European Journal of Operational Research 249 (2016) 577–591

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An approach using SAT solvers for the RCPSP with logical constraints

Mario Vanhoucke [a,b,c,*], José Coelho [d,e]

[a] Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium
[b] Vlerick Business School, Reep 1, 9000 Gent, Belgium
[c] UCL School of Management, University College London, Gower Street, London WC1E 6BT, United Kingdom
[d] INESC – Technology and Science, Porto, Portugal
[e] Universidade Aberta, Rua da Escola Politécnica, 147, 1269-001 Lisbon Portugal

ARTICLE INFO

Article history:
Received 12 July 2014
Accepted 28 August 2015
Available online 9 September 2015

Keywords:
Project scheduling
RCPSP
AND/OR/BI constraints
SAT

ABSTRACT

This paper presents a new solution approach to solve the resource-constrained project scheduling problem in the presence of three types of logical constraints. Apart from the traditional AND constraints with minimal time-lags, these precedences are extended to OR constraints and bidirectional (BI) relations. These logical constraints extend the set of relations between pairs of activities and make the RCPSP definition somewhat different from the traditional RCPSP research topics in literature. It is known that the RCPSP with AND constraints, and hence its extension to OR and BI constraints, is NP-hard.

The new algorithm consists of a set of network transformation rules that removes the OR and BI logical constraints to transform them into AND constraints and hereby extends the set of activities to maintain the original logic. A satisfiability (SAT) solver is used to guarantee the original precedence logic and is embedded in a metaheuristic search to resource feasible schedules that respect both the limited renewable resource availability as well as the precedence logic.

Computational results on two well-known datasets from literature show that the algorithm can compete with the multi-mode algorithms from literature when no logical constraints are taken into account. When the logical constraints are taken into account, the algorithm can report major reductions in the project makespan for most of the instances within a reasonable time.

1. Introduction

Resource-constrained project scheduling is a widely discussed project management topic which has roots in and relevance for both academic and practical oriented environments. Due to its inherent problem complexity, it has been the subject of numerous research projects leading to a wide and diverse set of procedures and algorithms to construct resource feasible project schedules. Thanks to its practical value, many of the research results have found their way to practical project management projects. From the initial attempts to construct project schedules using a quantitative scheduling approach, as proposed by the PERT/CPM methods, many extensions have been proposed to make the constructed project schedules more realistic.

The presence of renewable resource constraints with limited availability has led to the so-called resource-constrained project scheduling problem (RCPSP) which is one of the most investigated scheduling problems in the project management literature. Resource-constrained project scheduling is the process of constructing a project schedule within the limited amount of resources available. It requires the examination of the possible unbalanced use of resources over time to resolve over-allocations (the so-called resource conflicts) when more resources are required than available. The critical path based scheduling methods will often schedule certain activities simultaneously, but when more resources such as machines or people are needed than there are available, these activities will have to be rescheduled concurrently or even sequentially to resolve the resource constraints. Resource-constrained project scheduling is the process of resolving these resource conflicts such that the total project makespan is minimized.

In this paper, the well-known RCPSP is extended with two classes of logical constraints to model new relations between pairs of activities. Furthermore, a solution approach, based on the principles used by Coelho and Vanhoucke (2011, 2014) is proposed to solve this new problem to near-optimality in reasonable time. The paper is organized as follows. In Section 2, a time-indexed formulation for the

* Corresponding author at.: Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium. Tel.: +329643569.
E-mail address: mario.vanhoucke@ugent.be (M. Vanhoucke).

http://dx.doi.org/10.1016/j.ejor.2015.08.044
0377-2217/

well-known RCPSP is given and the AND precedence relations are extended to OR and bidirectional (BI) constraints. Section 3 reports on the new solution procedure to solve the problem heuristically in reasonable time, and includes the network transformation rules, the use of the satisfiability (SAT) solver and the metaheuristic search for resource feasible solutions. In Section 4, the design for a new public dataset is described and computational results for all the experiments are reported. In Section 5, general conclusions are drawn and paths for future research are briefly discussed.

2. Problem statement

The RCPSP can be stated as follows. A set of activities N, numbered from a dummy start node 0 to a dummy end node $n+1$, is to be scheduled without pre-emption on a set R of renewable resource types. Each renewable resource $k \in R$ has a constant availability a_k per period. Each nondummy activity $i \in N$ has a deterministic duration d_i and requires $r_{i,k}$ units of resource type $k \in R$. The start and end dummy activities representing the start and completion of the project have a duration and a renewable resource requirement equal to zero. A project network is represented by a topological ordered activity-on-the-node (AoN) format where A is the set of pairs of activities between which a finish-start precedence relationship with time lag 0 exists. We assume graph $G(N, A)$ to be acyclic. A schedule S is defined by a vector of activity start times and is said to be feasible if all precedence and renewable resource constraints are satisfied. The objective of the problem type is to find a feasible schedule within the lowest possible project makespan, and hence, the problem type can be represented as $m, 1T|cpm|C_{\max}$ using the classification scheme of Herroelen, Demeulemeester, and De Reyck (1999) or as $PS|prec|C_{\max}$ following the classification scheme of Brucker, Drexl, Möhring, Neumann, and Pesch (1999).

A time-indexed formulation of the RCPSP using the binary variable x_{it} that is equal to 1 if activity i is started at time instance t, and 0 otherwise, can be stated as follows (Pritsker, Watters, & Wolfe, 1969):

$$\text{Minimize} \sum_{t=es_{n+1}}^{ls_{n+1}} t x_{n+1,t} \tag{1}$$

subject to

$$\sum_{t=es_i}^{ls_i} (t+d_i) x_{i,t} \leq \sum_{t=es_j}^{ls_j} t x_{j,t} \quad \forall (i,j) \in A \tag{2}$$

$$\sum_{t=es_i}^{ls_i} x_{i,t} = 1 \quad \forall i \in N \tag{3}$$

$$\sum_{i=1}^{n} r_{i,k} \sum_{s=\max(t-d_i, es_i)}^{\min(t-1, ls_i)} x_{i,s} \leq a_k \quad \forall k \in R \text{ and } t = 1, \ldots, T \tag{4}$$

$$x_{i,t} \in \{0,1\} \quad \forall i \in N; t = 1, \ldots, T \tag{5}$$

where es_i and ls_i are used to denote the earliest and latest start for activity i given the project upper bound T using traditional forward and backward critical path calculations. Eq. (1) minimizes the total project makespan. Eq. (2) takes the finish-start precedence relations with a time-lag of zero into account. Eq. (3) guarantees that each activity can only start once at a certain time t. The renewable resource constraints are satisfied thanks to Eq. (4). Eq. (5) forces the decision variables to be binary values.

2.1. Logical precedence constrains

While most of the resource-constrained project scheduling algorithms focus on the traditional AND constraint, different types of logical constraints have been defined in literature. In this paper, three types of precedence relations between activities will be used to construct a resource feasible schedule. To that purpose, the set of precedence relations A will be split up into three disjoint sets A^{and}, A^{or} and A^{bi} to define these types of activity precedences. In the following subsections, Eq. (2) will be replaced by five new equations to model these three precedence classes.

2.1.1. AND constraints

Most algorithms to solve the RCPSP make use of an activity network with precedence relations with minimal and/or maximal time-lags that force an activity to start/finish no earlier than the start/finish of *all* its predecessor activities. Ever since the research on improved algorithms to solve resource-constrained project scheduling problem has started, a wide variety of problem formulations have been proposed in literature. For an overview of the research on the RCPSP, the reader is referred to the surveys of Herroelen, De Reyck, and Demeulemeester (1998), Icmeli, Erengüç, and Zappe (1993), Kolisch and Hartmann (2006), Özdamar and Ulusoy (1995) and Hartmann and Briskorn (2010), the classifications schemes of Brucker et al. (1999) and Herroelen et al. (1999) and the overview handbooks written by Demeulemeester and Herroelen (2002) and Vanhoucke (2012).

The traditional so-called *AND* constraints used in almost all problem formulations have been modelled in Eq. (2) as finish-to-start relations with a minimal time-lag of zero. In this paper, the set of AND constraints is denoted by the symbol A^{and} and hence, Eq. (2) can now be formulated as follows:

$$\sum_{t=es_i}^{ls_i} (t+d_i) x_{i,t} \leq \sum_{t=es_j}^{ls_j} t x_{j,t} \quad \forall (i,j) \in A^{\text{and}} \tag{6}$$

2.1.2. OR constraints

The *OR precedence relations* differ from the traditional AND constraints discussed in the previous section that they only require one predecessor to finish before the successor activity can start. Consequently, for an activity with multiple predecessors, a choice has to be made which predecessor will be the dominant one to force the activity not to start earlier. Unlike the AND constraints, all other predecessors can have an overlap with the activity. Scheduling tasks with and/or constraints are discussed in Belhe and Kusiak (1995), Gillies and Liu (1995), Adelson-Velsky and Levner (2002) and Möhring, Skutella, and Stork (2004). In order to define the set of OR constraints A^{or}, Möhring et al. (2004) have defined a set of waiting activities $N^w \subset N$ consisting of activities $j \in N^w$ for which all pairs $(P_j^{\text{or}}, j) \in A^{\text{or}}$ imply that activity j can only start after the finish of at least one activity $i \in P_j^{\text{or}}$. The set P_j^{or} (index $l = 1, \ldots, |P_j^{\text{or}}|$) is defined as the set of predecessor activities of waiting activity j linked by an OR constraint. A new binary decision variable y_{jl} is defined for each OR relation for each waiting activity j that is used in a big M constraint to model this new precedence relation, as follows:

$$\sum_{t=es_i}^{ls_i} (t+d_i) x_{i,t} \leq \sum_{t=es_j}^{ls_j} t x_{j,t} + M y_{jl} \quad \forall (P_j^{\text{or}}, j) \in A^{\text{or}}; l = 1, \ldots, |P_j^{\text{or}}| \tag{7}$$

$$\sum_{k=1}^{|P_j|^{\text{or}}} y_{jl} \leq |P_j^{\text{or}}| - 1 \quad \forall j \in N^w \tag{8}$$

Eq. (7) models the precedence relations with an additional big M to force OR relation between two activities. Only when $y_{jl} = 0$, the constraint forces activity j to start after the finish of its predecessor i. Eq. (8) assures that for each waiting activity j at least one y_{jl} is equal to zero according to the definition of the OR constraint.

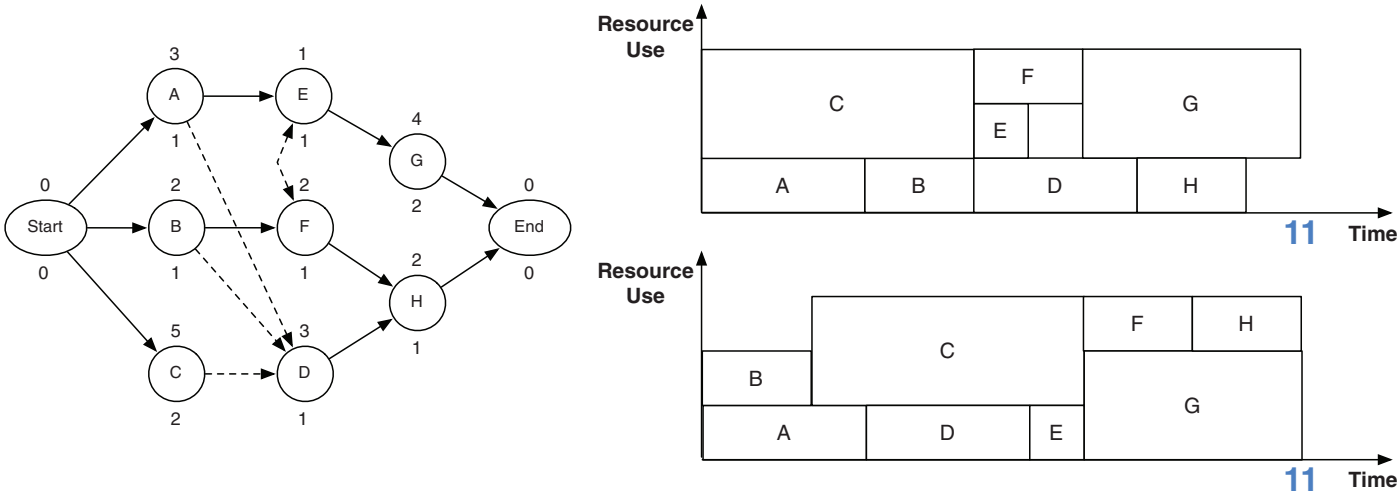


Fig. 1. An example project network and the optimal schedule.

2.1.3. Bidirectional constraints

A second extension of the traditional AND precedence relations is the presence of *bidirectional precedence constraints* that simple stipulate that pairs of activities cannot be executed in parallel, i.e. either the first one has to be executed in order to start the second one, or vice versa. A typical example where the precedence between activities is defined by a bidirectional precedence relation is where both activities make use of the same resource. Often, a (sequence-dependent) changeover time between the two activities accompanies this constraint, as a result of transferring the resource to another place where the execution of the successor activity takes place. This changeover time γ_{ij} is defined as the minimal time needed between the finish of activity i and the start of activity j to transfer and possibly set up the resource to be used by activity j. To that purpose, a new binary variable z_{ij} is defined to model that two activities cannot be executed in parallel, but should be scheduled in a sequence, with at least a minimal time interval to include the changeover times. Sequence-dependent changeover times have been mentioned in papers by Drexl, Nissen, Patterson, and Salewski (2000) and Herroelen, Demeulemeester, and De Reyck (2001).

$$\sum_{t=es_i}^{ls_i}(t+d_i+\gamma_{ij})x_{i,t} \le \sum_{t=es_j}^{ls_j} t x_{j,t} + M z_{ij} \quad \forall (i,j) \in A^{\text{bi}} \tag{9}$$

$$\sum_{t=es_j}^{ls_j}(t+d_j+\gamma_{ji})x_{j,t} \le \sum_{t=es_i}^{ls_i} t x_{i,t} + M(1-z_{ij}) \quad \forall (i,j) \in A^{\text{bi}} \tag{10}$$

Eq. (9) models the precedence relation between activities i and j and is only relevant in case the z_{ij} is equal to zero. Otherwise, if $z_{ij} = 1$, this equation becomes redundant and Eq. (10) then forces activity i to start after the finish of activity j.

2.2. Illustrative example

Fig. 1 displays an example project network in an activity-on-the-node format with 8 nondummy activities and one renewable resource with a limited availability of 3 units. The durations of all activities are displayed above each node while the resource demand is given below each node. All arcs are traditional AND relations with a minimal time-lag of zero, unless otherwise indicated. More precisely, activity D has a set of OR relations with its predecessors as indicated by the dotted arcs. A bi-directional relation between activities E and F exists, as indicated by the bi-directional dotted arc, and it is without loss of generality assumed that the changeover times are equal to zero, i.e. $\gamma_{EF} = \gamma_{FE} = 0$.

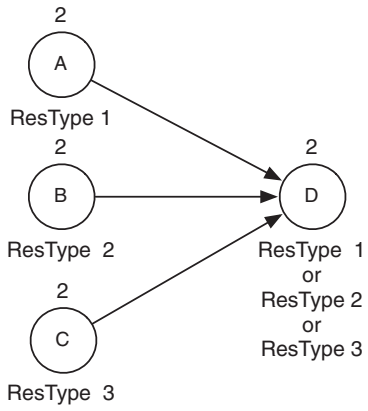
The schedule displayed on the top shows the optimal schedule in which all relations are traditional AND relations, leading to a makespan of 11 units. The schedule displayed at the bottom respects the OR relations of activity D and the bi-directional relationship between activities E and F. Its makespan is also 11 time units, but activity D starts before the finish of activity C, but after A and B. Moreover, E and F do not overlap, but instead F is scheduled after E. In the computational experiment section, the impact of adding OR and BI relations on the project makespan will be investigated under various settings and for various values of the changeover times.

2.3. Practical relevance

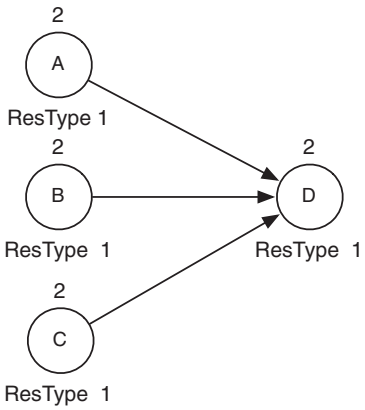
While most of the projects make use of the traditional AND constraints, the use and relevance of logical constraints have been mentioned and illustrated in the literature using various project examples. In Hartmann and Briskorn (2010), examples are given for "exclusive OR" (XOR) relations, which can be seen as a special case of OR constraints, for an R&D project in which only one activity must be executed, either activity "modify prototype" or "start production" after the execution of the predecessor activity "test prototype" and the choice of the activity depends on the outcome of this predecessor activity. Other examples appear in literature, such as Kuster and Jannach (2006) who refers to the use of logical constraints to model mutual exclusion and inclusion relationships between activities, Belhe and Kusiak (1995) who make use of hierarchically structured design activity network with AND, OR and XOR relations, as well as the book by Berthaut (2011) who shows that overlapping modes can be generalized to precedence modes in order to describe all precedence relationships between activities.

A straightforward practical application of OR constraints is given in the first example project network displayed on the left of Fig. 2 that shows three activities A, B and C that each make use of a different resource type (denoted as ResType x with x equal to 1, 2 or 3). However, activity D is a merging activity that can only be started once one of its predecessors is finished, and should therefore be modelled by OR constraints between activity D and its predecessors. The reason is that activity D can be executed by each of the resource types used by the previous activities, and its start depends on the earliest release of one of the resource types 1, 2 or 3.

BI constraints are more intuitive to understand since they mainly involve activities that make use of the same set of resources and can therefore not be executed in parallel. While this can be easily mod-



Example 1.
Activity D can be executed by one of three available resource types



Example 2.
Activities A, B and C make use of the same resource types and can therefore not executed in parallel. However, sequence-dependent setup times exist between the activities

Fig. 2. Two small example networks.

elled with traditional renewable resource constraints, the presence of sequence-dependent changeover times makes it an ideal problem to be modelled with BI constraints. In the second example on the right of Fig. 2, activities A, B and C make use of the same resource type 1, and changeover times are assumed to be sequence-dependent, i.e. $\gamma_{AB} \neq \gamma_{BA}$, $\gamma_{AC} \neq \gamma_{CA}$ and $\gamma_{BC} \neq \gamma_{CB}$. These activities should therefore be scheduled in a sequence which clearly depends on the values of these changeover times and the start and finish times of other activities in the network. The changeover times in the example can be best seen as setup times for resources when switching between activities. The literature on setup times has been reviewed and summarized in the paper by Hartmann and Briskorn (2010).

3. Solution method

Logical SAT constraints have been successfully used to solve the multi-mode resource-constrained project scheduling problem (MR-CPSP) by Coelho and Vanhoucke (2011, 2014). In these papers, each activity is duplicated in multiple activities according to their number of possible modes. A SAT algorithm is invoked to cope with the selection of modes for activities as well as to take the limited availability of nonrenewable resources into account, while a metaheuristic search procedure to solve the RCPSP aims at constructing a resource feasible schedule. The communication between the *activity list* of the metaheuristic search and the *variable list* used in the SAT search can be done by one single list of activities. Additionally, clause learning helps to prevent the occurrence of recurring conflicts in the SAT search and improves the performance of the algorithm, certainly when the stop criterion is set at large values. In the next section, it will be shown that this general approach can be translated to the RCPSP problem with logical constraints and leads to near optimal solutions for problem instances with reasonable size.

3.1. Network transformations

In order to construct near-optimal schedules for the RCPSP with AND, OR and BI constraints, the activity network will be transformed to an extended network to incorporate the new precedence logic. In doing so, the new network will only contain the traditional AND constraints such that the scheduling problem reduces to a traditional RCPSP which can be efficiently solved by a state-of-the-art (meta)heuristic search method from literature. Moreover, a SAT search will be added on top of this algorithm to guarantee that the logic of the OR and BI precedences is satisfied.

3.1.1. OR constraints

The network transformation for the OR constraints is relatively easy and consists of the duplication up of each waiting activity into several activities. More precisely, each waiting activity j that has $|P_j^{or}|$ predecessors with OR constraints is duplicated into $|P_j^{or}|$ activities. These activities are duplicates of the waiting activity j and have the same duration and resource demand. Each of these duplicated activities is linked by exactly one of the predecessors of waiting activity j using an AND constraint and with all successor activities of the waiting activity.

Fig. 3 shows how OR constraints are translated into an activity network with only AND relations. The waiting activity D has three predecessors with OR constraints, i.e. $|P_D^{or}| = 3$, and a set of successor activities modelled by the single set of nodes S_D. The activity network with both AND (solid arcs) and OR precedence relations (dotted arcs entering activity D) can be transformed into an activity network with only AND constraints by duplicating activity D into a set of activities that is equal to the number of OR relations entering the activity (i.e. $|P_D^{or}|$). In the example, activity D can only start when either activity A, activity B or activity C is finished, and hence, activity D is duplicated into activities D_1, D_2 and D_3. The OR relation is now shifted to the duplicated activities and requires that only one of the duplicated activities D_1, D_2 or D_3 is the real activity D. The SAT logic of Section 3.2 will guarantee that only one of the duplicated activities D_1, D_2 or D_3 will be selected as the real activity with a duration and resource requirement equal to the values of the original activity D, and the two other activities will then be transformed into dummy activities.

3.1.2. BI constraints

The transformation of networks with bidirectional constraints requires the duplication of activities in multiple activities and the addition of extra activities to model the changeover times. More precisely, each set of two activities with bidirectional constraints is duplicated into two times two activities with additional AND relations, and the changeover times are modelled by two extra activities that are put in-between these newly created duplicated activities. The construction of such a subnetwork can be best explained by means of a general example as shown in Fig. 4.

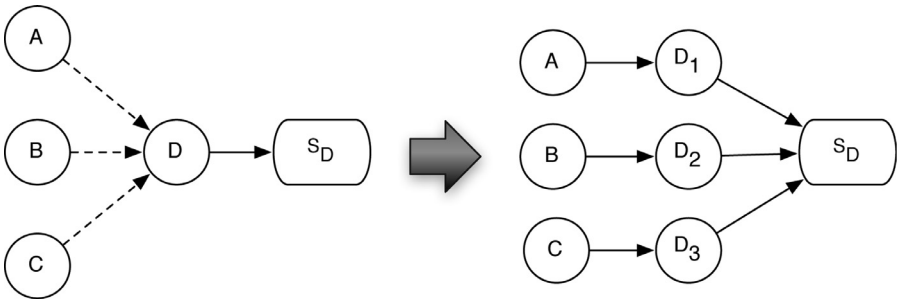


Fig. 3. Transforming OR precedence relations into AND relations.

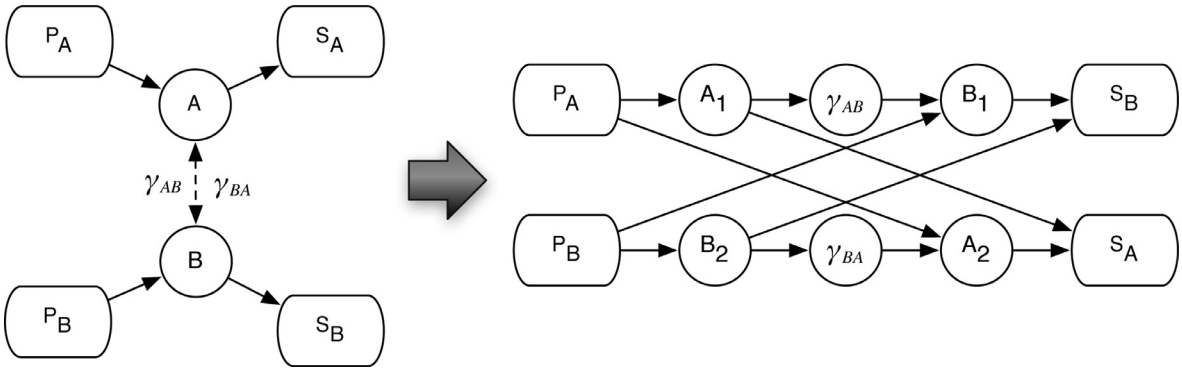


Fig. 4. Transforming BI precedence relations into AND relations.

This figure illustrates how a bidirectional constraint between activities A and B with changeover times γ_{AB} (between A and B) and γ_{BA} (between B and A) can be easily transformed into an activity network with only AND relations. Both activities A and B are duplicated in four new activities A_1, A_2, B_1 and B_2 to model that B_1 can only start after the finish of A_1, or alternatively, A_2 can only start after the finish of B_2. Only one of the two possibilities will occur and the other activities will be set to dummy activities, and this will be guaranteed by logical SAT constraints described in Section 3.2. In between these duplicated activities, a new activity is added with a duration equal to the changeover time γ_{AB} and γ_{BA}, respectively. Finally, extra AND relations are added to guarantee the logical relations between the original A and B activities with their set of successors (S_A and S_B) and predecessors (P_A and P_B).

3.2. SAT constraints

The selection of activities in the transformed network is done using a boolean satisfiability problem (SAT) solver to determine which of the duplicated activities have positive durations and which others are considered to be dummy activities. The activity with the positive duration then inherits the resource requirements of the original activity while the dummy activities do not require resources at all. The boolean satisfiability problem is a well-known decision problem where an expression of boolean variables (referred to as literals) linked by means of the logical operators *and*, *or* and *not* is questioned to be true or false. The problem has been studied extensively in literature (see e.g. the paper by Marques-Silva and Sakallah, 1999, amongst others) and is known to be NP-complete (Cook, 1971). In this section, the literal z_i is defined as a 0/1 variable of activity i to denote whether this activity will have a positive duration (1) or not (0).

OR constraints: The selection of exactly one duplicated activity to satisfy the OR constraint of Fig. 3 can be modelled by the following constraint:

Select only one activity: $z_{D_1} + z_{D_2} + z_{D_3} = 1$

This constraint can be easily translated into the conjunctive normal form (CNF) traditionally used in SAT research that is a conjunction of clauses linked by the "and" operator, as follows:

Select at least one activity: $(z_{D_1} \vee z_{D_2} \vee z_{D_3})$
Select at most one activity: $(\overline{z_{D_1}} \vee \overline{z_{D_2}}) \wedge (\overline{z_{D_1}} \vee \overline{z_{D_3}}) \wedge (\overline{z_{D_2}} \vee \overline{z_{D_3}})$

The second clause should consist of all pairs of activities, or in general, a constraint $z_1 + z_2 + \cdots + z_k = 1$ can be represented in the CNF as $(z_1 \vee z_2 \vee \cdots \vee z_k) \wedge (\overline{z_1} \vee \overline{z_2}) \wedge \cdots \wedge (\overline{z_1} \vee \overline{z_n}) \wedge (\overline{z_2} \vee \overline{z_3}) \wedge \cdots \wedge (\overline{z_{k-1}} \vee \overline{z_k})$.

Since the "select at most one activity" expressions might consist of quite a large number of clauses just for one activity, a special so-called k clause can replace this CNF equation by using only two clauses that can be generally modelled as follows: $(\overline{z_1} + \overline{z_2} + \cdots + \overline{z_k}) \geq k - 1$ resulting in exactly two clauses in total for each activity with OR constraints. The term "k clause" is used to refer to a special type of clause that requires k literals to be satisfied instead of 1 literal as is the case for normal clauses (Coelho & Vanhoucke, 2011).

BI constraints: The selection of pairs of activities to satisfy the BI constraints of Fig. 4 can be modelled using the following constraints:

Select only one start activity: $z_{A_1} + z_{B_2} = 1$
Select B_1 if A_1 is start activity: $z_{A_1} - z_{B_1} = 0$
Select γ_{AB} if A_1 is start activity: $z_{A_1} - z_{\gamma_{AB}} = 0$
Select A_2 if B_2 is start activity: $z_{A_2} - z_{B_2} = 0$
Select γ_{BA} if B_2 is start activity: $z_{B_2} - z_{\gamma_{BA}} = 0$

The translation of this set of clauses into CNF is done as follows:

Select only one start activity: $(z_{A_1} \vee z_{B_2}) \wedge (\overline{z_{A_1}} \vee \overline{z_{B_2}})$
Select B_1 if A_1 is start activity: $(z_{A_1} \vee \overline{z_{B_1}}) \wedge (\overline{z_{A_1}} \vee z_{B_1})$
Select γ_{AB} if A_1 is start activity: $(z_{A_1} \vee \overline{z_{\gamma_{AB}}}) \wedge (\overline{z_{A_1}} \vee z_{\gamma_{AB}})$
Select A_2 if B_2 is start activity: $(z_{A_2} \vee \overline{z_{B_2}}) \wedge (\overline{z_{A_2}} \vee z_{B_2})$
Select γ_{BA} if B_2 is start activity: $(z_{\gamma_{BA}} \vee \overline{z_{B_2}}) \wedge (\overline{z_{\gamma_{BA}}} \vee z_{B_2})$

While the first clause is similar to the clauses used for the AND constraints, the other clauses are new and can be generally modelled as $z_i = z_j$ constraints that can be represented in the CNF as $(z_i \vee \overline{z_j}) \wedge (\overline{z_i} \vee z_j)$.

3.3. General pseudocode

The pseudocode to transform activity networks with OR constraints into activity networks with only AND constraints (see Section 3.1) as well as the incorporation of SAT clauses to select the activities that satisfy the logic of all OR constraints (see Section 3.2) is given below.

$\forall j \in N^w$:
  Duplicate activity j in $|P_j^{or}|$ activities $a_1, \ldots, a_{|P_j^{or}|}$
  For $k = 1, \ldots, |P_j^{or}|$ do
    Precedence relations: Add the following arcs:
      $A^{and} = A^{and} \cup (k, a_k)$
      For $l = 1, \ldots, |S_j|$ do $A^{and} = A^{and} \cup (a_k, l)$
    SAT constraints: Add the following SAT clauses:
      At least one: $(z_{a_1} \vee z_{a_2} \vee \ldots \vee z_{a_k})$
      At most one (k clause): $(\overline{z_{a_1}} + \overline{z_{a_2}} + \cdots + \overline{z_{a_k}}) \geq k - 1$

Similarly, the pseudocode for the BI network transformations and the corresponding SAT clauses is given below.

$\forall (i, j) \in A^{bi}$:
  Duplicate activity i in two activities i_1 and i_2
  Duplicate activity j in two activities j_1 and j_2
  Add two activities to the project network: $N = N \cup \{\gamma_{ij}, \gamma_{ji}\}$
  Precedence relations: $A^{and} = A^{and} \cup (i_1, \gamma_{ij}) \cup (\gamma_{ij}, j_1) \cup (j_2, \gamma_{ji}) \cup (\gamma_{ji}, i_2)$
  Restore successors and predecessors of activities i and j:
    For $k = 1, \ldots, |S_i|$ do $A^{and} = A^{and} \cup (i_1, k) \cup (i_2, k)$
    For $k = 1, \ldots, |S_j|$ do $A^{and} = A^{and} \cup (j_1, k) \cup (j_2, k)$
    For $k = 1, \ldots, |P_i|$ do $A^{and} = A^{and} \cup (k, i_1) \cup (k, i_2)$
    For $k = 1, \ldots, |P_j|$ do $A^{and} = A^{and} \cup (k, j_1) \cup (k, j_2)$
  SAT constraints: add the following SAT clauses:
    Select start activity: $(i_1 \vee j_2) \wedge (\overline{i_1} \vee \overline{j_2})$
    Select relation $(i_1, \gamma_{i,j}, j_1)$: $(i_1 \vee \overline{j_1}) \wedge (\overline{i_1} \vee j_1) \wedge (i_1 \vee \overline{\gamma_{i,j}}) \wedge (\overline{i_1} \vee \gamma_{i,j})$
    Select relation $(j_2, \gamma_{j,i}, i_2)$: $(i_2 \vee \overline{j_2}) \wedge (\overline{i_2} \vee j_2) \wedge (\gamma_{j,i} \vee \overline{j_2}) \wedge (\overline{\gamma_{j,i}} \vee j_2)$

Applying these transformation and clause insertion rules to the activity network of Section 2.2 results in the extended network with 14 nondummy activities, as shown in Fig. 5. The OR relations of activity are duplicated into 3 activities, each with a duration and a resource demand equal to the values of activity D. The BI relations between activities E and F are translated into a subnetwork with four nondummy activities E_1 and E_2 (F_1 and F_2) with a duration and resource demand equal to the original activity E (F), and two additional changeover activities γ_{EF} and γ_{FE} with a duration equal to the changeover times (equal to zero in the example) and no resource demand.

3.4. Metaheuristic search

The metaheuristic search algorithm used to solve the problem instances is based on the algorithm of Coelho and Vanhoucke (2011) that was initially proposed to solve multi-mode resource-constrained project scheduling problem with limited renewable and nonrenewable constraints but without the presence of logical constraints. This search procedure consists of an activity selection algorithm to select activities in networks using a single priority list that can be used to model both the activity list and the mode list, as will be discussed in Section 3.4.2. The search procedure also makes use of an activity scheduling step using a state-of-the-art metaheuristic scheduling algorithm from literature, as will be briefly summarized in Section 3.4.1. In the current paper, this search logic is used to solve two types of resource-constrained project scheduling problems with logical constraints, as follows:

- The resource-constrained project scheduling problem with logical constraints as proposed in the model formulation using Eqs. (1)–(4), and further abbreviated as **RCPSP-Log**.
- The multi-mode resource-constrained project scheduling problem with logical constraints, further abbreviated as **MMRCPSP-Log**. This problem extends the **RCPSP-Log** to the presence of nonrenewable resource constraints as well as the use of multiple modes for each activity.

In order to solve these two problem types, the search procedure of Coelho and Vanhoucke (2011) needed some adaptations which will be described in the following sections. Due to the network transformations to cope with logical constraints, the project networks often increase dramatically in size, and hence, a first adaptation was made to heuristically cope with these large-sized project networks. The extension to multiple modes in the **MMRCPSP-Log** increases the project network size even further, which has led to an activity selection procedure that operates under a truncated search process followed with a greedy search step. Moreover, the incorporation of nonrenewable resource constraints for the multi-mode problem has also led to an additional improvement step to solve infeasible solutions when the nonrenewable resource constraints are not met. Fig. 6 shows a graphical representation of the algorithm to solve the problem. The flow chart consists of two parts, i.e. an activity selection flow to make the selection of activities using a SAT solver to satisfy all the constraints and a scheduling algorithm that aims at scheduling the nondummy (i.e. selected) activities to near-optimality. The algorithm starts with the random generation of a population of activity lists which are first added to the activity selection step to satisfy all constraint, and returns back to the activity scheduling step as a restricted feasible list that enters the initial population of solutions. This process is repeated until the required number of restricted activity lists is in the population set. Then, the algorithm passes to the activity scheduling step which aims at improving the current solutions in the population, using crossover operators and local search processes. Each time the activity scheduling step evaluates a newly constructed solution, a call to the activity selection step is made to check for constraints feasibility. Hence, the algorithm in Fig. 6 sequentially generates new solutions (activity scheduling) and checks constraints feasibilities (activity selection) in order to return a near-optimal solution after a predefined stop criterion. The details of each step are outlined in Sections 3.4.1 and 3.4.2 and example calculations on the extended network of Fig. 5 are shown in Appendix A.

3.4.1. Activity selection

The right part of Fig. 6 graphically displays the adapted SAT search, further referred to as the GSAT heuristic (Selman, Levesque, & Mitchell, 1992), to obtain a feasible selection of activities from the transformed and extended project networks that satisfy all the logical constraints, and, when relevant, the limited amount of nonrenewable constraints. Although not explained in this paper, the presence of limited nonrenewable resources when solving multi-mode project scheduling problems, as will be done in parts of our computational tests, also results in nonrenewable resource constraints clauses which

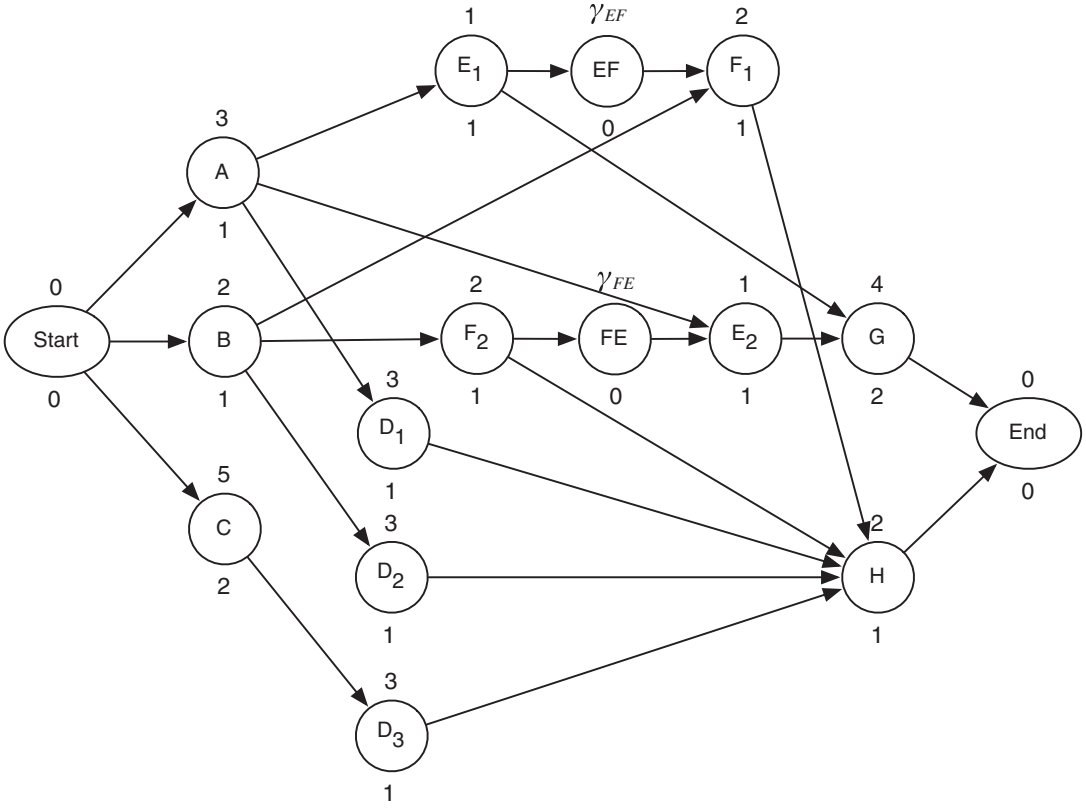


Fig. 5. The transformed network of Fig. 1.

also must be satisfied in a similar way as the clauses for the logical constraints defined in Section 3.2. The construction of the constraint clauses for the nonrenewable resource constraints is based on a simple yet efficient enumeration scheme proposed by Coelho and Vanhoucke (2011).

Fig. 6 starts with a population of activity lists that will be used for both the activity selection and activity scheduling steps, which size is equal to the number of activities in the extended project network after transformations. An activity selection mechanism, displayed at the right of the figure, selects from each activity list only a subset of activities to incorporate the logic of the three types of precedences (for problems **RCPSP-Log** and **MMRCPSP-Log**) as well as the limited nonrenewable resource constraints (for problem **MMRCPSP-Log**) (see the computational experiments of Section 4).

The selection of these activities is done by incorporating a set of SAT clauses as explained in Section 3.3. While in Coelho and Vanhoucke (2011) an optimal SAT solver using the DPLL algorithm of Davis, Logemann, and Loveland (1962) is used which always returns a feasible set of activities from the extended activity list, this approach could no longer be followed for the current paper due to the large size of the projects after the network transformations. Therefore, an alternative approach, referred to as GSAT, is used that works as follows:

- Backtrack limits: The DPLL algorithm assigns values (true/false) to activities and backtracks when infeasibilities occur but this intensive search process is truncated after a predefined number of backtracking limits. Computational experience has shown that this backtracking limit is best set at low values and has been set to 20 in all our experiments.
- Set remaining values: The set of values that have been set by the truncated DPLL are kept fixed, and the remaining values for the variables are assigned in a greedy way to obtain an activity list for which each variable has a value true or false. More precisely, for each unassigned set of variables representing activities in the extended network, exactly one variable is set to true. This resulting SAT solution possibly leads to an infeasible activity list assignment when nonrenewable resources are taken into account (only for problem **MMRCPSP-Log**) since a number of these clauses might be violated (#violations in Fig. 6).
- Improvement step (only for problem **MMRCPSP-Log**): The number of violations in the clauses is reduced by sequentially repeating a greedy search algorithm that swaps the values between activities in the extended network which corresponds to flipping the value of the literature from true/false to false/true. For every swap, the impact on the number of violations is measured, which might lead to an increasing number of clauses with violations (deteriorations are displayed by downward arcs) or a decreasing number of clauses with violations (improvements are represented by upward arcs). After swapping each activity, the swap that has resulted in the best improvement is kept and this process is repeated several times until a stop criterion is met. In our computational experiments, the best value of the stop criterion is set at a repetition of maximum 4 times. Obviously, when no further violations occur, this process is truncated earlier.

This search approach might result in an activity selection which is feasible towards all logical constraints and nonrenewable resource limits, and this remaining activity list with the activities equal to true as the real activity list (all remaining activities become dummy activities) now enters the scheduling step to obtain a feasible and near-optimal solution. In case the remaining activity list still contains a number of violations, a penalty is added to the resulting project makespan which is a function of the number of violations and the algorithm also continues with the activity scheduling step.

3.4.2. Activity scheduling

The population of activity lists generated in the previous step contains a list of activities for which some of the activity durations are set to zero (dummy activities) resulting in restricted activity lists. These lists are now input to a traditional resource-constrained project

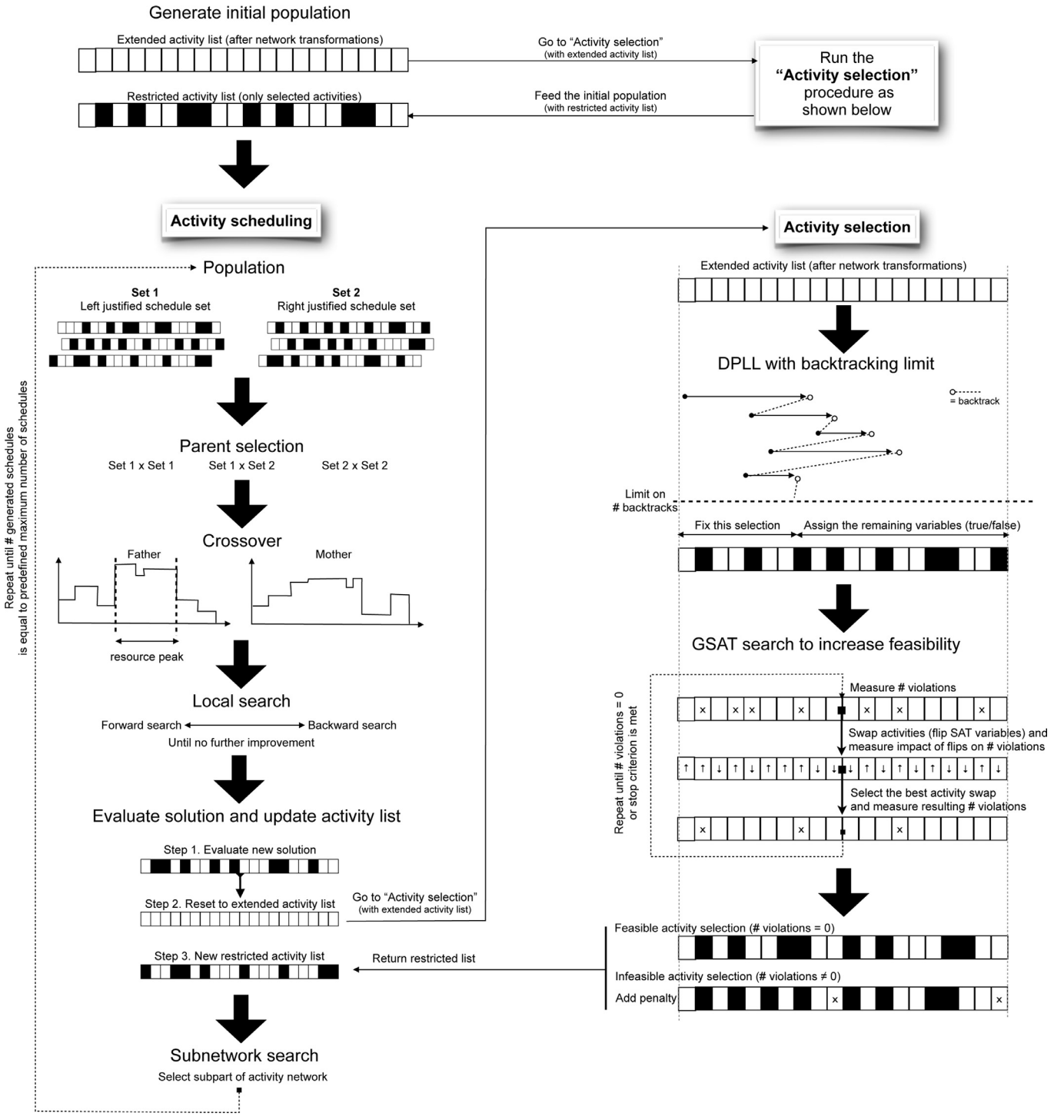


Fig. 6. A graphical representation of the algorithm.

scheduling algorithm to construct a resource-feasible schedule. The way the search process to near-optimal solutions is carried out is based on a collection of principles borrowed from literature that have been proven to perform excellent.

The dual population algorithm, consisting of so-called left- and right-justified schedules is based on the recommendations of Valls, Ballestin, and Quintanilla (2005) and selects parents to construct new child solutions as combinations from both groups using a 2-tournament selection. A two-point crossover operator based on a modified version of the peak crossover operator of Valls, Ballestin, and Quintanilla (2008) using the so-called resource utilization ratio that measures the resource utilization at time unit t allows the selection of time intervals for which the resource utilization is high, so-called peaks, and time intervals with low resource utilization. The local search consists of an iterative forward and backward search of Li and Willis (1992) to improve the two separate populations containing left- and right-justified schedules. This process is initiated on the reduced activity list, and is sequentially repeated on only a subpart of the constructed schedule until a stop criterion is met. This decomposition approach is based on the recommendations of Debels and Vanhoucke (2007) who have shown that an intensive search on subparts of schedules leads to improved solutions.

This metaheuristic search process for activity scheduling is graphically displayed on the left pane of Fig. 6 and contains no fundamentally new techniques compared to the best known performing techniques from literature. Each time the metaheuristic algorithm produces a new activity list, the objective function of this new solution must be evaluated. Prior to each evaluation, the newly generated activity list is passed to the activity selection algorithm as an unrestricted list (setting the dummy activity durations back to their original values) and a new restricted list that satisfies all the constraints is returned to the activity scheduling step. The objective of this new activity list is evaluated (i.e. the feasible schedule obtained by the restricted list), and possibly enters the population. In the computational results section, it will be shown that the combined activity selection and activity scheduling step leads to promising results on two well-known project datasets using different settings for the logical constraints.

4. Computational experience

4.1. Test design

Two datasets have been used to validate the performance of our novel algorithm using resource-constrained project scheduling problems with single-mode and multi-mode activities. The single-mode tests are ran on the single-mode instances of the PSPLIB dataset downloaded from www.om-db.wi.tum.de and presented in Kolisch and Sprecher (1996). All multi-mode tests are done on the triple set MMLIB50, MMLIB100 and MMLIB+ downloaded from www.projectmanagement.ugent.be and presented in Van Peteghem and Vanhoucke (2014), as well as on the multi-mode instances of the PSPLIB.

Since all instances from these datasets only contain AND relations, both OR and BI constraints have been generated under a controlled design. To that purpose, two metrics κ_1 and κ_2 are used to transform some of these AND relations to OR and/or BI relations. These two parameters allow us not only to control the amount of AND relations that will be transformed into OR or BI relations, but also guarantee that future researchers can simply replicate this approach leading to exactly the same networks and the location of the AND/OR/BI relations. The parameter κ_1 determines the start activity that will be selected while parameter κ_2 is used to determine the percentage of activities that will be selected for the transformation from AND to OR constraints. More precisely, activity i will be selected as an activity for which the AND relations with all its predecessor activities will be transformed into OR relations if the following equation $(i+\kappa_1) \bmod \kappa_2 = 0$ holds. As an example, if $\kappa_1 = 3$ and $\kappa_2 = 10$ then the first activity will be $i = 7$ $((7+3) \bmod 10 = 0)$, the second one activity 17, and so on. The selection of bidirectional constraints is done in a similar way. The two parameters κ_1 and κ_2 are used to select activities, and the BI constraint is added with the immediate predecessor of this activity with the lowest activity number in the network.

The following three sections report computational results to evaluate the performance of the algorithm. All algorithms were coded in C++ and run on Intel Core™ i7-3610QM CPU with 2GB RAM. Each table in the following sections will display the values for the κ_1 and κ_2, and hence, the percentage of logical constraints can be calculated as specified earlier. In order to easily see this percentage, the rows with label percent Log display this percentage, and ranges between 0 percent (no logical constraints) and 100 percent (all OR or BI constraints). The stop criterion is calculated as the number of generated schedules and is set at 1000, 5000, 50,000 and 500,000 schedules.

Section 4.1.1 shows the results for the MRCPSP instances without the use of logical constraints and serves as a benchmark experiment to validate and compare our procedure with the best performing results available in literature. In Section 4.1.2, the single-mode RCPSP instances are solved under various settings of the percentage OR and BI constraints, and these results are compared with the results without logical constraints. Similarly, Section 4.1.3 displays the results for the MRCPSP instances and shows the impact of the network transformations on the size of the project network in case both multiple modes as well as logical constraints are used. In Section 4.1.4, the impact of increasing values for the changeover times used between two activities with BI constraints is discussed. Section 4.1.5 briefly discusses the impact of the random components of the solution approach on the stability of the test results.

In order to facilitate comparison for future researchers with the summary results displayed in the tables of the next sections, an MS Excel file has been put available at www.projectmanagement.ugent.be containing all the relevant information for each project instance. The project instances can be easily generated using the κ_1 and κ_2 parameters as discussed earlier. In the tables of the next sections, the following key parameters will be used to display the quality of the procedures:

- #Act: The number of activities in the project network after the network transformations of Section 3.1.
- CPU: Average runtime of the procedure until the stop criterion is met, expressed in seconds.
- #Opt: Number of optimal solutions found, expressed as a percentage of the total number of files.
- #Imp: Number of improvements found compared to the best known solutions reported on the PSPLIB and MMLIB websites reported in June 2014. Since these best known solutions can changeover time, they are also available in the MS Excel file that is available online. This metric is expressed as the number of instances, and not as a percentage.
- Percent Opt: Percentage deviation from the optimal solutions (only valid for the J10–J20 PSPLIB instances for which optimal solution are found).
- Percent LB: Percentage deviation from a known lower bound. The lower bounds used for the PSPLIB instances are described in detail on the PSPLIB website, while for the MMLIB instances, the lower bound is equal to the critical path. All bounds are also displayed in the MS Excel file discussed earlier.
- Percent UB: Percentage deviation from the best known solution (upper bound) as reported on the website of the PSPLIB and MMLIB instances. Similarly, these upper bounds are also given in the MS Excel file for each instance.

Note that not all multi-mode instances for the PSPLIB dataset are feasible, and therefore, the percentages calculated using the predefined metrics are always based on the files for which feasible solutions exist. The MMLIB instance set contains only feasible instances, as explained in Van Peteghem and Vanhoucke (2014).

4.1.1. Multi-mode results without logical constraints

Since no results are available for the single-mode and multi-mode version of the RCPSP with logical constraints, we have first tested our algorithm on data instances from literature on multi-mode instances without logical constraints. Only if these results can compete with results from literature, it can be concluded that the algorithm is a good choice for extension to logical constraints. To that purpose, Table 1 displays results for the multi-mode instances without logical constraints on the PSPLIB and MMLIB instances, and serves as a intermediate table to validate the performance of our algorithm in comparison with the best known procedures from literature. The table shows that the results are competitive with current state-of-the-art procedures to solve the MRCPSP as reported by Van Peteghem and Vanhoucke (2014) who present a list of all multi-mode procedures available in literature and benchmark each procedure under a stop criterion of 5000, 50,000 and 500,000 schedules using the percent Opt (for J10–J20 instances of the PSPLIB) or percent LB (for the PSPLIB J30 instances and all MMLIB instances).

Table 1
Comparative results for the multi-mode test PSPLIB and MMLIB instances without logical constraints.

Set	Subset	#Feasible	1000	5000	50,000	500,000	1000	5000	50,000	500,000
			Percent Opt or Percent LB				Percent Opt or Percent UB			
	J10	536	0.68	0.07	0	0	0.68	0.07	0	0
	J12	547	1.35	0.20	0.01	0	1.35	0.20	0.01	0
	J14	551	2.01	0.30	0.02	0.01	2.01	0.30	0.02	0.01
PSPLIB	J16	550	2.71	0.53	0.04	0.01	2.71	0.53	0.04	0.01
	J18	552	3.31	0.54	0.07	0.02	3.31	0.54	0.07	0.02
	J20	554	4.13	0.94	0.15	0.01	4.13	0.94	0.15	0.01
	J30	552	20.70	14.62	12.75	12.43	7.12	1.91	0.39	0.14
	50	540	38.47	29.42	24.63	23.32	12.28	4.82	1.04	0.02
MMLIB	100	540	46.80	34.60	26.13	23.99	18.94	8.83	2.07	0.39
	+	3240	146.69	120.79	96.97	n.a.	31.28	16.57	3.87	n.a.
			#Opt				CPU			
	J10	536	90.49	98.69	100	100	0.04	0.2	1.98	25.96
	J12	547	80.26	96.34	99.82	100	0.05	0.26	2.57	34.54
	J14	551	66.79	92.92	99.27	99.64	0.07	0.35	3.49	38.63
PSPLIB	J16	550	56.73	87.82	98.91	99.82	0.11	0.55	5.38	43.84
	J18	552	50.00	86.78	97.83	99.46	0.14	0.71	7.01	59.82
	J20	554	42.24	78.88	95.67	99.46	0.17	0.83	8.23	59.49
	J30	552	26.45	49.46	54.53	55.43	0.33	1.26	12.99	138.1
	50	540	7.04	21.85	34.26	37.96	1.14	5.62	53.89	482.7
MMLIB	100	540	1.85	8.89	31.48	40.37	4.18	20.49	171.1	1,718.4
	+	3240	0.43	2.31	6.79	n.a.	7.25	31.69	334.8	n.a.

PSPLIB. In Van Peteghem and Vanhoucke (2014), it has been shown that the best performing procedure in literature tested on the PSPLIB instances displays an average percent LB equal to 0, 0.32, 13.66 for the J10, J20 and J30 instances respectively for 5000 schedules. The results in Table 1 are slightly worse and therefore on the fourth place on the list of procedures. Increasing the stop criterion to 50,000 schedules obviously leads to improvements but a comparison is no longer possible due to the lack of results available for the PSPLIB instances. Our procedure however shows that the percentage improvement increases more rapidly for increasing levels of the stop criterion compared to the procedures in literature. This has also been observed in the paper written by Coelho and Vanhoucke (2011).

MMLIB. For the MMLIB instances, the best values for percent LB are reported to be equal to 23.79 percent, 24.02 percent and 92.76 percent for the 50,000 schedule limits for the MMLIB50, MMLIB100 and MMLIB+ instances respectively (Van Peteghem & Vanhoucke, 2014). The results in Table 1 are slightly worse but show that our procedure is now able to obtain the second best results in literature. Moreover, new solutions have been found for the MMLIB instances compared to the ones reported in Van Peteghem and Vanhoucke (2014), but these numbers are not shown in the table. More precisely, 6 and 41 improved solutions have been found for the MMLIB50 dataset under a stop criterion of 50,000 and 500,000 schedules, 15 and 76 for the MMLIB100 dataset and 295 for the MMLIB+ dataset (only for the 50,000 schedules stop criterion). These improved solutions have been reported on the multi-mode website where the best known solutions without logical constraints are available.

A similar reasoning can be made for the other criteria percent UB, #Opt and CPU and show that out procedure is not always able to outperform the best performing algorithm in literature, but can often compete with it or can, as previously mentioned, sometimes find improved solutions. Therefore, we conclude that the procedure presented in this paper is competitive with the multi-mode procedures in literature and although it cannot be considered as the best performing algorithm to solve the MRCPSP in literature, it is certainly a good alternative for extending it with logical constraints for which currently no results exist.

4.1.2. Single-mode results: impact of OR and BI relations

Table 2 displays results for the single-mode instances of the PSPLIB sets J30, J60, J90 and J120 instances. The results with both the OR and BI constraints must be compared with the reported results without logical constraints (column 3 with $\kappa_1 = \kappa_2 = -$ and percent Log = 0 percent). All tests have been performed with changeover times equal to zero. For each column the positive values for κ_1, κ_2 (and the resulting value for percent Log) and the number of activities (#Act) are reported as a result of the network transformations discussed in Section 3.1. The results show that even for J120 instances, the maximum number of activities increases to only 256 and 330 for the OR and BI constraints, respectively (under percent Log = 100 percent) which illustrates that the transformations can be used even for large project instances.

The rows with labels percent Opt (J30 instances) or percent LB and percent UB (for all other instances) show the makespan decrease when introducing logical constraints. The results show that percentages up to 20 percent and more are not an exception when introducing BI or OR constraints, however, the improvements seem to decrease for larger instances. The number of improvements shows a similar trend. The average runtime of our procedure is still acceptable, even for a stop criterion up to 50,000 schedules.

It is believed that Table 2 is the first table presented in literature for single-mode RCPSP instances with logical constraints and provides enough detailed results for comparison for future algorithms that might be presented in literature.

4.1.3. Multi-mode results: impact of OR and BI relations

Table 3 contains similar results as Table 2, but now for the J10–J20 PSPLIB instances summarized in one single run, as well of for the PSPLIB J30 instances. A similar layout is used in Table 4 to show the results for the three classes of MMLIB instances. The run on the relatively small J10–J20 PSPLIB multi-mode instances confirms the previously found results and shows that decreased makespans can be obtained by using logical constraints. However, the results also show that the transformed network instances are now much more complex, and in some cases, the procedure cannot find a solution of the same quality when compared with the instances without logical constraints for the same stop criterion. Only when the stop criterion is set to high values, significant improvements could be found. The main reason lies in the increase of the number of activities due to the network transformation, which sometimes leads to projects with up to 1622 activities for the MMLIB+ instances. This huge increase in the project network obviously leads to a much more complex search process of the algorithm, and hence, a decreased performance.

Table 2
Results of logical constraints for the single-mode PSPLIB test instances.

				OR				BI			
		κ_1	–	1	1	1	1	1	1	1	1
		κ_2	–	10	5	2	1	10	5	2	1
		Percent Log	0	10	20	50	100	10	20	50	100
J30		Max.	32	38	42	54	67	44	52	76	88
	#Act	Avg.	32	35.1	37.1	45	57	39.8	43.9	61.6	76
		Min.	32	32	33	36	47	32	32	44	64
		1000	0.17	−2.67	−4.22	−10.72	−17.74	−1.89	−3.43	−10.00	−13.84
	Percent Opt	5000	0.08	−2.97	−4.57	−11.55	−19.40	−2.04	−3.59	−10.39	−14.53
		50,000	0.02	−3.11	−4.85	−12.37	−20.85	−2.14	−3.71	−10.52	−14.72
		1000	0.02	0.02	0.02	0.03	0.05	0.03	0.03	0.06	0.09
	CPU	5000	0.10	0.11	0.12	0.17	0.27	0.14	0.16	0.32	0.48
		50,000	0.98	1.01	1.10	1.57	2.40	1.30	1.54	3.01	4.50
		1000	0	200	284	430	463	165	247	399	441
	#Imp	5000	0	218	310	443	473	177	256	412	452
		50,000	0	233	320	458	475	184	270	415	455
J60		Max.	62	72	80	101	130	86	102	142	170
	#Act	Avg.	62	67.4	71.9	87	111	76.9	87.6	121.8	152.7
		Min.	62	62	63	71	92	66	70	98	134
		1000	1.43	−1.23	−2.85	−9.25	−15.62	−1.80	−4.53	−12.81	−18.06
	Percent LB	5000	1.05	−1.74	−3.53	−10.51	−17.57	−2.25	−5.04	−13.67	−19.43
		50,000	0.84	−2.11	−3.98	−11.55	−19.27	−2.51	−5.31	−14.18	−20.21
		1000	0.78	−1.88	−3.50	−9.90	−16.27	−2.45	−5.17	−13.45	−18.70
	Percent UB	5000	0.41	−2.38	−4.16	−11.14	18.20	−2.88	−5.67	−14.29	−20.05
		50,000	0.21	−2.74	−4.61	−12.18	−19.89	−3.13	−5.93	−14.79	−20.81
		1000	0.03	0.03	0.04	0.06	0.10	0.05	0.06	0.14	0.25
	CPU	5000	0.15	0.18	0.21	0.33	0.57	0.24	0.33	0.74	1.27
		50,000	1.16	1.39	1.60	2.49	4.32	1.97	2.71	6.18	10.65
		1000	0	208	286	387	415	193	290	418	438
	#Imp	5000	0	228	304	402	439	209	309	438	458
		50,000	0	250	331	429	459	232	333	458	473
J90		Max.	92	106	117	150	193	128	152	208	252
	#Act	Avg.	92	100	106.7	129.2	165	114.8	132.2	181.4	230.5
		Min.	92	94	96	109	137	100	112	156	208
		1000	2.24	−0.85	−2.65	−8.91	−15.37	−1.97	−5.46	−14.00	−20.05
	Percent LB	5000	1.75	−1.48	−3.37	−10.28	−17.27	−2.55	−6.06	−14.83	−21.55
		50,000	1.41	−1.97	−3.99	−11.52	−19.21	−2.93	−6.49	−15.50	−22.74
		1000	1.09	−2.00	−3.79	−10.05	−16.51	−3.11	−6.59	−15.11	−21.16
	Percent UB	5000	0.63	−2.60	−4.49	11.40	−18.39	−3.66	−7.16	−15.91	−22.64
		50,000	0.31	−3.07	−5.09	−12.62	−20.30	−4.02	−7.57	−16.55	−23.79
		1000	0.04	0.05	0.06	0.1	0.19	0.07	0.10	0.25	0.47
	CPU	5000	0.20	0.25	0.29	0.49	0.95	0.36	0.54	1.3	2.4
		50,000	1.93	2.41	2.90	4.93	9.51	3.31	5.00	12.09	22.60
		1,000	0	225	295	380	395	221	304	414	425
	#Imp	5,000	0	238	310	394	404	237	328	433	439
		50,000	0	252	331	421	434	262	371	462	476
J120		Max.	122	140	158	196	256	166	194	270	330
	#Act	Avg.	122	132.2	141.4	171	219	151.6	175	240	307.4
		Min.	122	124	129	142	182	130	142	210	282
		1,000	6.44	4.88	4.18	2.10	−0.22	3.29	1.22	−3.56	−6.18
	Percent LB	5,000	5.18	3.50	2.74	0.52	−1.79	1.89	−0.26	−5.30	−8.13
		50,000	4.09	2.25	1.29	−1.35	−3.72	0.66	−1.66	−7.27	−10.43
		1,000	3.26	1.71	1.01	−1.05	−3.37	0.15	−1.89	−6.62	−9.23
	Percent UB	5,000	2.06	0.40	−0.36	−2.57	−4.89	−1.18	−3.30	−8.30	−11.11
		50,000	1.04	−0.79	−1.74	−4.37	−6.73	−2.34	−4.62	−10.17	−13.31
		1,000	0.05	0.06	0.08	0.14	0.28	0.08	0.13	0.34	0.68
	CPU	5,000	0.26	0.32	0.39	0.69	1.39	0.50	0.78	2.00	3.98
		50,000	2.45	3.09	3.80	6.72	13.82	3.94	6.23	16.14	32.31
		1,000	0	136	180	268	316	189	284	392	419
	#Imp	5,000	0	167	216	309	341	242	357	447	468
		50,000	0	209	267	373	406	334	456	551	557

Nevertheless, it is believed that this multi-mode table can still be used as best known results for the MRCPSP with logical constraints and serves as a comparison table for future research.

4.1.4. Changeover times

In the previous experiments, the values of the changeover times between activities are set equal to zero, i.e. $\gamma_{ij} = \gamma_{ji} = 0$. However, the previously mentioned MS Excel file that can be downloaded from www.projectmanagement.ugent.be also contains solutions for positive values for the changeover times and the results will be briefly discussed in this section. More precisely, the changeover times have been increased from 0 to 1, 2, 4 and 8, no OR constraints have been added to the tests and the percentage of BI constraints (percent Log) is varied from 10 percent to 20 percent, 50 percent and 100 percent. Tests have been carried out on the single-mode J30–J120 PSPLIB instances and the multi-mode J30 PSPLIB instances with a stop criterion of 50,000 schedules. Each time, the percentage reduction in the makespan is reported as the previously mentioned percent Opt (for the J30 single-mode instances) or percent Opt (for all other instances) and these values are compared with the results with only

Table 3
Results of logical constraints for the multi-mode PSPLIB test instances.

				OR				BI			
		κ_1	–	1	1	1	1	1	1	1	1
		κ_2	–	10	5	2	1	10	5	2	1
		Percent Log	0	10	20	50	100	10	20	50	100
J10–J20		Max.	62	74	80	98	113	78	86	126	134
	#Act	Avg.	47	52	54	68	84	53	56	75	86
		Min.	32	32	32	35	47	32	32	40	48
		1000	2.38	0.51	−0.40	−5.17	−9.99	1.31	0.84	−1.30	−2.33
	Percent LB	5000	0.43	−1.93	−3.26	−9.60	−15.69	−0.91	−1.54	−5.09	−6.64
		50,000	0.05	−2.56	−4.12	−11.77	−19.75	−1.34	−2.01	−5.97	−7.82
		1000	2.38	0.51	−0.40	−5.17	−9.99	1.31	0.84	−1.30	−2.33
	Percent UB	5000	0.43	−1.93	−3.26	−9.60	−15.69	−0.91	−1.54	−5.09	−6.64
		50,000	0.05	−2.56	−4.12	−11.77	−19.75	−1.34	−2.01	−5.97	−7.82
		1000	0.10	0.1	0.11	0.12	0.13	0.11	0.11	0.15	0.17
	CPU	5000	0.51	0.55	0.57	0.63	0.69	0.59	0.62	0.92	1.06
		50,000	4.88	5.11	5.25	5.83	6.46	5.90	6.50	11.92	15.28
		1000	0	516	765	1,854	2,457	357	483	1,195	1,396
	#Imp	5000	0	866	1,279	2,539	3,055	507	702	1,674	1,965
		50,000	0	1,024	1,493	2,787	3,191	551	761	1,865	2,163
J30		Max.	92	110	119	143	167	116	132	180	204
	#Act	Avg.	92	101	107	131	167	107	116	150	179
		Min.	92	95	98	119	167	92	92	116	156
		1000	20.70	18.57	17.86	14.03	10.26	18.42	17.70	13.27	11.39
	Percent LB	5000	14.62	12.11	10.66	5.44	1.33	11.95	10.44	4.57	1.19
		50,000	12.75	9.58	7.87	0.60	−4.98	9.87	8.27	0.97	−3.26
		1000	7.12	5.16	4.47	0.82	−2.64	5.04	4.31	0.26	1.42
	Percent UB	5000	1.91	−0.47	−1.80	−6.79	−10.69	−0.56	−1.94	−7.46	−10.56
		50,000	0.39	−2.54	−4.18	−11.08	−16.38	−2.26	−3.75	−10.52	−14.40
		1000	0.22	0.23	0.24	0.26	0.29	0.25	0.27	0.34	0.39
	CPU	5000	1.09	1.13	1.16	1.3	1.44	1.26	1.35	1.76	2.07
		50,000	10.91	11.34	11.78	13.02	14.43	13.51	15.01	22.77	28.68
		1000	0	69	98	206	286	72	102	217	274
	#Imp	5000	0	139	201	384	428	133	184	366	416
		50,000	0	211	297	472	507	178	247	438	496

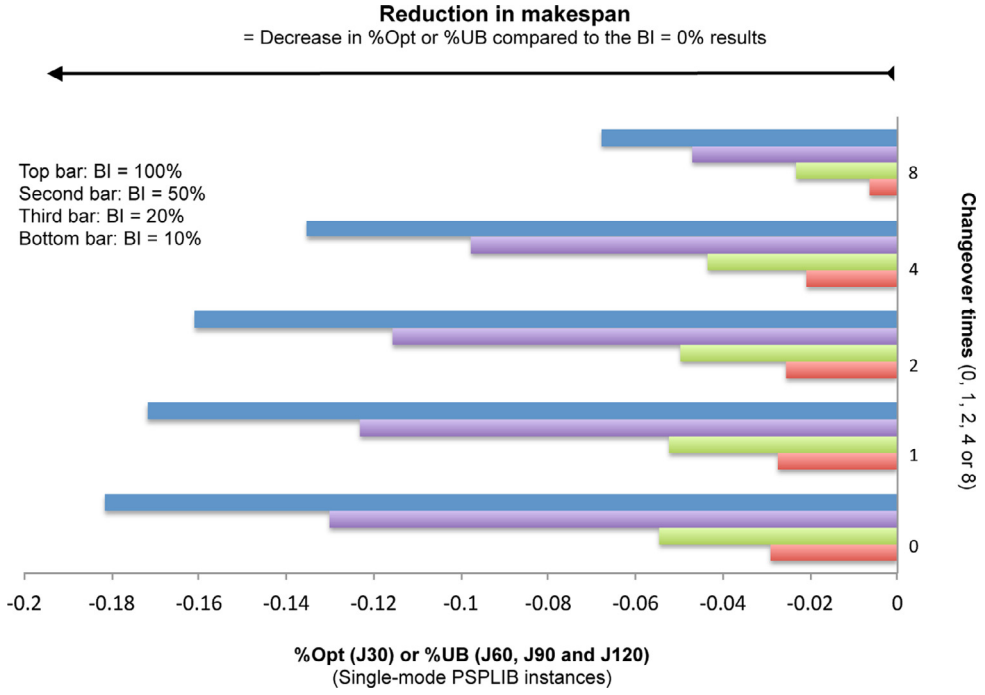


Fig. 7. The impact of increasing changeover times on the single-mode PSPLIB J30–J120 instances.

AND constraints (no OR and no BI constraints). Results are shown in Figs. 7 and 8.

Fig. 7 displays the results for the single-mode PSPLIB instances J30–J120, and clearly shows that all tests have resulted in a makespan decrease compared to the case where no OR and BI constraints are used. Obviously, the higher the percentage of BI constraints, the more significant the makespan reduction. The graphs also show that increasing values for the changeover times lead to a lower percentage of reductions in the project makespan, but even with changeover times up to 8 and a percentage of BI constraints of only 10 percent, there are still some small improvements compared to the base case.

Table 4
Results of logical constraints for the multi-mode MMLIB test instances.

			OR				BI				
		κ_1	–	1	1	1	1	1	1	1	1
		κ_2	–	10	5	2	1	10	5	2	1
		Percent Log	0	10	20	50	100	10	20	50	100
50		Max.	152	242	293	479	782	192	216	264	304
	#Act	Avg.	152	200	240	375	593	167	180	215	261
		Min.	152	167	194	290	437	152	152	176	200
		1000	38.47	37.70	36.51	33.61	27.05	39.50	40.46	45.29	55.61
	Percent LB	5000	29.42	28.54	27.24	24.13	18.88	29.97	30.34	32.38	38.39
		50,000	24.63	22.71	21.04	16.44	11.00	24.51	24.27	24.18	25.57
		1000	12.28	11.61	10.58	8.15	2.42	13.26	14.00	17.76	26.17
	Percent UB	5000	4.82	3.99	2.89	0.19	−4.42	5.32	5.58	7.11	11.99
		50,000	1.04	−0.66	−2.13	−6.18	−11.04	0.93	0.72	0.51	1.60
		1000	0.91	1.15	1.36	2.17	3.81	1.23	1.3	1.51	1.81
	CPU	5000	5.44	6.89	8.16	12.89	22.98	5.94	6.32	7.33	8.73
		50,000	48.67	60.95	71.15	110.01	187.92	55.87	60.76	73.49	89.26
		1000	0	13	24	81	193	2	7	5	5
	#Imp	5000	0	49	86	196	314	14	29	42	19
		50,000	6	132	220	362	434	42	67	102	97
100		Max.	302	560	761	1391	2351	366	414	486	590
	#Act	Avg.	302	472	629	1,123	1929	330	356	425	517
		Min.	302	389	482	836	1358	302	310	350	438
		1000	46.80	46.10	45.05	42.70	35.01	48.60	51.75	65.22	101.08
	Percent LB	5000	34.60	34.49	34.04	32.09	27.20	36.50	37.98	47.81	75.52
		50,000	26.13	25.21	24.43	21.94	18.59	26.56	26.90	30.90	47.79
		1000	18.94	18.29	17.35	15.14	8.14	20.48	23.20	34.57	65.83
	Percent UB	5000	8.83	8.69	8.29	6.42	1.87	10.37	11.65	19.92	43.68
		50,000	2.07	1.17	0.39	−1.95	−5.20	2.37	2.61	5.82	20.01
		1000	2.97	4.56	6.27	13.27	31.09	4.6	4.86	5.68	6.81
	CPU	5000	17.89	27.46	37.43	78.18	185.16	21.17	22.43	25.87	30.53
		50,000	158.64	238.92	321.84	651.19	1478.84	183.46	195.18	226.22	270.97
		1000	0	2	4	21	145	1	1	1	0
	#Imp	5000	0	15	34	79	202	4	7	4	1
		50,000	15	93	142	238	304	31	47	37	5
*		Max.	902	1703	2279	4169	7049	1062	1182	1362	1622
	#Act	Avg.	452	668	866	1493	2518	490	524	616	736
		Min.	152	167	191	281	410	152	152	176	200
		1000	146.69	145.70	145.21	141.92	135.19	150.04	155.69	181.17	232.72
	Percent LB	5000	120.79	120.20	119.39	118.80	118.38	123.15	127.79	147.45	187.84
		50,000	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
		1000	31.28	30.58	30.24	27.96	23.78	33.25	36.54	51.03	80.89
	Percent UB	5000	16.57	16.20	15.63	14.99	14.33	17.96	20.46	31.42	54.40
		50,000	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
		1000	7.06	10.04	12.82	21.97	38.22	7.97	8.71	11.15	15.21
	CPU	5000	27.25	37.95	47.79	80.18	137.2	28.54	30.65	36.43	45.14
		50,000	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
		1000	0	10	16	46	192	1	2	2	3
	#Imp	5000	0	41	73	174	369	6	15	17	11
		50,000	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.

The results for the multi-mode PSPLIB instances as shown in Fig. 8 show similar results, but the improvements are less outspoken. Increasing values of the changeover times result in makespan improvements for changeover time values up to 2. When the changeover times increase even further to 4 or 8, the procedure can no longer report improvements, and on the contrary lead to an increase in the makespan compared to the base case with only AND constraints. Obviously, the reason is that the size of the network has grown significantly, both due to the BI constraints and the multiple modes in the activities, and hence the procedure is no longer able to fully optimize the search towards improved solutions with high values for the changeover times.

4.1.5. Stability of results

Since the meta-heuristic approach proposed in this paper is not deterministic, the solutions obtained and reported in the previous sections should be tested against the stability of the procedure. To that purpose, this section reports on results for a limited computational experiment used with random replications. Obviously, using replications on each test instance would increase the computational time prohibitively for the large numerical study of the previous sections, and therefore, the following choices have been made. A new computational experiment has been set up to test the stability of the procedure by selecting the first instance every ten instances. The reason of this choice lies in the fact that the PSPLIB library contains three parameters (coefficient of network complexity CNC, resource factor RF and resource strength RS) that vary every 10 instances. This choice results in a selection of 48 projects for the J30, J60 and J90 instances and 60 projects for the J120 instances. In doing so, it is assured that exactly one instance for each combination of the parameter values is used for the PSPLIB instance generations. Furthermore, the experiments also have been restricted to the single-mode instances with 50 percent OR and 50 percent BI values for the logical constraints using a stop criterion of 50,000 schedules. Each instance is solved 10 times (10 replications) and the results are reported in Table 5. The columns with label "St.Dev." report the standard deviation of the makespans while the columns labelled with "# St.Dev. = 0" report the number of instances with a zero standard deviation. The row with label "Avg." reports the average standard deviation for all test instances.

The table shows that the average standard deviations are always lower than one time unit, with a lot of instances with a zero standard deviation for all 10 replications. The average standard deviations of

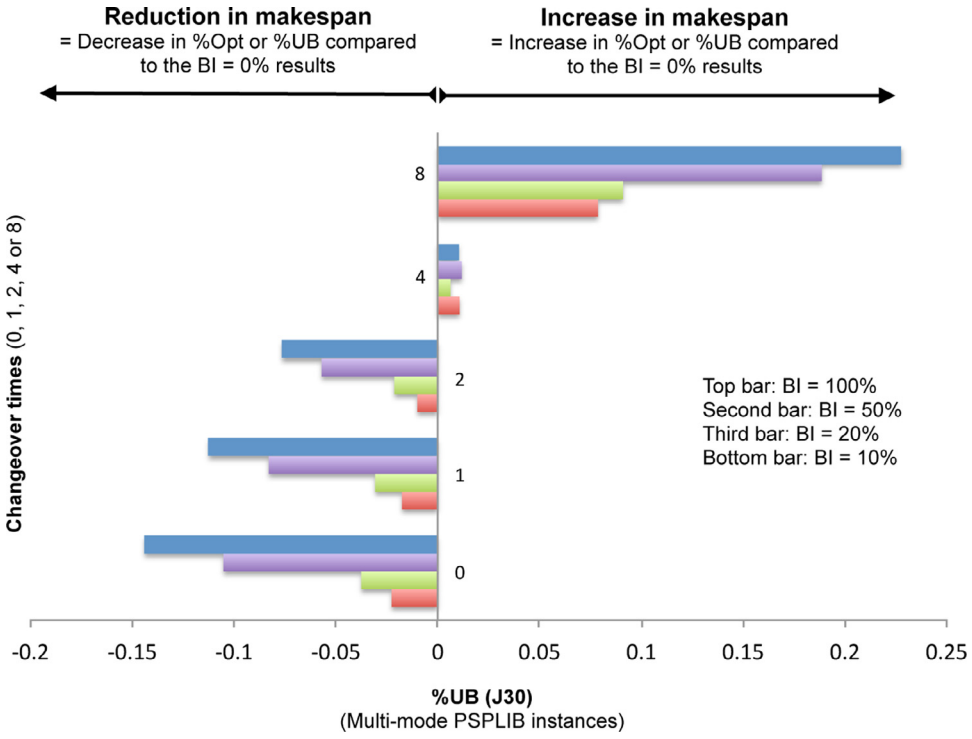


Fig. 8. The impact of increasing changeover times on the multi-mode PSPLIB J30 instances.

Table 5
Stability of the results using a sample of the test experiments.

Set	St.Dev		# St.Dev. = 0	
	OR 50 percent	BI 50 percent	OR 50 percent	BI 50 percent
J30	0.20	0.03	60	92
J60	0.39	0.18	40	71
J90	0.52	0.21	35	65
J120	0.68	0.52	8	25
Avg.	0.46	0.25		

0.46 and 0.25 for the OR 50 percent and BI 50 percent experiments, respectively, illustrate that the makespans vary with only 1.84 time units and 1 time unit with a probability of 95 percent for the 10 replications (based on the calculations of a two standard deviation interval). The worst results have been found for the J120 instances with ID = 46 (for OR 50 percent) and ID = 31 (for BI 50 percent) with a standard deviation of 1.85 and 1.70, respectively (not shown in the table). Since there is currently no other approach available in literature that solves similar problems, a comparison with other procedures cannot be made. However, the selection of instances for the stability results reported in this section can be easily replicated, allowing to benchmark new results both on quality (previous sections) and stability (this section).

5. Conclusions

In this paper, the single-mode and multi-mode resource-constrained project scheduling problems from literature are extended with two types of logical constraints. The first type considers the use of OR relations between activities and specifies that only one of the predecessors must be finished before an activity can start. The second type is referred to as the bi-directional (BI) constraints and specifies that two activities cannot be scheduled in parallel, but in-between these activities a changeover time might possibly be incurred. These two new logical constraints extend the classic RCPSP problem and it is solved by a new procedure discussed in this paper.

The new procedure makes use of a heuristic RCPSP solver and a SAT solver and relies on network transformations that extend the project network and transforms the OR and BI constraints into traditional AND constraints. In doing so, the project can be solved by any (meta)heuristic project scheduling algorithm without explicitly taking these logical constraints into account. The SAT solver guarantees that logical clauses between activities implicitly take the OR and BI constraints into account.

The procedure is tested on two well-known datasets from literature, the PSPLIB (single-mode and multi-mode) and the MMLIB (multi-mode). A first test experiment on MRCPSP instances without logical constraints shows that the procedure is able to compete with the best procedures from literature. A second test on single-mode instances with logical constraints shows promising results as the size of the transformed networks does not increase too much, and promising results that can be used for future research comparison purposes have been generated within a reasonable time. The third experiment has used the MRCPSP instances with logical constraints and a similar table for future comparison purposes has been displayed. These results show that the size and complexity of the networks increase rapidly, and it was therefore not always possible to find improved solutions (compared to the solutions of these instances when the logical constraints are ignored). Future research should therefore focus on the development of dedicated algorithms for these multi-mode instances to find improved solutions in a reasonable time.

Acknowledgement

We acknowledge the support given by the "Bijzonder Onderzoeksfonds" (BOF) for the project under contract number BOF12GOA021.

Appendix A

In this appendix, it will be shown how the relation between the "activity scheduling" and "activity selection" steps of the algorithm of Fig. 6 select the activities from the example network of Fig. 5 and schedule them to obtain the optimal solution presented in Fig. 1.

A.1. Activity selection

In order to select a subset of activities from the extended project network, the following clauses need to be generated:

No constraints: For the activities Start, A, B, C, G, H and End with no OR or BI constraints, a special k clause (with $k = 7$) must be generated equal to $z_{\text{Start}} + z_{\text{A}} + z_{\text{B}} + z_{\text{C}} + z_{\text{G}} + z_{\text{H}} + z_{\text{End}} \geq 7$ to assure that each of these 7 activities is selected (i.e. set to true) in the "activity selection" step.

OR constraints: In order to model the OR constraints for activities D_1, D_2 and D_3, only two clauses must be generated, as explained in Section 3.2. A normal clause $z_{D_1} + z_{D_2} + z_{D_3} \geq 1$ is necessary to select at least one of the three activities, while a special k clause $\overline{z_{D_1}} + \overline{z_{D_2}} + \overline{z_{D_3}} \geq 2$ assures that at most one of these activities will be selected.

BI constraints: The bidirectional constraints for the activities E_1, E_2, F_1 and F_2 including the changeover time activities EF and FE can be modelled using the following set of clauses:

$$z_{E_1} + z_{F_2} = 1 \text{ or } ((z_{E_1} + z_{F_2}) \text{ and } (\overline{z_{E_1}} + \overline{z_{F_2}}))$$
$$z_{E_1} - z_{F_1} = 0 \text{ or } ((z_{E_1} + \overline{z_{F_1}}) \text{ and } (\overline{z_{E_1}} + z_{F_1}))$$
$$z_{E_1} - z_{EF} = 0 \text{ or } ((z_{E_1} + \overline{z_{EF}}) \text{ and } (\overline{z_{E_1}} + z_{EF}))$$
$$z_{E_2} - z_{F_2} = 0 \text{ or } ((z_{E_2} + \overline{z_{F_2}}) \text{ and } (\overline{z_{E_2}} + z_{F_2}))$$
$$z_{FE} - z_{F_2} = 0 \text{ or } ((z_{FE} + \overline{z_{F_2}}) \text{ and } (\overline{z_{FE}} + z_{F_2}))$$

Assume now that an example extended activity list will be used equal to {Start, A, D_1, B, D_2, C, D_3, E_1, F_2, EF, FE, F_1, E_2, G, H, End}, then the DPLL runs as described along the following lines. The set T is used to refer to the set of assigned literals at a given moment (initially, $T = \{\}$) with z_x a variable x set to true and $\overline{z_x}$ a variable x set to false.

Step 1. The unit clause rule for the activities with no OR or BI constraints sets the literals for these 7 activities to true, and hence, $T = \{z_{\text{Start}}, z_{\text{A}}, z_{\text{B}}, z_{\text{C}}, z_{\text{G}}, z_{\text{H}}, z_{\text{End}}\}$. Now that this clause has been satisfied, the search continues with the OR and BI clauses.

Step 2. The selection of the first non-assigned variable from the extended activity list results in activity D_1. This variable is set to true, and hence, $T = \{z_{\text{Start}}, z_{\text{A}}, z_{\text{B}}, z_{\text{C}}, z_{\text{G}}, z_{\text{H}}, z_{\text{End}}, z_{D_1}\}$. Since the OR clauses force that only one of the three variables can be set to true, the remaining variables for activities D_2 and D_3 will be set to false, resulting in $T = \{z_{\text{Start}}, z_{\text{A}}, z_{\text{B}}, z_{\text{C}}, z_{\text{G}}, z_{\text{H}}, z_{\text{End}}, z_{D_1}, \overline{z_{D_2}}, \overline{z_{D_3}}\}$. At this phase of the search, all OR clauses are satisfied, and in the next step, the search on the BI clauses continues on the remaining literals that are not assigned yet.

Step 3. The selection of the first non-assigned variable from the extended activity list results in activity E_1 and hence, the variable z_{E_1} is set to true. Setting this variable to true automatically sets the variable $\overline{z_{F_2}}$, z_{F_1} and z_{EF} to satisfy the first three clauses of the BI constraints. Moreover, the value for $\overline{z_{F_2}}$ also implies that the variables $\overline{z_{E_2}}$ and $\overline{z_{FE}}$ are set to satisfy the last two clauses of the BI constraints. As a results, the set T is equal to $\{z_{\text{Start}}, z_{\text{A}}, z_{\text{B}}, z_{\text{C}}, z_{\text{G}}, z_{\text{H}}, z_{\text{End}}, z_{D_1}, \overline{z_{D_2}}, \overline{z_{D_3}}, z_{E_1}, \overline{z_{F_2}}, z_{F_1}, z_{EF}, \overline{z_{E_2}}, \overline{z_{FE}}\}$ and all clauses are satisfied.

A.2. Activity scheduling

A restricted activity list is returned from the previous steps containing only these activities with a z_x value set to true. This list is equal to {Start, A, B, C, G, H, End, D_1, E_1, F_1, EF}. This activity list will be scheduled according to a serial schedule generation scheme, resulting in the optimal solution displayed at the bottom schedule of Fig. 1. The metaheuristic search procedure will however continue its search on this restricted activity list to find other solutions, as is normally the case for any meta-heuristic to solve a resource-constrained project scheduling problem with only AND constraints.

References

Adelson-Velsky, G. M., & Levner, E. (2002). Project scheduling in and-or graphs: A generalization of Dijkstra's algorithm. *Mathematics of Operations Research, 27*, 504–517.

Belhe, U., & Kusiak, A. (1995). Resource constrained scheduling of hierarchically structured design activity networks. *IEEE Transactions on Engineering Management, 42*(2), 150–158.

Berthaut, F. (2011). *Optimal resource-constraint project scheduling with overlapping modes. Document de travail*. Faculté des Sciences de l'administration, Université Laval.

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

Coelho, J., & Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research, 213*, 73–82.

Coelho, J., & Vanhoucke, M. (2014). The resource-constrained multi-mode project scheduling problem. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on project management and scheduling*. Springer International Publishing AG.

Cook, S. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on theory of computing* (pp. 151–158).

Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem proving. *Communications of ACM, 5*(7), 394–397.

Debels, D., & Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project scheduling problems. *Operations Research, 55*, 457–469.

Demeulemeester, E., & Herroelen, W. (2002). *Project scheduling: A research handbook*. Kluwer Academic Publishers.

Drexl, A., Nissen, R., Patterson, J., & Salewski, F. (2000). ProGen/πx—An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research, 125*, 59–72.

Gillies, D., & Liu, J. W. (1995). Scheduling tasks with and/or precedence constraints. *SIAM Journal on Computing, 24*, 797–810.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research, 207*, 1–15.

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research, 25*, 279–302.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (1999). A classification scheme for project scheduling problems. In J. Weglarz (Ed.), *Project scheduling—Recent models, algorithms and applications* (pp. 1–26). Dordrecht: Kluwer Academic Publishers.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (2001). A note on the paper "resource-constrained project scheduling: Notation, classification, models and methods" by Brucker et al. *European Journal of Operational Research, 128*, 679–688.

Icmeli, O., Erengüç, S., & Zappe, C. (1993). Project scheduling problems: A survey. *International Journal of Operations & Production Management, 13*, 80–91.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*, 23–37.

Kolisch, R., & Sprecher, A. (1996). PSPLIB—a project scheduling problem library. *European Journal of Operational Research, 96*, 205–216.

Kuster, J., & Jannach, D. (2006). Handling airport ground processes based on resource-constrained project scheduling. In M. Ali, & R. Dapoigny (Eds.), *Advances in applied artificial intelligence: Vol. 4031*. In *Lecture notes in computer science*. Berlin/Heidelberg: Springer.

Li, K., & Willis, R. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research, 56*, 370–379.

Marques-Silva, J., & Sakallah, K. (1999). GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers, 48*, 506–521.

Möhring, R., Skutella, M., & Stork, F. (2004). Scheduling with and/or precedence constraints. *SIAM Journal on Computing, 33*, 393–415.

Özdamar, L., & Ulusoy, G. (1995). A survey on the resource-constrained project scheduling problem. *IIE Transactions, 27*, 574–586.

Pritsker, A., Watters, L., & Wolfe, P. (1969). Multi-project scheduling with limited resources: A zero-one programming approach. *Management Science, 16*, 93–108.

Selman, B., Levesque, H., & Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proceedings of the tenth national conference on artificial intelligence (AAAI'92)* (pp. 440–446). AAAI Press.

Valls, V., Ballestin, F., & Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research, 165*(2), 375–386.

Valls, V., Ballestin, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 185*(2), 495–508.

Van Peteghem, V., & Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research, 235*, 62–72.

Vanhoucke, M. (2012). *Project management with dynamic scheduling: Baseline scheduling, risk analysis and project control*: XVIII. Springer.