Alternative Process Plans in Wire Harnesses Production

R. Čapek[*], P. Šůcha, Z. Hanzálek
Department of Control Engineering, Faculty of Electrical Engineering,
Czech Technical University, Karlovo náměstí. 13, 121 35 Prague 2, Czech Republic

Abstract

This paper deals with a scheduling problem with alternative process plans that was motivated by a production of wire harnesses where certain parts can be processed manually or automatically by different types of machines. Only a subset of all the given activities will form the solution, so the decision whether the activity will appear in the schedule has to be made during the scheduling process. The problem considered is an extension of the resource constrained project scheduling problem with unary resources, positive and negative time-lags and sequence dependent setup times. We have proposed the problem representation by a special graph allowing to define alternative process plans. For this representation of the problem, an integer linear programming model is formulated. Finally a heuristic algorithm based on priority schedule construction with an unscheduling step is proposed and used to solve the case study of the wire harnesses production.

1 Introduction

1.1 Motivation

The motivation for our research is the scheduling of production processes which typically involves more than one way how to complete the product. Such alternative process plans occur in the production of wire harnesses in Styl Plzeň, a cooperative society. Operations to produce a wire harness can be performed in various ways, using fully automated machines, semiautomated machines or manually operated ones with special equipment. Not only are the resource requirements different, but the processing times, precedence relations and also the number of activities in each process plan can differ in general too. The process plan defines a set of activities such that their execution leads to the completion of a product. We use the term *alternative process plans* since there are more ways to complete the product in our problem while only one of them has to be executed. Hence the goal of the scheduling is to choose a subset of all activities that forms one process plan and to schedule them according to the given criterion.

[*]Corresponding author. Email: capekrom@fel.cvut.cz

Traditional scheduling algorithms according to [4] assume exactly given set of activities to be scheduled, i.e. there is only one process plan. In this paper, the traditional scheduling approach is extended by a definition of alternative process plans, hence the traditional time scheduling and the decision which activities will form the schedule are both integrated into one problem. The problem can be formalized as an extension of $PS|temp, o_{ij}|C_{max}$, i.e. the resource constrained project scheduling problem with positive and negative time-lags, sequence dependent setup times and dedicated resources. Sequence dependent setup times serve to cover the time needed to change the equipment or set up a machine between two different operations. Time-lags are useful to specify the relative time position of two activities in general. They have been used for modeling production which involves chemical and thermal processes [16]. Both setup times and time-lags are used to model the production of wire harnesses.

1.2 Related Works

The problem outlined in the previous subsection is addressed in the literature as *scheduling with alternative* (*optional*) *activities* (*tasks*) or problem with *alternative process plans* or *scheduling with alternatives* [3, 1, 2, 8, 9]. To avoid any misunderstandings, let us assume that the notions *activity*, *operation* and *task* have the same meaning and we will prefer to use the term *activity* in this article. To represent alternative process plans, some type of special graph is usually used in the existing works. Beck and Fox [3] established the *Modified Temporal Graph* with so called *XorNodes*, *AndNodes* and *ActivityNodes* to model the possibility of choice among the process plans. Another approach to model the alternative process plans in scheduling, similar to the Modified Temporal Network methodology, was presented by Barták and Čepek [1, 2]. They use a special type of graph called *Temporal Network with Alternatives*, which is a directed acyclic graph where the nodes represent activities and the arcs correspond to temporal constraints. If the graph can be decomposed to either alternative or parallel subgraphs nested one in another, then the graph is called *Nested Temporal Network with Alternatives*. Capacho and Pastor [8, 9] studied an assembly line balancing problem with alternatives, where certain parts can be processed in several alternative modes and the goal is to balance the workload of the available resources. They evaluated a heuristic approach based on the schedule construction with various priority-rule methods [10]. The direct application of this approach prohibits the possibility to change their order according to the information acquired from the partial schedule, which is not sufficient due to general temporal constraints in our case. Kis [13] presented the constructive algorithm for the job-shop problem with alternative process plans using a special graph to represent the problem instance.

The resource constrained project scheduling problem (RCPSP) is a well known problem with many real applications. Brucker [6] summarized the notation of the RCPSP including both single-mode and multi-mode problems with various resource environments and activity characteristics. With the proposed classification, the resource constrained project scheduling problem with positive and negative time-lags and sequence dependent setup times can be formulated as $PS|temp, o_{ij}|C_{max}$.

1.3 Contribution and outline

This paper presents the production scheduling problem with alternative process plans motivated by the real production of wire harnesses. The problem representation based on the modification of Nested Temporal Network with Alternatives is in Section 2. The model also considers sequence dependent setup times and general temporal constraints. Section 3 contains the mathematical formulation. A heuristic method, where the choice of process plan and traditional scheduling are executed simultaneously, is proposed in Section 4. Computational experiments are discussed in Section 5 and the case study of the wire harnesses production is presented in Section 6. Section 7 concludes the work.

The main contributions of this paper are:

a) representation of the problem with alternative process plans using the Nested Temporal Graph with Alternatives,

b) an integer linear programming model for an exact solution of small instances,

c) a heuristic solution and its evaluation for large instances.

2 Problem Statement

2.1 Basic Definition

Let the production consist of n indivisible operations performed on the specified machines according to the process plan. Consequently, there is a set of $n+2$ non-preemptive activities $\mathcal{T} = \{0, \ldots n+1\}$ to be scheduled on a set of m dedicated resources $\mathcal{R} = \{R_1 \ldots R_m\}$. Each activity i is characterized by the processing time p_i and processor assignment θ_i. Activities 0 and $n+1$ with $p_0 = p_{n+1} = 0$ denote *dummy* activities. Precedence relations together with the definition of alternative process plans are specified via Nested Temporal Graph with Alternatives. General temporal constraints, also called time-lags, are determined by matrix L where $l_{ij} \in \mathbb{R}$ for all $(i,j) \in \mathcal{T}^2$ represents the inequality in the form $s_i + l_{ij} \leq s_j$; s_i is the start time of activity i in the schedule. If there is no temporal constraint from i to j, then $l_{ij} = -\infty$. Furthermore, $l_{0i} = r_i$, $l_{i0} = -\tilde{d}_i$ and $l_{i\ n+1} = p_i$ for all $i \in \mathcal{T}$, where r_i is the release time and $\tilde{d}_i$ is the deadline of activity i. If activity i is not constrained by release time and deadline, then $l_{0i} = 0$ and $l_{i0} = -\infty$. Sequence dependent setup times o_{ij} are considered as an additional time needed between the activities scheduled consequently on the same resource. We presume that the setup times satisfy the triangular inequality $o_{ij} + o_{jk} \geq o_{ik}$ for all $\{i,j,k\} \in \mathcal{T}^3$ [5].

All activities chosen to be present in the schedule are called *selected activities*, the activities which are not present in the schedule are called *rejected activities*. The optimality criterion is the minimization of the schedule length. Therefore, the goal of the scheduling is to find a subset of all activities according to the logical constraints such that the execution of these activities with respect to the resource and temporal constraints leads to the schedule with the shortest duration.

The above described problem can be classified as $PSm, 1, 1|nestedAlt, temp, o_{ij}|C_{max}$ using $\alpha|\beta|\gamma$ notation [4, 6] where $temp$ denotes the generalized temporal constraints and o_{ij} represents the setup times. According to the resource specification $PSm, 1, 1$, there are m resources with unary capacity and each activity requires, at most, 1 unit of the resource. We add the term $nestedAlt$ to the field describing constraints of the problem to denote the presence of alternative process plans in the nested form (see the following subsection).

2.2 Nested Temporal Graph with Alternatives

The $PSm, 1, 1|nestedAlt, temp, o_{ij}|C_{max}$ problem representation is based on the modification of Nested Temporal Network with Alternatives proposed by Barták and Čepek [2]. We extend this model by generalized temporal constraints which do not implicate any logical constraint. The set of activities $\mathcal{T}$ corresponds to the nodes of the oriented graph $G = (V, E)$, called Nested Temporal Graph with Alternatives, where E is a set of edges representing general temporal constraints and logical constraints. Each edge $e_{ij} \in E$ is labeled by a pair (l_{ij}, a_{ij}) where l_{ij} is the general temporal constraint (see the previous subsection) and a_{ij} denotes the type of the edge.

According to [2], the input/output parallel branching for activity i means that if i is the selected activity, then all its direct predecessors/successors of i have to be selected activities also. The input/output alternative branching for activity i has the following meaning: if i is the selected activity, then exactly one of its predecessors/successors has to be the selected activity. Let $a_{ij} = 1$ for all $e_{ij} \in E : e_{ij}$ is a part of the output parallel branching for activity i or e_{ij} is a part of the input parallel branching for activity j or $\delta_i^+ = \delta_j^- = 1$, where δ_i^+ is the out-degree of node i and

δ_j^- is the in-degree of node j (see [14]). Furthermore, let $a_{ij} = 2$ for all $(i,j) \in \mathcal{T}^2 : e_{ij}$ is a part of the output alternative branching for activity i or a part of the input alternative branching for activity j. Finally, $a_{ij} = 0$ otherwise. We assume that $l_{ij} \geq p_i$ for all $(i,j) \in \mathcal{T}^2 : a_{ij} \neq 0$.

Edges of the graph G with $a_{ij} = 1$ and $a_{ij} = 2$ represent the logical constraints of the problem. Edges with $a_{ij} = 0$ are additional temporal constraints, which have to be satisfied only if both i and j are the selected activities. Let $G^{prec} \subseteq G$ be a graph with nodes $V(G^{prec}) = V(G)$ and edges $E(G^{prec}) = \{(i,j) \in E(G) : a_{ij} \neq 0\}$. Then G^{prec} is an instance of the Nested Temporal Network with Alternatives. In other words, logical constraints of the $PSm,1,1|nestedAlt,temp,o_{ij}|C_{max}$ problem has to be in the nested form, which can be recognized in $\mathcal{O}\left(n^2\right)$ time [2]. Edges with $a_{ij} = 0$ can be connect arbitrary pair of nodes, since the do not imply any logical constraint.

3 Mathematical Model

In this section, the integer linear programming model for the $PSm,1,1|nestedAlt,temp,o_{ij}|C_{max}$ problem is formulated. The start time of activity i is determined by the variable $s_i \in \mathbb{R}_0^+$. Furthermore, let v_i be a binary decision variable denoting whether activity i is included in the schedule or not. If $v_i = 1$, then activity i is present in the schedule and it is the selected activity, otherwise it is not included in the schedule and therefore it is the rejected activity. Let i and j be activities assigned to the same processor, then let x_{ij} be a binary decision variable such that $x_{ij} = 1$ if activity i is followed by activity j (i.e. $s_i + p_i <= s_j$) and $x_{ij} = 0$ otherwise. Finally, UB is a positive parameter such that $UB = \sum_{\forall i \in \mathcal{T}} max\left(p_i + \underset{\forall j \in \mathcal{T}}{max}(o_{ij}), \underset{\forall j \in \mathcal{T}}{max}(l_{ij})\right)$.

The inequality (1) in the ILP formulation represents the general temporal constraints. The start times are constrained by l_{ij} only if both i and j are selected activities. The inequalities (2) and (3) stand for the resource constraints for each pair of activities with the same dedication, i.e. $\theta_i = \theta_j$. Equations (4-7) describe the propagation rule for selecting/rejecting activities. The C_{max} value is equal to the completion time of the last activity in the schedule, i.e. activity $n+1$ with $p_{n+1} = 0$ and, therefore, the minimization of s_{n+1} is a criterion of the ILP model.

$$\min s_{n+1}$$

subject to:

$$s_i + l_{ij} \leq s_j + UB \cdot (2 - v_i - v_j) \quad \forall (i,j) \in \mathcal{T}^2 \qquad (1)$$

$$s_j + p_j + o_{ji} \leq s_i + UB \cdot (2 - v_i - v_j + x_{ij}) \quad \forall (i,j) \in \mathcal{T}^2 : i < j, \theta_i = \theta_j \qquad (2)$$

$$s_i + p_i + o_{ij} \leq s_j + UB \cdot (3 - v_i - v_j - x_{ij}) \quad \forall (i,j) \in \mathcal{T}^2 : i < j, \theta_i = \theta_j \qquad (3)$$

$$\sum_{\forall i \in \mathcal{T} : a_{ij}=2} v_i = v_j \qquad \forall j \in \mathcal{T} \qquad (4)$$

$$\sum_{\forall i \in \mathcal{T} : a_{ji}=2} v_i = v_j \qquad \forall j \in \mathcal{T} \qquad (5)$$

$$v_i = v_j \qquad \forall i,j \in \mathcal{T}^2 : a_{ij} = 1 \qquad (6)$$

$$\sum_{\forall i \in \mathcal{T}} v_i \geq 1 \qquad (7)$$

where:

$$s_i \in R_0^+;\ v_i, x_{i,j} \in \{0,1\}$$

4 Heuristic Algorithm

Since $PSm,1,1|nestedAlt,temp,o_{ij}|C_{max}$ belongs to a class of NP-hard problems, the optimal solution can be obtained only for small instances. For a large problem instances, we propose a heuristic algorithm that does not ensure finding an optimal solution, but is able to handle problems with a significantly larger amount of activities than using the exact mathematical model. The idea of this algorithm, called Iterative Resource Scheduling with Alternatives (IRSA), is based on an IRS algorithm for $PS|temp,o_{ij}|C_{max}$ inspired by software pipelining and presented by [12] and [15]. It is a constructive method where activities are being added to the schedule according to their actual priority or being removed if the partial schedule is not feasible. The input of the algorithm is an instance of the $PSm,1,1|nestedAlt,temp,o_{ij}|C_{max}$ problem. The output of the algorithm is a schedule S determined by the selected activities and their start times, i.e. $S = [s,v]$. Although the IRSA algorithm is designed to solve single-resource activities and unary resources, it can be extended to solve problems with multi-resource activities and resources with a non-unary capacity.

4.1 Initialization

The algorithm (see Alg 1) starts with the estimation of the bounds for the schedule length. The upper bound C_{max}^{UB} is equal to UB (see Section 3). The lower bound is computed as $C_{max}^{LB} = ES_{n+1}$, i.e. the estimated earliest start time of activity $n+1$ computed as the shortest path between nodes 0 and $n+1$ in G^{prec} by Dijkstra's algorithm [14]. The earliest start time ES_i denotes the minimum time at which activity i can be scheduled with respect to the temporal constraints (resource constraints are not considered).

Let *open* be an activity with output alternative branching, then *close* is the corresponding activity with input alternative branching if the first common successor of all direct successors of the *open* activity is the *close* activity. The $open_i$ and $close_i$ for all $i \in \mathcal{T}$ are the activities such that $open_i$ is a predecessor, $close_i$ is a successor of activity i and the difference of their topology numbers in G^{prec} (see [14]) is minimal. Priority for activity i is computed

Alg 1 IRSA(budgetRatio, maxModifications, instance)

```
compute $C^{LB}_{max}$ and $C^{UB}_{max}$;
set initial $priority$;
$budget = budgetRatio \cdot n$;
$actualRestarts = 0$;
while $C^{UB}_{max} \geq C^{LB}_{max}$
 $S = findSchedule\left(C^{UB}_{max}, priority, budget\right)$ ;
 if $S$ is feasible
  $s = shiftLeft\left(S\right)$ ;
  $C^{UB}_{max} = s_{n+1} - 1$;
 else
  if $actualModifications < maxModifications$
   $priority = modifyPriority\left(priority, S\right)$;
   $actualRestarts = actualRestarts + 1$;
  else
   break;
  end
 end
end
```

as $priority_i = c_1 \cdot u_{i,n+1} - c_2 \cdot u_{open_i,close_i}$ where $u_{i,j}$ is the longest path between nodes i and j in G computed by the Floyd-Warshall algorithm [14] and c_1 and c_2 are constants (we use $c_1/c_2 = 5/3$). In the example in Figure 3, the *open* and *close* for activity 8 are activities 1 and 21 respectively. For activity 33, the *open* and *close* are activities 31 and 35.

4.2 Main loop

In each iteration of the algorithm main loop, the function $findSchedule$ tries to find the schedule with the given upper bound while the number of steps is limited by the parameter *budget*, which is usually set as a number of activities multiplied by the parameter *budgetRatio*. If a feasible schedule is found, all activities are shifted to the left by the label-correcting algorithm [7] so that the constraints and the order of activities in S are kept. A new upper bound of the schedule length is computed as $C^{UB}_{max} = s_{n+1} - 1$ and the next iteration of the loop is performed. If a feasible schedule S is not found for the given schedule length, the algorithm modifies the priority according to the returned partial schedule.

A general observation for heuristic algorithms is that more incorrect decisions are made at the beginning and, therefore, the priority of the earliest scheduled activities and activities that have been added to the schedule more often is decreased. The function $findSchedule$ is then called for the same upper bound C^{UB}_{max} using the modified priority. If the schedule was not found and the maximum number of priority modification steps determined by the parameter $maxModifications$ is exhausted, the algorithm returns the best schedule.

Alg 2 Inner loop of IRSA

```
$findSchedule\left(C^{UB}_{max}, priority, budget\right)$
 $scheduled = \{\}$ ;
 $nAdds_i = 0 \; \forall i \in \mathcal{T}$;
 $s_i = 0 \; \forall i \in \mathcal{T}$;
 $v_i = 0 \; \forall i \in \mathcal{T}$;
 while $budget \geq 0$
  $priority = updatePriority\left(priority, nAdds, v\right)$;
  $k = \max_{\forall j \in scheduled}\left(priority_i\right)$ ;
  $ES_k = \max_{\forall j \in scheduled}\left(s_j + l_{kj}\right)$ ;
  $LS_k = C^{UB}_{max} - p_k$;
  $[conflicts, s_k] = findSlot\left(k, scheduled, ES_k, LS_k\right)$ ;
  $nAdds_k = nAdds_k + 1$;
  $[s, scheduled] = insertActivity\left(k, s_k, conflicts\right)$ ;
  $v = findSelected\left(v, scheduled\right)$ ;
  if schedule is complete
   return $S$;
  end
  $budget = budget - 1$;
 end
 return $S$;
end
```

4.3 Inner loop

In the inner loop of the IRSA, priorities are updated in the function *updatePriority* (see Alg 2) such that the priority is increased for the activities marked as selected and proportionally decreased to the number of inclusions of the activity to the schedule. Activity k with the highest priority is found among the not yet scheduled activities and a time window $\langle ES, LS \rangle$ where activity k can be scheduled is computed. The earliest start time ES is calculated as the minimum time such that all temporal constraints $s_j + l_{jk} \leq s_k$ for all $j \in scheduled$ are satisfied, where *scheduled* is a set of activities that forms the current partial schedule. The latest start time LS is set to the maximal value such that the activity is completed before the given C^{UB}_{max}. The function $findSlot$ tries to find the earliest time slot within the given time window with respect to the resource constraints. If no feasible time position is found, then the time slot is set to ES and the function returns all conflicting activities. Activity k is then inserted into the partial schedule and all activities marked as conflicting are removed in order to keep the partial schedule feasible at any time. The list of the selected/rejected activities is then updated; the scheduled activities are marked as selected, activities whose selection explicitly results from equations (4-6), with respect to the scheduled activities, are also marked as selected activities and all activities that cannot be added to the schedule without violating equations (4-6) are marked as rejected

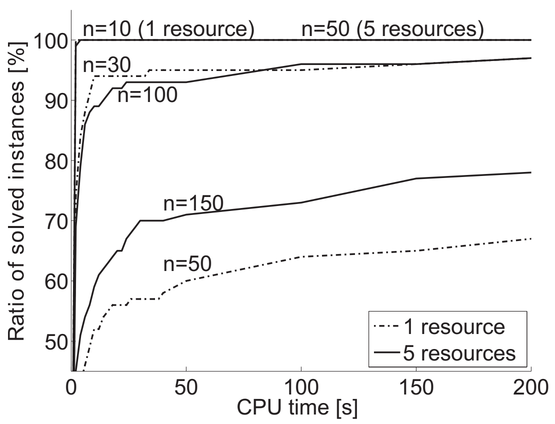

Figure 1. Ratio of solved instances for ILP

activities. The selection/rejection of other activities is not decided yet. If each activity is marked as scheduled or rejected, then the schedule S is complete.

5 Computational Experiments

In this section, the performance evaluation for the integer linear programming model proposed in Section 3 and the heuristic algorithm IRSA is shown. Up to our knowledge, there are no standard benchmarks for the $PSm, 1, 1|nestedAlt, temp, o_{ij}|C_{max}$ problem, hence randomly generated instances have been used. Experiments were performed on a PC with 2x Intel Core 2 Quad CPU at 2.83GHz with 8GB of RAM. To solve ILP the problems, the ILOG CPLEX 11.2 was used, the IRSA algorithm was implemented in Matlab R2008a.

5.1 Mathematical Model Complexity

Each randomly generated instance contains Nested Temporal Graph with Alternatives, where a ratio of alternative and parallel branchings can be specified. The maximum value of the activity processing time, the number and lengths of the positive and negative time-lags and the number of resources can be also set for each generated instance.

The integer linear programming model was tested with the processing time chosen from a uniform distribution on the interval $\langle 1, p_{max} \rangle$ where $p_{max} = 15$, the number of time-lags was set to $|E(G^{prec})| + n/3$ for the positive values and $n/5$ for the negative values, where $|E(G^{prec})|$ is the number of edges in graph G^{prec} defined in Section 2. Figure 1 shows the number of solved instances in dependence on the solving time. 500 random instances have been generated for each number of activities n. We measured how many instances the ILP solver was able to solve within a given amount of time. As ons can see, the larger instances can be solved with the increasing number of dedicated resources. This is obvious due to the smaller amount of resource constraints to be resolved.

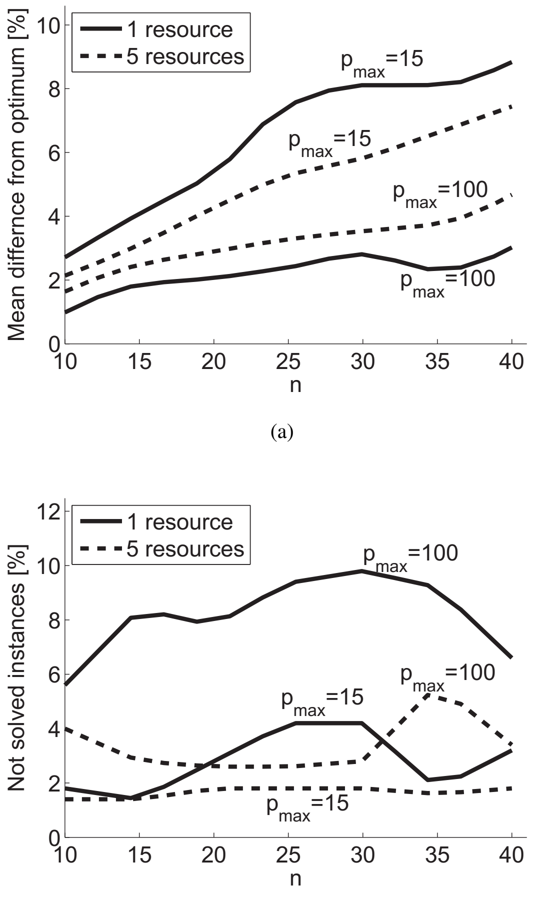

Figure 2. Performance of IRSA algorithm

5.2 Performance Evaluation of IRSA algorithm

The IRSA algorithm was tested on 500 feasible instances for each number of activities and resources. The parameters of the algorithm were set to $budgetRatio = 6$ and $maxModifications = 2$. The generator of the instances has default settings as in the evaluation of the ILP model. Figure 2(a) shows the mean difference of the IRSA algorithm from the optimal value given by the ILP solver for feasible instances. Problems with the different number of resources and different maximum processing times are considered. Figure 2(b) demonstrates the number of feasible instances (out of 500) that IRSA was not able to solve. The number of resources does not have an important influence on the results obtained by scheduling with the IRSA algorithm. Instances with shorter processing times of activities lead to results with a bigger difference from the optimal value. On the other hand, the number of instances for which the IRSA was not able to find a solution is slightly lower in the case with shorter processing times.

6 Case Study

A case study of production scheduling where we applied the IRSA algorithm is presented in this section. One type of wire harness made in Styl Plzeň is taken for an illustration and its production process is transformed to an instance of the $PSm, 1, 1|nestedAlt, temp, o_{ij}|C_{max}$ problem. The production process consists of two main phases. First, each wire that will be a part of the final product is marked, cut, stripped and furnished with a terminator in the machine section. Operations can be performed either on one of two universal machines capable to perform all four operations as one indivisible activity or separately on specialized semi-automated machines. In the second phase, the product is manually completed and packed for shipping.

Due to the various number of activities in each process plan, we cannot use the multi-mode resource constrained project scheduling problem [11]. Each operation can require more than one resource at a time, e.g. setting a terminator on a wire requires one machine and one applicator head. Our IRSA algorithm can handle only mono-resource activities, hence multi-resource activities are represented by an appropriate number of mono-resource activities with the same processing time, different resource specification and synchronization by time-lags (time-lags $l_{ij} = 0$ and $l_{ji} = 0$ lead to $s_i = s_j$). The setup times are used to represent the time needed to change the settings of some machine between processing different types of wires.

The instance of Nested Temporal Graph with Alternatives in Figure 3 represents the part of the wire harness production in the machine section. The harness consists of four different types of wires. Wire type 1 can be produced by four alternative ways in the machine part (activities 1-21). Wires of type 2 (activities 22-30), 3 (activities 31-39) and 4 (activities 40-48) have a similar process of production, where marking, cutting and stripping can be performed in two different ways using one universal machine or two semi-automated ones. All wires have to be processed in the machine section before they can be completed by manual labor and packed for transport (represented only by activity 49 due to lucidity).

Each edge of the graph is labeled by a pair of numbers, where the first one is the time-lag constraint and the second one is the type of edge (see Section 2.2). Edges of type 1 and 2 are plotted as solid lines and edges of type 0, i.e. the additional time-lags, are plotted as dotted lines. Based on the assumption that $l_{0i} \geq 0$ for all $i \in 1 \ldots n$, one could expect the edge of type 0 from node 0 to each other node of the graph, but inclusion of all edges to the graph would be disorienting. Moreover, if we assume that release times of all activities are equal to zero, then the time-lag constraints l_{0i} of all activities are satisfied by constraining activities $1, 22, 31$ and 40 only. The similar situation is for time-lags $l_{i\,n+1}$ since they are implicitly satisfied by $l_{49\,n+1}$.

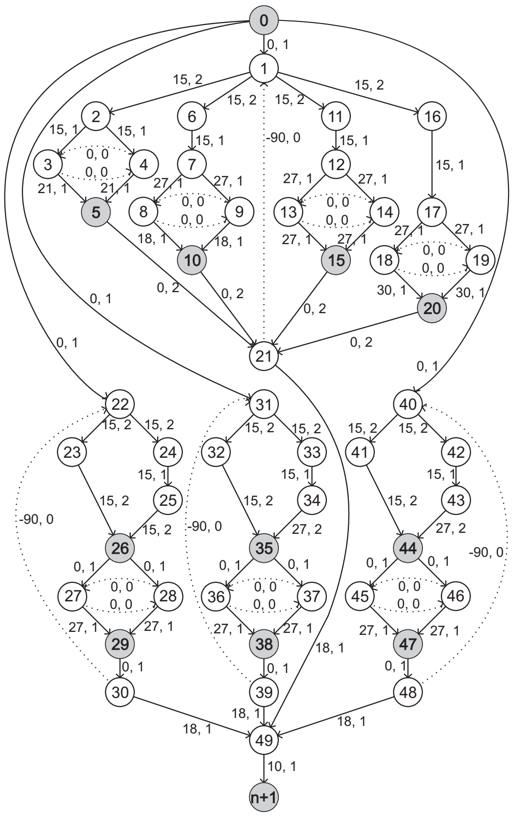

Figure 3. Case study graph instance

The fact that all four wires have to be processed before the harness can be completed is represented by the output parallel branching from node 0 consisting of four edges of type 1. As mentioned before, the wire of type 1 can be produced by four alternative ways, where only one of them has to be chosen. Therefore, node 1 corresponding to the preparation of the material for wire 1 has the output alternative branching consisting of four edges of type 2. Later on, there is the input alternative branching for node 21.

The nodes of the graph filled with gray color correspond to dummy activities, which are added to the problem to keep the instance in the nested form. Such dummy activities have zero processing time and they do not require any resource. The edge from node 21 to node 1 represents the demand that the production process of one wire has to be completed at most 90 minutes after it is started. This edge with the negative time-lag value ensures that the unfinished products are not accumulating in the area.

The activities have the following resource demands: activities 1, 22, 31 and 40 are the preparative operations and they require the same operating staff. Activities 2, 6, 11, 16, 24, 33 and 42 correspond to marking of wires performed on the specialized marking unit. Activities 7, 12, 17, 25, 34 and 43 stand for the cutting and stripping of wires on the multipurpose machine, called KOMAX, capable to perform marking, cutting, striping and for some

Act	p_i	θ_i	Act	p_i	θ_i	Act	p_i	θ_i
0	0	-	17	27	3	34	27	3
1	15	1	18	30	6	35	0	-
2	15	2	19	30	6	36	27	4
3	21	3	20	0	-	37	27	6
4	21	3	21	18	7	38	0	-
5	0	-	22	15	1	39	18	7
6	15	2	23	15	3	40	15	1
7	27	3	24	15	2	41	15	3
8	18	3	25	27	3	42	15	2
9	18	6	26	0	-	43	27	3
10	0	-	27	27	4	44	0	-
11	15	2	28	27	6	45	27	4
12	27	3	29	0	-	46	27	6
13	27	4	30	18	7	47	0	-
14	27	6	31	15	1	48	18	7
15	0	-	32	15	3	49	21	8
16	15	2	33	15	2	$n+1$	0	-

Table 1. Numerical data for case study

$$o_{ij} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 2 & 0 & 0 & 3 & 2 & \cdots \\ 0 & 0 & 1 & 0 & 0 & 0 & 3 & 2 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 1 & \cdots \\ 0 & 0 & 3 & 2 & 0 & 0 & 3 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Figure 4. A part of the setup times matrix

wires also to furnish them with the terminator. Activities 23, 32, and 41 represent marking, cutting and stripping on the KOMAX machine. A pair of activities 3-4 correspond to cutting, stripping and furnishing with the terminator for the wire on KOMAX. Since the terminator needs a special applicator together with the KOMAX machine to be connected to the wire, activity 3 is dedicated to KOMAX and activity 4 require the applicator and they are synchronized by time-lags. The same situation is for pairs of activities 8-9, 13-14, 18-19, 27-28, 36-37 and 45-46 representing application of the terminator for wires. The only difference is in the resource specification of these activities. Finally, activities 21, 30, 39 and 48 represent the transport of wires to the manual section of the manufacture. Processing times of activities and their resource specification are in Table 1. A part of the matrix with setup times is illustrated in Figure 4.

The optimal solution of the problem is shown in Figure 5. As we can see, not all activities are included in the schedule. Such an instance contains 51 activities that have to be scheduled on a set of 9 resources. Using the ILP model proposed in Section 3, the optimal solution has been found in 0.35s. The same solution was obtained in 0.25s by the IRSA algorithm. If two similar instances

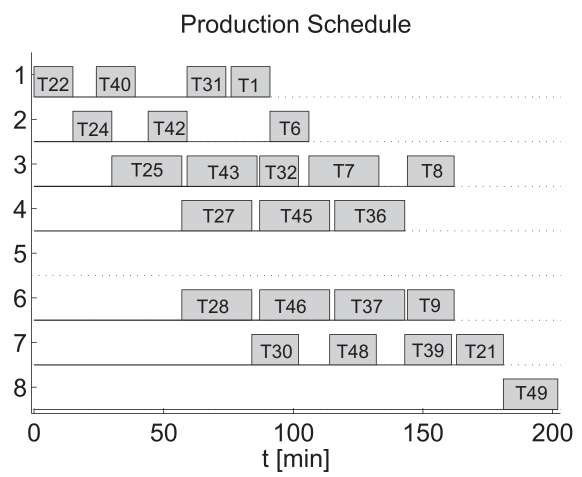

Figure 5. Case study solution

(i.e. two independent orders for the same wire harness) are considered and joined into one instance, then the number of activities raises to 100 and the optimal solution is not found within 48 hours of the ILP solver runtime. On the other hand, IRSA can find the solution in 0.8s, while the difference from the best known solution (found by the ILP solver with 60s runtime limit) is 8%. For 10 joined instances, the number of activities is 492 and the IRSA algorithm returns the solution in 8.6s while the difference against the solution found by the ILP solver with 600s runtime limit is 6.5%.

7 Conclusion

We have presented the production scheduling problem with alternative process plans, motivated by the production of the wire harnesses in Styl Plzeň. We have decided to represent the structure of the problem by Nested Temporal Graph with Alternatives. The mathematical model able to solve instances with up to 50 activities for the case with one resource is presented. In order to solve larger problems we have developed the heuristic algorithm IRSA implemented in the Matlab environment. Computational experiments demonstrate good performance of this algorithm with a mean difference from the optimal value of the makespan less than 10%, while solving time for instances with 100 activities is less than 1s. Instances with up to 2000 activities can be solved in the order of a few minutes. From our experience with the IRS algorithm [12], we can suppose that the implementation of IRSA in C# language can be up to 40 times faster than the one in Matlab. The IRSA algorithm is successfully applied on the production of wire harnesses in Styl Plzeň.

8 Acknowledgements

This work was supported by The Czech Science Foundation under Project P103/10/P401 and by the Ministry of Education of the Czech Republic under projects OE09004 and 1M0567.

A List of variables

n	number of activities
m	number of resources
i, j, k	indices
$\mathcal{T}$	set of activities, equal to set of nodes in graph G
$\mathcal{R}$	set of resources
p_i	processing time
r_i	release time
$\tilde{d}_i$	deadline
l_{ij}	general temporal constraint
u_{ij}	the longest path between activities i and j
θ_i	resource dedication
o_{ij}	setup time
e_{ij}	edge between nodes i and j
a_{ij}	type of edge between nodes i and j
C_{max}	schedule length
v_i	activity selected/rejected
s_i	start time
x_{ij}	binary decision variable of the ILP model
UB	positive constant
S	schedule

References

[1] R. Barták and O. Čepek. Temporal networks with alternatives: Complexity and model. In *Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference (FLAIRS), Florida, USA*, pages 641–646. AAAI Press, 2007.

[2] R. Barták and O. Čepek. Nested temporal networks with alternatives: recognition and tractability. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Ceara, Brazil*, pages 156–157. ACM, 2008.

[3] J. C. Beck and M. S. Fox. Constraint-directed techniques for scheduling alternative activities. *Artificial Intelligence*, 121:211–250, 2000.

[4] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling Computer and Manufacturing Processes*. Springer-Verlag New York, Inc., 1996.

[5] P. Brucker. *Scheduling Algorithms*. Springer-Verlag New York, Inc., 2007.

[6] P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112:3–41, 1999.

[7] P. Brucker and S. Kunst. *Complex Scheduling*. Springer-Verlag New York, Inc., 2006.

[8] L. Capacho and R. Pastor. The ASALB problem with processing alternatives involving different tasks: Definition, formalization and resolution. In *International Conference Computational Science and Its Applications (ICCSA), Glasgow, UK*, pages 554–563. Springer, 2006.

[9] L. Capacho and R. Pastor. ASALBP: the alternative subgraphs assembly line balancing problem. *International Journal of Production Research*, 46:3503–3516, 2008.

[10] L. Capacho, R. Pastor, A. Dolgui, and O. Guschinskaya. An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*, 15:109–132, 2009.

[11] B. De Reyck and W. Herroelen. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119:538–556, 1999.

[12] Z. Hanzálek and P. Šůcha. Time symmetry of project scheduling with time windows and take-give resources. In *4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA), Dublin, Ireland*, pages 239–253. Springer, 2009.

[13] T. Kis. Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, 151:307–322, 2003.

[14] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer-Verlag, 2008.

[15] B. R. Rau. Iterative modulo scheduling: an algorithm for software pipelining loops. In *MICRO 27: Proceedings of the 27th annual international symposium on Microarchitecture*, pages 63–74. ACM, 1994.

[16] N. Trautmann and C. Schwindt. Large-scale short-term planning in chemical batch production. In *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, pages 490–497. Springer, 2007.