithm for earliness-tardiness scheduling problem with sequence dependent setup time. In: Proceedings of the 36th Conference on Decision & Control. pp. 1219–1223. IEEE (1997)
31. Yuan, X.M., Khoo, H.H., Spedding, T.A., Bainbridge, I., Taplin, D.M.R.: Minimizing total setup cost for a metal casting company. Winter Simulation Conference 2, 1189–1194 (2004)