Constraints
Manuscript Draft

Manuscript Number: CONS213

Title: Optimal Methods for Resource Allocation and Scheduling - A Cross-Disciplinary Survey

Article Type: Survey Paper (Thomas Schiex)

Keywords: Scheduling; Resource Allocation; MRCPSP; Constraint Programming; Operations Research

Abstract: Resource Allocation and Scheduling problems consist into assigning over time resources from a candidate pool to a set of activities; activities are connected by precedence relations and their duration may depend on allocation decisions; side constraints may restrict possible resource assignments. Practical problems in this class arise in many fields, such as industrial scheduling and embedded system design.

An allocation and scheduling problem results from the composition of an assignment and a scheduling problem; the large variety of possible combinations justifies the absence of a single formal categorization of this problem class, whereas pure Assignment Problems (AP) and Resource Constrained Project Scheduling Problems (RCPSP) have been extensively studied by the research community.

Scheduling problems are well known to be among the toughest ones in combinatorial optimization: adding an assignment component results in a dramatical complexity increase. While up to 120 activities can be managed by modern approaches in the context of the pure RCPSP, instances with 20-30 activities prove to be very challenging if resource assignments are considered.

As a consequence, specific expertise on scheduling or resource assignment is a necessity to tackle their combination, yet it is far from being sufficient. In particular, the intrinsically hybrid nature of the problem provides motivation for the cross-contamination of techniques from different research fields.

This work addresses the lack of survey papers explicitly targeting mixed resource allocation and scheduling; we aim to provide a coherent overview of this class of problems as it is tackled in the Operations Research and Constraint Programming community; in particular, the focus is on exact methods and deterministic problems. Our discussion includes hybrid as well as pure OR/CP approaches.

Optimal Methods for Resource Allocation and Scheduling – A Cross-Disciplinary Survey

Michele Lombardi and Michela Milano

Submitted: December 2010

Abstract

Resource Allocation and Scheduling problems consist into assigning over time resources from a candidate pool to a set of activities; activities are connected by precedence relations and their duration may depend on allocation decisions; side constraints may restrict possible resource assignments. Practical problems in this class arise in many fields, such as industrial scheduling and embedded system design.

An allocation and scheduling problem results from the composition of an assignment and a scheduling problem; the large variety of possible combinations justifies the absence of a single formal categorization of this problem class, whereas pure Assignment Problems (AP) and Resource Constrained Project Scheduling Problems (RCPSP) have been extensively studied by the research community.

Scheduling problems are well known to be among the toughest ones in combinatorial optimization: adding an assignment component results in a dramatical complexity increase. While up to 120 activities can be managed by modern approaches in the context of the pure RCPSP, instances with 20-30 activities prove to be very challenging if resource assignments are considered.

As a consequence, specific expertise on scheduling or resource assignment is a necessity to tackle their combination, yet it is far from being sufficient. In particular, the intrinsically hybrid nature of the problem provides motivation for the cross-contamination of techniques from different research fields.

This work addresses the lack of survey papers explicitly targeting mixed resource allocation and scheduling; we aim to provide a coherent overview of this class of problems as it is tackled in the Operations Research and Constraint Programming community; in particular, the focus is on exact methods and deterministic problems. Our discussion includes hybrid as well as pure OR/CP approaches.

1 Resource Allocation and Scheduling

According to Baker (see [2]), a scheduling problem consists in "allocating scarce resources to activities over time". Many scheduling approaches have an emphasis on the *temporal* aspect of the problem; from this perspective, computing a schedule amounts to decide when to start each activity, while the resource requirements are assumed to be a priori known. The classical Resource Constrained Project Scheduling problem falls into this category, as well as Job Shop Scheduling and many variations thereof.

However, there is some practical evidence that in many application contexts resource allocation aspects cannot be disregarded. According to [83, 5, 13, 74], many real world problems feature *optional activities* which can be left non-executed (usually with an impact on costs) or *alternative recipes* to execute an activity; a recipe may specify different resource requirements or require the execution of different sets of sub-activities.

Here, we take into account a broad class of optimization problems where the *resource assignment* for each activity must be decided; the choice can be made out of a set of possible alternatives, or according to problem dependent constraints. The considered class contains some of the hardest combinatorial optimization problems ever: for example the NP-completeness of the classical Multi-mode Resource Constrained Project Scheduling Problem (MRCPSP, see [46]) has been proven in [74] and 30 activity instances from the PSPLIB [75] still set tough challenges to complete search methods [127]. Such a complexity is due to the addition of a resource assignment step over a notoriously hard problem (scheduling instances with 120 activities are already very hard for state-of-the-art techniques [109]).

In the Operations Research (OR) community, resource allocation and scheduling problems have been mainly considered under the flag of the MRCPSP and related trade-off problems (see [56, 117]). A fine grained classification has been proposed (see [51, 25]), resulting into a pretty crowded problem zoo. The thorough investigation of each problem class has lead to the identification of advanced optimization techniques.

On the opposite, the so called Constraint based Scheduling area [4] has an emphasis on the design of algorithmic *components* for different problem specificities; the Constraint Programming (CP) framework provides integration support, enabling one to model and solve complex real-world problems with limited effort. While CP techniques have proven very successful on pure scheduling problems, this is not the case when allocation decisions are taken into account, due to the search space blow-up and (usually) weaker propagation.

Specifically, both Integer Linear Programming (ILP) and CP techniques can claim individual successes with resource allocation and scheduling problems, but practical experience indicates that neither approach dominates the other in terms of computational performance. This raises interest in hybrid algorithmic techniques, to take advantage of the mutual strengths of heterogeneous methods and compensate for their weaknesses. Constraint Programming offers an ideal framework for the development of hybrid algorithms, due to the ability to customize the search strategy and encapsulate complex algorithmic techniques within global constraints; moreover, *the sharp separation between search and model* allows one to exploit filtering and propagation, whatever the search method is.

Due to the hardness of this problem class, the main body of the allocation and scheduling literature consists of heuristic appraoches; several methods have been employed, such as Large Neighborhood Search (e.g. [80]), decoupled search stages (e.g. [48, 123]), priority rule base scheduling (e.g. [29, 91]), local search (e.g. [74]) or metaheuristics (e.g. [120, 92, 106, 68, 94, 95]).

Therefore, a relatively small number of exact search strategies has been developed. Nevertheless, those approaches are of very high interest, for several reasons: (1) some advanced techniques actually allow the solution of practical size instances on specific domains (see [20, 30]); (2) exact search methods pro-

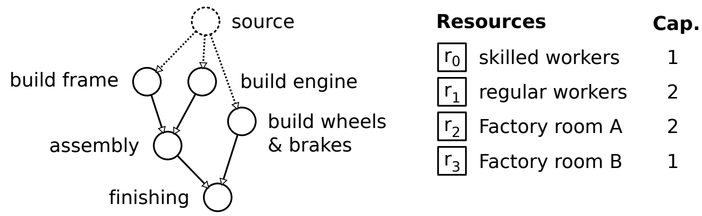

Figure 1: An example of project graph: building a motorbike

vide the backbone for very effective heuristic search methods (truncated search, restarts or local moves in Large Neighborhood Search [82, 50, 80]); finally (3) heuristics relying on exact methods offer better support for side constraints, frequently occurring in practical settings.

We provide a cross disciplinary survey on resource assignment and scheduling, from a constraint-based perspective; according to the equation:

`Constraint Programming = model + propagation + search`

we give an overview of state-of-art techniques and identify adopted *models*, *propagation* and bounding algorithms and *search* strategies. In particular, the focus is on *exact* methods and *deterministic* problems. In detail, Section 2 introduces the considered problem class and presets CP and OR models; Sections 3 and 4 are devoted to filtering algorithms and bounding rule, while am overview of search strategies is given in Section 5.

2 Modeling Techniques

2.1 Reference Problem Class

We consider a generic allocation and scheduling problem (reference problem) defined over a set of *activities* subject to temporal constraints; we adopt an Activity on Node (AoN) representation and describe the temporal constraints via a directed graph $G = \langle A, E \rangle$, where $A = \{a_0, a_1, \ldots\}$ is the set of nodes/activities and E is a set of *precedence relations*; following the MRCPSP literature, this will be referred to as *project graph*. Additional temporal constraints can become part of the problem as a result of *scheduling decisions*.

Activities require for their execution some resources r_k (see [46]) from a set R, with finite capacity c_k. The amount of r_k required by activity a_i depends on *allocation decisions*, i.e. on the set of resources an activity is actually assigned to; similarly, the duration of each activity is allocation dependent. Case specific side constraints restrict the possible resource assignment. A solution of the problem is a set of scheduling and allocation decisions which satisfy all problem constraint and optimize some performance measure.

The provided description does not correspond to any classical combinatorial optimization problem; rather, it provides a generic framework where many practical scheduling problems fit. Figure 1 shows a simple project graph for building a motorbike. The resource pool R includes workers and factory rooms; in principle, every resource assignment is possible: side constraints can be used to force each activity to require one worker and one room. It is customary to add 0-duration fake source and sink nodes to the graph, in case they are missing.

The next section describes the main modeling techniques developed for the reference problem; in a first stage we assume temporal constraints are simple end-to-start precedence relations and all resources are renewable (i.e. they provide c_k energy units per time unit, see [46]). Formally, the cumulative requirement of activities executing at an instant τ cannot exceed the capacity c_k, for each time instant $\tau \in [0, eoh]$, where eoh is the End of Horizon.

2.2 Constraint Based Models

Scheduling problems are classically modeled in CP by introducing for every activity a_i three integer variables [4], namely: (1) $\mathtt{S_i}$, representing the activity start time (i.e. the first time instant where the activity is executing); (2) $\mathtt{E_i}$ representing the activity end time (i.e. the first time instant where the activity is not executing); (3) $\mathtt{D_i}$ representing the activity duration. Start, end and duration variables must satisfy the constraint $\mathtt{E_i} = \mathtt{S_i} + \mathtt{D_i}$. Table 1 contains a quick reference for the notation used for variables throughout the paper.

Allocation decisions can be modeled by means of binary variables $\mathtt{X_{ik}}$ such that $\mathtt{X_{ik}} = \mathtt{1}$ if a_i requires/is-assigned-to r_k (the method is mentioned in [14]); duration variables are linked to allocation decisions by a constraint $\mathtt{D_i} = d_i(\mathtt{X})$, where $\mathtt{X}$ represents the whole set of $\mathtt{X_i}$ variables and the function $d_i(\mathtt{X})$ encodes the dependency on the resource assignment.

Bounds for the start and end variables domains are referred to by means of conventional names; namely $\min(\mathtt{S_i})$ is the Earliest Start Time – $EST(a_i)$ – and $\max(\mathtt{S_i})$ is the Latest Start Time – $LST(a_i)$; the Earliest End Time and Latest End Time – $EET(a_i)$ and $LET(a_i)$ – are defined analogously for the $\mathtt{E_i}$ variable. Precedence relations are modeled as linear constraints $\mathtt{E_i} \leq \mathtt{S_j}$; resource restrictions are enforced via the *cumulative* constraint (see [1]). Side constraints may restrict the possible resource assignments.

Figure 2 shows a basic CP model for cumulative, non-preemptive resource allocation and scheduling with no side constraint; the objective to be minimized is some function F of scheduling and allocation variables. The most common problem objective is the makespan, i.e. the highest completion time; formally $F(\mathtt{X}, \mathtt{S}, \mathtt{E}) = \max_{a_i \in A} \mathtt{E_i}$. The `cumulative` constraint allows durations and requirements to be specified as decision variables; in the example this is $\mathtt{RQ_{ik}}$, forced to equal the function $rq_{ik}(\mathtt{X})$ which encodes the dependency on the resource assignment.

Notation		Domain – *Meaning*	Context
temporal decisions	$\mathtt{S_i}$	$[0..eoh]$ – *start time of* a_i	CP, MRCPSP Disjunctive
	$\mathtt{E_i}$	$[0..eoh]$ – *end time of* a_i	CP, MRCPSP Disjunctive
	$\mathtt{D_i}$	$[0..\max(d_i(\mathtt{X}))]$ – *duration of* a_i	CP
	$\mathtt{T_i}$	$[0..eoh]$ – *time variable*	DTP_{FD}
	$\bar{\mathtt{A}}_\mathtt{i}$	$\perp \cup \{[s, e) \mid s \leq e\}$ – *interval variable*	Time Intervals
	$\mathtt{E_{i,h,\tau}}$	$\{0, 1\}$ – *1 if* a_i *ends at* τ *in mode* m_h	MRCPSP Time Indexed
resource assignments	$\mathtt{X_{ik}}$	$\{0, 1\}$ – *1 if* a_i *requires* r_k	CP
	$\mathtt{Y_i}$	$\mathbb{N}$ – *finite domain variable*	DTP_{FD}
	$\mathtt{EX_i}$	$\{0, 1\}$ – *execution variable*	Alternative activities
	$exec(\bar{\mathtt{A}}_\mathtt{i}$	$\{0, 1\}$ – *execution meta-constraint*	Time Intervals
	$\mathtt{M_{ih}}$	$\{0, 1\}$ – *1 if* a_i *has mode* m_h	MRCPSP Disjunctive

Table 1: Notation

$$
\begin{array}{rll}
\text{min:} & F(\mathtt{X},\mathtt{S},\mathtt{E}) & \\
\text{subject to:} & \mathtt{E_i} = \mathtt{S_i} + \mathtt{D_i} & \forall a_i \in A \\
& \mathtt{E_i} \leq \mathtt{S_j} & \forall (a_i, a_j) \in E \\
& \mathtt{cumulative}(\mathtt{S},\mathtt{D},\mathtt{RQ_{ik}}, c_k) & \forall r_k \in R \\
& \mathtt{D_i} = d_i(\mathtt{X}) & \forall a_i \in A \\
& \mathtt{RQ_{ik}} = rq_{ik}(\mathtt{X}) & \\
\text{with:} & \mathtt{S_i},\mathtt{E_i},\mathtt{D_i} \in \{0, \ldots eoh\} & \\
& \mathtt{X_{ik}} \in \{0, 1\} &
\end{array}
$$

Figure 2: A CP model for non-preemptive resource allocation and scheduling, with no side constraints

This approach is sufficient to *model* a wide range of problems; unfortunately, `cumulative` filtering can be very weak when requirement or duration variables are unbound, so that specific approaches to deal with resource allocation and scheduling are in practice needed.

Alternative Resources A first formalization allows an activity a_i to require one of a set $\bar{R}$ of *alternative resources* [65, 48, 100]; in this case an implicit constraint requires $\sum_{r_k \in \bar{R}} \mathtt{X_{ik}}$ to be exactly one for activities requiring $\bar{R}$. A fixed requirement (say $rq_{\bar{R}}$) is specified for the whole set, so that the corresponding $rq_{ik}(\mathtt{X})$ function is $rq_{\bar{R}} \cdot \mathtt{X_{ik}}$. Classical `cumulative` filtering algorithms [78] can be used once the resource selection has been performed. Propagation before selection is typically much weaker.

Alternative resources with unary capacity can be modeled within the framework of the *Disjunctive Temporal Problem with Finite Domain Constraints* (DTP_{FD}, see [97]); the basis for the approach is a network of time points (temporal variables) $\mathtt{T_i}$; each time point is coupled with a finite domain variable $\mathtt{Y_i}$. A time point can represent an activity start or end. Between each pair of time points $\mathtt{T_i}, \mathtt{T_j}$ one may post a *disjunction* of linear inequalities in the form $\mathtt{T_i} - \mathtt{T_j} \leq B(\mathtt{Y_i}, \mathtt{Y_j})$. Basically, the value of bounding function $B(\mathtt{Y_i}, \mathtt{Y_j})$ depends on the value of the finite domain variables. Variables $\mathtt{Y_i}$ can be linked to allocation decisions $\mathtt{X_{ik}}$ so that two activities may be required not to overlap if they are processed by the same unary resource.

Alternative Activities The so called *Alternative Activities* have been introduced by Beck in [13, 14] and can be used to model alternative resource assignments and their impact on durations. In such approach, each activity is assigned an *execution variable* $\mathtt{EX_i}$, with values in the discrete domain $\{0, 1\}$; namely, $\mathtt{EX_i} = 0$ if activity a_i does not execute, $\mathtt{EX_i} = 1$ in case a_i executes and $\mathtt{EX_i} = \{0, 1\}$ as long as it is undecided.

The project graph is extended by including *XOR nodes*, marking the start (and the end) of nested alternative blocks; formally, let x be a XOR node and $Succ(x)$ denote the set of successor activities, then $\sum_{a_i \in E^+(x)} \mathtt{EX_i} = 1$.

Each of the alternatives can represent a different recipe to execute the same logical activity (i.e. a duration value and the required resources), so that the XOR node corresponds to an allocation decision. As a consequence, variables $\mathtt{X_{ik}}$ can be dismissed or chained to $\mathtt{EX_i}$. Observe that XOR nodes allow one to model alternative *plans*, besides alternative resource assignments.

Alternative activities and plans are considered in *Temporal Networks with Alternatives* (TNA), introduced by Barták in [5, 7]. In the approach by Beck each XOR node has a corresponding join node, so that the network is always *nested*; in [7], conversely, general network structures are considered, in principle allowing one to express more complex assignment constraints. Unfortunately, the work proves that completing a partial assignment of execution variables (e.g. completing a partial resource assignment) is NP-complete, unless the network is nested [6, 8]; some tractable substructures can however be identified an exploited to perform propagation [7, 8].

Optional activities and Time Intevals Starting from 1994, *optional activities* are taken into account; namely, in [86] the activity representation of ILOG-Scheduler is extended, so that a 0 durations denotes a non-executing activity. In [122] specific propagation algorithms for optional activities are considered; they make use of *execution variables*, but activities are not required to be part of mutually exclusive groups. Optional activities requiring different resources can be used to model complex resource assignments decisions, by introducing proper side constraints.

In [80, 84, 83, 85], the constraint engine is extended to handle optional activities as first class variables: those are referred to as *time-interval variables*. A time interval variable $\bar{\mathtt{A}}_\mathtt{i}$ has values in the domain $\bot \cup \{[s, e) \mid s, e \in \mathbb{Z}, s \leq e\}$; namely, either the variable is *non-executed* ($\bar{\mathtt{A}}_\mathtt{i} = \bot$) or its value is *a half-open interval* $[s, e)$ with integer bounds. Time interval variables can be connected by precedence constraints, or can be organized in *alternative* and *hierarchical* blocks; each block corresponds to a macro-interval $\bar{\mathtt{A}}_\mathtt{i}$, spanning over a set of (possibly alternative) sub-intervals $\bar{\mathtt{A}}_\mathtt{j}$. Non-executed variables have no effect on the constraints they are involved in.

The so called *execution constraints* ($exec(\bar{\mathtt{A}}_\mathtt{i})$) force a variable to be executed and can be aggregated into logical expressions (e.g $exec(\bar{\mathtt{A}}_\mathtt{i}) \Rightarrow exec(\bar{\mathtt{A}}_\mathtt{j})$). Resource constraints are taken into account by associating time-intervals to step functions (referred to as *cumul functions*), representing the resource usage profiles. Complex resource allocation and scheduling problems can be modeled by encoding resource assignment as execution decisions for interval variables. Note however this technique requires the introduction of an exponential number of variables in case of independent resource groups (e.g. the workers and the factory rooms from Figure 1).

2.3 Mixed Integer Linear Programming models

Most of the OR literature about resource allocation and scheduling goes under the flag of the so-called Multi-mode Resource Constrained Scheduling Problem (MRCPSP), first introduced in [46]. Unlike in the classical RCPSP, each activity a_i of the MRCPSP can be executed in one out of a *set of possible modes* M_i. Each mode m_h represents an alternative way to execute the activity and specifies a set of resource requirements $rq_{i,k,h}$ and a duration value $d_{i,h}$.

Multiple activity modes give rise to several kinds of trade-off between (1) the duration of an activity and its use of resources (time/resource trade-off), (2) the duration of an activity and its cost (time/cost trade-off), (3) the quantity and combination of resources employed by the activity (resource/resource trade-off); hence, the MRCPSP can be thought as a generalization of other trade-off

$$
\begin{array}{lll}
\min & F(\mathtt{E}) & (1)\\
\text{s.t.} & \sum_{m_h \in M_i} \sum_{\tau=0}^{eoh} \mathtt{E}_{\mathtt{i},\mathtt{h},\tau} = 1 \quad \forall a_i \in A & (2)\\
& \sum_{m_h \in M_i} \sum_{\tau=0}^{eoh} \tau \cdot \mathtt{E}_{\mathtt{i},\mathtt{h},\tau} \leq \sum_{m_h \in M_j} \sum_{\tau=0}^{eoh} (\tau - d_{j,h}) \cdot \mathtt{E}_{\mathtt{j},\mathtt{h},\tau} \quad \forall (a_i, a_j) \in E & (3)\\
& \sum_{a_i \in A} \left[\sum_{m_h \in M_i} rq_{i,k,h} \sum_{\tau=\tau_0}^{\tau_0+d_{i,h}-1} \mathtt{E}_{\mathtt{i},\mathtt{h},\tau} \right] \leq c_k \quad \forall r_k \in R,\ \forall \tau_0 = 0 \ldots eoh & (4)\\
& \mathtt{E}_{\mathtt{i},\mathtt{h},\tau} \in \{0,1\} \quad \forall a_i \in A,\ \forall m_h \in M_i,\ \forall \tau = 0 \ldots eoh & (5)
\end{array}
$$

Figure 3: A Time Indexed model for non-preemptive MRCPSP

problems such as the Discrete Time-Resource Trade-off Problem (see [36, 42]). Similarly to alternative activities and time intervals, the multi-mode formulation requires to introduce an exponential number of modes to model independent resource assignments (e.g. workers and rooms in Figure 1).

Classical Mixed Integer Linear Programming (MILP) models for the MR-CPSP can be classified into *time indexed* and *disjunctive*; alternative formulations have been provided to cope with dimensionality issues (see [126]), with no substantial improvement. A model based on an activity-as-a-box representation is reported in [108].

Time Indexed Model In a time indexed model binary variables $\mathtt{E}_{\mathtt{i},\mathtt{h},\tau}$ are introduced to denote whether an activity a_i is scheduled to *finish* at time τ in mode m_h; note the different indexing use to distinguish CP end variables (i.e. $\mathtt{E}_\mathtt{i}$) from variables in this model. In Figure 3, we report the model used in [117]; in the model, *eoh* is the maximum possible finish time (end of horizon). Constraints (2) require each activity to be finished by the end of horizon. Constraints (3) enforce end-to-start precedence relations and constraints (4) resource capacity restrictions.

The time indexed model allows a linear representation of resource constraints; as a drawback, the use of a discrete representation of time sets scalability issues (as the number of variables depends on the length of the horizon). The presence of multiple modes further complicates the problem as it stresses dimensionality issues.

Disjunctive Model In a disjunctive model, a start variable $\mathtt{S}_\mathtt{i}$ is introduced for each activity a_i; mode assignments are represented by variables $\mathtt{M}_{\mathtt{ih}}$, such that $\mathtt{M}_{\mathtt{ih}} = 1$ iff activity a_i is executed in mode m_h; in practice, each $\mathtt{M}_{\mathtt{ih}}$ corresponds to a *set* of $\mathtt{X}_{\mathtt{ik}}$ assignments in the reference model from Section 2.1. A complete model (a slight elaboration over [55]) is shown in Figure 4. There, Constraints (7) model end-to-start precedence relations, Constraints (9) force a mode to be assigned to each activity. The notation $A(\mathtt{S}, \tau)$ refers to the set of tasks executing at time τ, i.e. such that $\mathtt{S}_\mathtt{i} \leq \tau < \mathtt{S}_\mathtt{i} + \sum_{m_h \in M(a_i)} d_{ih} \cdot \mathtt{M}_{\mathtt{ih}}$.

Disjunctive models require a smaller number of decision variables compared to time indexed ones; however, sets $A(\mathtt{S}, \tau)$ in Constraints (8) do not have a simple linear representation; this can be provided by preventing the overlapping execution of all Minimal Forbidden Sets [64, 63, 79], i.e. minimal size sets of

$$
\begin{aligned}
\min \quad & F(\mathtt{S},\mathtt{M}) && (6)\\
\text{s.t.} \quad & \mathtt{S}_j - \mathtt{S}_i \geq \sum_{m_h \in M_i} d_{ih} \cdot \mathtt{M_{ih}} \quad \forall (a_i, a_j) \in E && (7)\\
& \sum_{a_i \in A(\mathtt{S},\tau)} \sum_{m_h \in M_i} rq_{i,k,h} \cdot \mathtt{M_{ih}} \leq c_k && (8)\\
& \sum_{m_h \in M_i} \mathtt{M_{ih}} = 1 \quad \forall a_i \in A && (9)\\
& \mathtt{S_i} \geq 0 \quad \forall a_i \in A && (10)\\
& \mathtt{M_{ih}} \in \{0,1\} \quad \forall a_i \in A, \forall m_h \in M_i && (11)
\end{aligned}
$$

Figure 4: A Disjuctive model for non-preemptive MRCPSP

activities which would cause a resource over-usage (see Section 5.2). However, the number of minimal forbidden sets is in general exponential in the size of the graph[1], so that the method generally requires the addition of a large number of constraints. The issue is further stressed in MRCPSP by the need to take into account possible mode assignments; as a matter of fact, all approaches based on disjunctive models rely on specific search strategies to take care of resource constraints (see Section 5).

2.4 Decomposition Based Approaches

Allocation and scheduling problems have intrinsically hybrid nature, as they result from the combination of an assignment and a scheduling component. In particular, once a full resource assignment is available, an allocation and scheduling problem reduces to a pure scheduling problem (e.g. the MRCPSP reduces to classical RCPSP). It is therefore possible to decompose the overall problem into two separate, interacting stages.

Stripping out the allocation component makes the resulting problem much easier, as a consequence of: (1) the search space reduction; (2) the increased propagation effectiveness and (3) the availability of a much better established pool of algorithmic techniques (see the abundance of RCPSP literature, for example [25, 4, 53]). Decomposition is often used in MRCPSP heuristics, more or less explicitly ([68, 120] for some examples); in the context of exact approaches the main embodiment of this technique is Logic Based Benders' Decomposition Framework (LBD, formalized in [59]; similar approaches appear in [66, 31]).

LBD is a generalization of classic Benders' Decomposition in Operations Research [17]. The method breaks a combinatorial problem into a *master- and a sub-problem* (see Figure 5), which are solved in sequence; the master solution is used to prime the subproblem, then a cut is generated and the process repeats until convergence is reached. In the context of allocation and scheduling problems, the resource assignment part is usually tackled in the master problem, while the subproblem is a pure RCPSP instance. Unlike in the classical Benders' approach, LBD does not require the subproblem to be linear.

An attentive decomposition can result into dramatically smaller (and easier) subproblems. Furthermore, LBD allows one to mix heterogeneous techniques: while MILP models stand out as natural candidates for the resource assignment

[1] As a special case, the number of minimal forbidden sets is quadratic if only unary capacity resource are considered

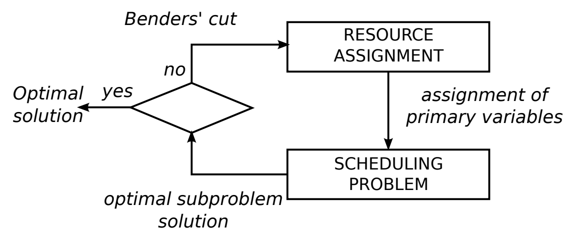

Figure 5: Structure of the Logic based Benders' Decomposition Approach

part, CP is much more effective in dealing with temporal domains and (non-linear) resource constraints. As a consequence, each technique is effectively employed for what it is best for, allowing impressive speed-ups [57, 61].

Logic Based Benders' Decomposition has been applied to allocation and scheduling problems in [67, 52, 57, 60, 61], to real time multiprocessor scheduling in [30] and to power consumption minimization in [107]. The best results are obtained when the subproblem can be split into *independent components*, based on the master problem solution (e.g. in [60]). In [20, 18], decomposition is recursively aplpied to break an overly complex allocation problem and obtain a balanced multi-stage LBD chain.

The main drawback with decomposed approaches is the loss of valuable information due to the decoupling between resource allocation and scheduling; in the context of LBD, particular care should be devoted in choosing a suitable decomposition and developing effective Benders' cuts (see Section 5).

2.5 Variants

Resource allocation and scheduling problems find motivation in real world applications, ranging from steel production planning [123] to software development and optimization [77]. Such a diversity gives rise to a number of problem specific restrictions and makes it hard to devise a uniform problem formalization; this is the main reason for focusing this survey on specific techniques and their composition, rather then on problem classification.

Nevertheless, some variations of the reference problem are sufficiently common to deserve dedicated discussion; those include the use of objective functions other than the makespan, the availability of resources of different types and generalizations of the precedence constraints.

2.5.1 Objective Function Types

Objective functions for Resource Allocation and Scheduling usually match those of pure scheduling problems; a comprehensive list can be found in [53].

Time Based Objectives Makespan minimization is by far the most common problem objective, but time based functions in general (e.g. lateness, tardiness, and earliness) have particular importance; the lateness LT_i of an activity a_i is the deviation between the end time $\mathtt{E_i}$ and a given due date dl_i, hence $LT_i = \mathtt{E_i} - dl_j$. The tardiness $TD_i = \max(0, L_i)$ is a similar measure, but cannot be negative; the earliness ER_i is defined analogously as $ER_i = \max(0, -L_i)$. Several time-based performance measure are proposed as possible objectives in [112], while [40] is the only approach taking into account robustness and rescheduling costs.

Regular and Non-Regular Objectives Problem objectives can be either regular or non-regular; the notion of *regular performance measure* is introduced in [105] for the RCPSP and extended in [114] to the multiple mode case; roughly speaking, a performance measure is called regular if the objective function value may only improve by reducing the end time of some activity (without changing the activity mode).

In the case of a regular objective, the set of optimal solutions always includes a *tight* schedule (see Section 4.2.3). For this reason several algorithms focus on the construction of tight schedules to reduce the search space and improve the run time; extending those approaches to non-regular objectives may be tricky. As a matter or fact, non-regular objectives are more popular in CP approaches, thanks to the flexibility of the search method.

Non-regular objectives include earliness associated costs, or the minimization of resource usage [117]; obtaining a smooth resource profile [112, 117] is another example of non-regular objective, modeled in a straightforward fashion with the time indexed approach, but more tricky for CP and disjunctive formalizations. Several types of regular and non-regular functions are taken into account in [82, 50]; the tardiness objective is considered in [58], while three non-regular objectives are considered in [80] (namely earliness/tardiness costs, number of executed tasks, the satisfaction degree of soft temporal constraints).

An interesting class of non-regular objectives (formalized in [89]) consists of functions not directly depending on start time assignments; the number of executed activities in [80] or the bus traffic in [90] are examples of such non-regular objectives. When tackled via a decomposition based approach, this class of functions gives rise to pure feasibility subproblems, with no cost function [57], due to the lack of direct dependency on start times.

Other Objective Functions Other objective functions are taken into account in the context of heuristic approaches, e.g. the Net Present Value in [95], a cost measure based on resource usage and mode assignment in [125] or weighted tardiness in [123].

2.5.2 Resource Types

Several resource types have been considered in the context of allocation and scheduling problems; the most common are renewable resources, non-renewable resources and doubly constrained resources [46].

Renewable Resources This is the simplest type of resource, considered in the classical RCPSP and in the reference problem from Section 2.1. The resource availability is renewed at each time unit and the capacity is constant. Examples include manpower, industrial machines or CPUs. Finding a feasible schedule with renewable resources is a polynomial problem if no deadline constraint or maximal time lag is specified (see Section 2.5.3).

Non-renewable Resources This type of resource (usually considered in all MRCPSP approaches) has a starting availability and is consumed throughout the whole scheduling horizon, until it is depleted; project budget is a good example of a non-renewable resource. Taking into account non-renewable resources generally makes finding a feasible mode assignment an NP-complete problem [74], so that the whole optimization process becomes considerably harder [112].

So-called cumulative resources (non-renewable resources which can be refilled by some activities) are considered in [98] in the context of a pure RCPSP problem. This type of resource is known in the CP community as *reservoir*.

Doubly Constrained Resources Those resources are constrained per period (e.g. per period cash flow) as well as for the overall project (e.g. total expenditures); doubly constrained resources can be modeled by combining a renewable and a non-renewable resource and are therefore never directly treated.

Partially Renewable Resources This type of resource requires the specification of a list of sub-periods $\Pi = \{\pi_0, \pi_1, \ldots\}$; the resource is renewed at the beginning of each sub-period. Partially renewable resources are a generalization of both renewable and non-renewable resources; they are introduced in [23] for the RCPSP and considered in [127] in the context of allocation and scheduling.

Variable Capacity and Requirements Resources with time *varying capacity* are taken into account in [114, 112]. In [11] it is shown (on the RCPSP) that resource capacities varying with time can be transformed into constant capacities if minimal and maximal time lags are available (see Section 2.5.3); this is done by introducing for each time interval with reducd capacity, an artificial activity consuming the resource; the activity position is fixed using a minimal and a maximal time lag. The special case of resource breaks, e.g. vacation time, and break-interruptible activities is considered in [28] (in the MRCPSP context) and in [3, 4] (in Constraint Programming).

Time varying *resource requirements* are instead considered in [44]; in principle, the time indexed model can easily take this case into account [127], but it is hard to estimate the resulting problem difficulty.

2.5.3 Precedence Constraints

Besides traditional end-to-start arcs, the most relevant classes of precedence constraints include:

Start/Start, Start/End, End/End Precedence Relations End-to-start relations can be generalized by introducing start-to-start, end-to-end and start-to-end precedence constraints. The new precedence relation types can be converted one into another, as described in [45] for the RCPSP if the durations are fixed. As a note, the multi-mode RCPSP with this kind of precedence relations is known as Generalized MRCPSP.

Generalized Precedence Relations and Time Windows This case subsumes the previous one; additionally, minimal and maximal time lags label precedence relation and constrain the time distance between the involved activities; formally in case of an end-to-start arc, the produced schedule must satisfy $\delta_{min} \leq \mathtt{E_j} - \mathtt{S_i} \leq \delta_{max}$; where δ_{min}, δ_{max} respectively are the the minimum and maximum time lag labeling arc (t_i, t_j). The Multi-mode RCPSP with this type of precedence constraint is known as MRCPSP with Generalized Precedence Relations (GPR) and is tackled with exact approaches in [37, 35].

Minimal time lags allow one to model setup time between specific pairs of activities; maximum time lags may model “best before” constraints, occurring for example in chemical- and food-industries. Time windows constraining the execution of each activity a_i between a release time rs_i and a deadline dl_i can be assimilated to minimal/maximal time lags from a fixed source node.

Approaches for the MRCPSP can be extended quite easily to take into account minimal time lags; maximal time lags and deadline constraints, however, make finding a feasible schedule NP-complete and require deeper algorithm modifications. CP approaches are more easily adjusted, since they handle activity release dates and deadlines by means of the domain store.

A maximal time lag on an arc (a_i, a_j) can be converted into a *negative* minimal time lag on the complementary arc (a_j, a_i), as shown in [25]. The resulting network contains a feasible schedule if and only if no cycle with positive length exists [11]; this stresses the analogy between project graph with time lags and Simple Temporal Networks in Artificial Intelligence [41], for which analogous results have been produced.

In allocation and scheduling problems, *the time lag may depend on the resource assignment* [48]; for the MRCPSP this translates into mode dependent time lags [108, 55]; despite being considered only by a few works, assignment dependent time lags are an actual need in many industrial applications (e.g. data communication costs in scheduling for multiprocessor systems [76]).

3 Propagation

Filtering and propagation algorithms for resource constraints have been developed by the CP community for Alternative Resources and Alternative Activities. *Temporal reasoning* with and without time lags has been investigated in the context of the Critical Path Method and Temporal Constraint Networks.

In the followings, we overview existing propagation techniques for resource allocation and scheduling problems. Additionally we recall that, if a decomposition approach is adopted, any technique for pure scheduling problems is applicable (refer to [25, 53] and [78, 4] for some references).

3.1 Temporal Reasoning

The classical Critical Path Method [70] relies on longest path computation to get bounds on the time window of each activity a_i. In particular, lower bounds (i.e. a so-called Earliest Start Schedule) are usually obtained by assuming each activity runs with the shortest possible duration. Temporal reasoning and time window determination has an important role in reducing the size of time-indexed models (see Section 2.3).

Issues with complex precedence relations If precedence relations other than end-to-start are considered and activity durations are variable (i.e. they depend on resource assignment), then a a makespan *reduction* can occur when an activity duration is *prolonged*. This happens if a backward arc is part of the critical path, for example in the chain: $\mathtt{E_i} \rightarrow \mathtt{E_j} \rightarrow \mathtt{S_j} \rightarrow \mathtt{S_k}$. In this case a_j is a *backward critical activity* (see [45]). In such a situation, determining the resource assignment with minimum project length may prove difficult even if no side constraint is specified [35].

Correct bounds are obtained in Constraint Programming by the use of *local* propagation on the precedence constraints [4] (instead of path computation) and in Temporal Constraint Networks [41] by relying on specific consistency algorithms. Nevertheless, when complex precedence constraints and resource dependent durations are taken into account, one should be aware that shortest activities do not necessarily result in the shortest possible schedule.

Time Window Rule Whenever during the solution process the start time of an activity is forced to occur out of its possible time window, the current search node can be fathomed. This is known in the MRCPSP literature as *time* *window rule* [54, 113, 55] and is naturally taken into account in CP by start time domains and temporal propagation. An extension of the rule to interruptible activities is proposed in [28].

3.2 Alternative Resources

Let a_i be an activity requiring $r_{i,\bar{R}}$ units of one out of a set $\bar{R}$ of m alternative resources, propagation can be performed as if a_i was split in m alternative activities $a_i^0, \dots a_i^{m-1}$, each requiring $r_{i,\bar{R}}$ units of a specific resource in $r_k \in \bar{R}$. Then an alternative resource constraint (see [48, 99]) maintains the constructive disjunction between the time window of the alternative activities:

$$EST(a_i) = \min_{r_k \in \bar{R}} \{EST(a_i^k)\} \qquad LST(a_i) = \max_{r_k \in \bar{R}} \{LST(a_i^k)\}$$
$$EET(a_i) = \min_{r_k \in \bar{R}} \{EET(a_i^k)\} \qquad LET(a_i) = \max_{r_k \in \bar{R}} \{LET(a_i^k)\}$$

any propagation technique for scheduling problems with no resource assignments can be used to compute the time window of each sub-activity a_i^k.

Alternatively [99], the whole set $\bar{R}$ can be treated as a single resource with capacity $\sum_{r_k \in \bar{R}} c_k$; then classical filtering algorithm for the cumulative constraint can be applied, but the resulting propagation is usually quite weak until the actual resource assignment is performed.

3.3 Alternative Activities

All approaches targeting alternative activities mix a *logical* layer (constraints on the execution variables) and a *temporal* layer (constraints on the start/end variables); exploiting the interaction between the layers is the critical point to enable effective propagation.

The first propagators for alternative activities, taking into account both execution and temporal variables, are discussed in [14], together with an extension of the edge-finding algorithm. Propagators for discrete resource constraints with *optional* activities (i.e. with no alternative execution constraint) have been considered in [122] and in [121] for the time-interval variables in IBM-ILOG CP Optimizer.

Additionally, *time-interval variables* (see Section 2.2) enable joint logical and temporal propagation through the interaction of a logical network (similar to the implication graph from [24]) and a Temporal Constraint Network. Nodes of the logical networks correspond to interval variables $\bar{\mathtt{A}}_\mathtt{i}$, while nodes in the temporal network are *interval endpoints*; in particular, here we respectively refer as $start(\bar{\mathtt{A}}_\mathtt{i})/end(\bar{\mathtt{A}}_\mathtt{i})$ to the start/end endpoint of interval variable $\bar{\mathtt{A}}_\mathtt{i}$.

Temporal constraints between two endpoints (say $end(\bar{\mathtt{A}}_\mathtt{i})$, $start(\bar{\mathtt{A}}_\mathtt{j})$), are in the form $exec(\bar{\mathtt{A}}_\mathtt{i}) \wedge exec(\bar{\mathtt{A}}_\mathtt{j}) \Rightarrow (end(\bar{\mathtt{A}}_\mathtt{i}) + \delta_{ij} \leq start(\bar{\mathtt{A}}_\mathtt{j})$, where δ_{ij} is a (possibly negative) time lag.

Whenever the logical network can infer the relation $exec(\bar{\mathtt{A}}_\mathtt{i}) \Rightarrow exec(\bar{\mathtt{A}}_\mathtt{j})$ the arc can propagate the conditional bounds from $start(\bar{\mathtt{A}}_\mathtt{j})$ to $end(\bar{\mathtt{A}}_\mathtt{i})$. Similarly, if the relation $exec(\bar{\mathtt{A}}_\mathtt{j}) \Rightarrow exec(\bar{\mathtt{A}}_\mathtt{i})$ can be inferred by the logical network, then the other half of the propagation can be performed. This allows temporal bound propagation even if the execution status is not yet fixed.

Nested Temporal Networks with Alternatives (TNA) [8] allow efficient logical reasoning. In particular, temporal and logical propagation rules for nested parallel and alternative blocks are presented in [8, 9]: in particular, (1) temporal bounds can be obtained based on the value of the execution variables and (2) activities may be deemed non-executable due to temporal restrictions. Finally, in [10] the authors discuss three methods to detect equivalence classes (activities with the same execution condition); despite the problem is NP-hard, some of those equivalence classes can be detected efficiently and used to perform propagation; this is done via identification of nested structures or by relying on singleton arc-consistency.

The issue of connecting a logical and a temporal layer is also tackled in the context of the *Disjunctive Temporal Problem with Finite Domain Constraints* (DTP_{FD}, see Section 2.2); here, as long as the finite domain variables $\mathtt{Y_i}$, $\mathtt{Y_j}$ associated to an edge are not fixed, the *constructive disjunction* of bounds $B_{ij}(\mathtt{Y_i}, \mathtt{Y_j})$ is used for temporal propagation. As in the case of TNAs, temporal reasoning can be used to filter out inconsistent values for the finite domain variables. Ensuring consistency of the finite domain network requires exponential time in general, but can be done efficiently for specific problem classes (an example is identified in the reference).

Alternative activities and some related propagation rules are finally discussed in the context of stochastic scheduling [22, 118]: despite the model is very different from the one considered here (alternatives are out of the user control), the developed propagation rules apply to resource allocation and scheduling problems.

4 Lower Bounds and Bounding Rules

In the context of the Multi-mode RCPSP people have focused on *bounding and dominance rules*; namely: (1) bounding rules refer to inferring restrictions on time windows, mode assignments and scheduling decisions; (2) dominance rules are analogous to symmetry breaking constraints: they identify classes of equivalent solutions (in terms of objective function value) and narrow the search space by forcing the selection of a specific one.

4.1 Lower Bounds

Lower bounds are a fundamental component of many search strategies (such as branch & bound, branch & cut, branch & price); in CP, a lower bound can be encapsulated in a global constraint, improving the effectiveness on optimization problems and providing access to useful information, such as reduced costs [49].

Computing good lower bounds for pure scheduling problems with non-unary resources and generic precedence constraints is notoriously difficult (see [96, 71, 26, 27, 21]). Not surprisingly, adding resource assignment decisions makes it even harder. The only exception are Critical Path based bounds (discussed in Section 3.1), which are cheap to compute and often effective; however, they do not perform well on instances with high Resource Strength (definition in [73]).

Ideally, the time indexed or the disjunctive formulation of the MRCPSP provide ready to use lower bounds; to the best of the authors knowledge, however, this approach is only adopted in [127] for makespan minimization. The method makes use of a bounding technique in two stages: in a preprocessing step, a lower bound on inter task distances is used in conjunction with Critical Path reasoning to derive tighter time windows and reduce the model size; the bound is then employed during the solution process to fathom search nodes. Valid linear inequalities (cuts) are applied to improve tightness.

Lower bounds for cost functions other than the makespan are considered in

[48] (setup times dependent on the resource assignment) and in [40] (rescheduling cost). An interesting technique exploited in [127] and [40] to strengthen a lower bound consists in performing truncated tree search with a maximum depth; the weakest bound on the tree frontier is valid for the root node. In [40], the approach is further improved by means of an effective look ahead technique from [39].

4.2 Bounding Rules

Bounding rules have been developed by the OR community for the RCPSP and MRCPSP problems; those are tree reduction techniques and check if the current search node can be preventively fathomed. Unlike filtering algorithms, bounding rules are not attached to a constraint, but are executed as part of the search method; as a consequence, the rule formulation is tailored on a specific branching scheme (see Section 5.2), despite the main underlying ideas usually have broader applicability.

There is no automatic coordination mechanism (such as propagation), so that the combination of different rules is up to the algorithm designer. From a CP perspective, bounding rules may serve as a basis for the development of filtering algorithms.

4.2.1 Feasibility Based Rules

Nonrenewable Resource Rule This rule appears in [54], in [113] (with the name "nonrenewable resource consumption") and in [40] (as "resource infeasibility rule"). The rule considers each *nonrenewable* resource r_k: if scheduling each currently unscheduled activity in the mode with the lowest request for r_k would exceed its capacity c_k, then the current partial schedule cannot be feasibly completed. The rule is given a very efficient static formulation (see Section 4.2.2) in [112] (where it is referred to as "input data adjustment"). This kind of reasoning is subsumed in CP by usual constraint propagation on the capacity constraints and the resource assignment variables.

Non-delayability and Immediate Selection Those two rules, respectively reported in [113, 112, 54, 25] and [55] are based on the principle that a forced scheduling choice should be immediately performed. The general rule states that, if the current set of scheduling options contains an activity with no other chance to be scheduled, the choice should be immediately performed. The mentioned works contain specific adaptations to the different scheduling schemes.

Single Enumeration Rule The single enumeration rule, introduced in [114] and further applied and refined in [112, 54] is a type of dynamic symmetry breaking constraint for precedence tree branching (see Section 5.2). The rule targets two activities a', a'', scheduled in two subsequent search steps i and $i+1$ in mode m' and m''; if their assigned start times do not depend on which activity is scheduled at step i and which one at $i+1$, then only one sequence needs to be considered.

4.2.2 Static Bounding/Dominance Rules

Static bounding rules are introduced in [113] and used in most of the exact approaches for the MRCPSP developed later on. Static rules are applied *prior* to the beginning of search and consist in the removal of non-executable modes, inefficient modes and redundant resources. The rule application is iterative, until a fix-point is reached.

Non-executable Modes In [113, 112] a mode m_h for an activity a_i is defined non executable w.r.t. a renewable resource r_k if its requirement exceeds the resource capacity, i.e. $r_{i,k,h} > c_k$; a mode is non-executable w.r.t. a nonrenewable resource r_k if its requirement exceeds the resource capacity, reduced by the sum of the minimum requirements of all other activities, i.e. if $r_{i,k,h} > c_k - \sum_{a_j \neq a_i} \min_{m_r \in M_j}(r_{j,k,r})$.

Inefficient Modes A mode m_h for activity a_i is defined in [113] as *inefficient* if there exist some other mode m_r for a_i with shorter duration and lower consumption for all resources. Note that the removal of inefficient modes requires the objective function to be regular in order to be applied; additionally, some caution must be observed with precedence constraint other than end-to-start (see Section 2.5.3).

Redundant Resource A nonrenewable resource r_k is said to be redundant if its capacity exceeds the sum of the maximum consumption of all activities, i.e. $c_k > \sum_{a_i \in A} \max_{m_h \in M_i}(r_{i,k,h})$.

4.2.3 Dominance Rules

Dominance rules are similar to symmetry breaking constraints; they are based on the observation that, given some problem assumptions, there must exists an optimal schedule with specific properties. One can therefore focus on the generation of a schedule with such properties and reduce the search space, with no loss of optimality.

Dominance rules should be applied only after the validity of their enabling assumptions has been checked. In particular, most of the dominance rules are devised for *regular* cost functions; their applicability to non-regular objectives and other variants is seldom discussed in details in the literature. All presented dominance rules target a search process where a partial schedule is built by assigning a start time to unscheduled activities, proceeding in chronological order.

Left-shift Rules This extremely important class of dominance rules is based on a property of regular objective functions, and on the concept of left-shift (discussed in details in [115]). Some definitions are briefly listed.

A *one period left shift* is an operation on a single activity a_i within a feasible schedule S, which derives a feasible schedule S', such that $\mathtt{E}'_\mathtt{i} = \mathtt{E}_\mathtt{i} - 1$ (where $\mathtt{E}'_\mathtt{i}$ is the end time of a_i in S') and no other schedule modification occurs. A *local left shift* is a left shift of activity a_i which is obtainable by one or more successively applied one-period left shifts. A *global left shift* of activity a_i is a left shift which is not obtainable by a local left shift. A *multi-mode left shift* [113] is a left shift of a_i where the activity is allowed to change mode.

Then, a schedule is *semi-active* if it is feasible and no activity can be locally left-shifted; a schedule is *active* if it is feasible and no activity can be left-shifted (either locally or globally). A *tight* schedule is a feasible schedule where no multi-mode left-shift (either local or global) can be performed. Now, with a regular objective function (such as the makespan), the set of optimal schedules is guaranteed to contain an active schedule (in the case of the RCPSP) or a tight schedule (for the MRCPSP).

A pruning rule can be devised based on those properties; the general version of this *left shift rule* states that a partial schedule in which an activity a_i can be left-shifted without violating the precedence and the resource constraints needs not to be completed (as it is dominated by another active or tight schedule). Specific left-shift rules depend on the considered type of left-shift and are employed in [112, 40, 54, 113] (and described in [25]).

Since the application of dominance rules is restricted to specific problem assumptions, such rules are usually not provided by off-the-shelf constraint solvers. As an exception, the left-shift rule is at the base of the schedule or postpone strategy (see [87, 50, 109] and Section 5.1). This is due both to its effectiveness and to the broad diffusion of regular cost functions in practical settings.

Multi-mode Rule This rule (used in [112, 54, 25, 113, 36]) is based on the so-called mode-reduction operation. Similarly to the case of left-shifts, some definitions are given prior to the actual rule statement.

A *mode reduction* [113] on an activity a_i within a feasible schedule is an operation changing the mode of a_i to one with shorter duration, without changing its finish time and without violating the constraints or changing the modes or finish times of the other activities. A schedule is called *mode-minimal* if no mode reduction can be performed. Finally, if there is an optimal schedule for a given instance, then there is an optimal schedule which is *both tight and mode-minimal*; some care must be observed with non-standard precedence constraints (see Sectin 2.5.3). Note that there may be tight schedules which are not mode-minimal, and mode-minimal schedules which are not tight (for an example see [113]).

The rule states that, if a mode reduction can be performed on an activity a_i with $\mathtt{E_i}$ equal to the current scheduling time, then the current partial schedule needs not to be completed.

Order Swap Rule An *order swap* [54, 25] is an operation on a feasible schedule targeting two activities a_i, a_j with $j > i$, such that a_i, a_j are scheduled in sequence, i.e. $\mathtt{E_i} = \mathtt{S_j}$. The order swap consist in an exchange of the start time of the two activities, with no violation of precedence or resource constraint; changing the mode of any activity or the start time of any activity other than a_i and a_j is not allowed.

A schedule in which no order swap can be performed is called *order monotonous*. If the order swap does not affect the objective function value (this is the case for the makespan), the set of order monotonous schedules is guaranteed to contain an optimal schedule. Therefore, before an activity a_i is scheduled at time τ, if an order swap is allowed with any scheduled activity having end time τ, then the current search node needs not to be further extended. Note the order swap subsumes the immediate selection rule.

4.2.4 Multi-mode Cutset Rules

This family of rules requires to store information about past search. During the solution process, the current partial schedule is compared with the stored data; in case any solution obtainable from the current partial schedule cannot be better than the solution obtained from a previously evaluated partial schedule, then backtracking is performed. The presented formulation of the cutset rules is devised for a makespan minimization objective, but does extend to regular measure of performance [112].

Given a partial schedule $\bar{S}$, the *cutset* $C(\bar{S})$ is the set of activities scheduled so far; besides the cutset, the rule requires to store the completion time $E(\bar{S})$ (i.e. the highest end time among activities in $\bar{S}$) and the leftover capacities of all nonrenewable resources. Then:

Rule 1: Dominated Heads let $\bar{S}'$ be the partial schedule to be extended at time τ in the current search step, having cutset $C(\bar{S}')$; if a stored partial schedule $\bar{S}''$ exists, with:

- the same cutset, i.e. $C(\bar{S}'') = C(\bar{S}')$,
- lower or equal completion time, i.e. $E(\bar{S}'') \leq E(\bar{S}')$,
- higher or equal leftover capacities for all non-renewable resources,

then the current partial schedule needs not to be completed.

A second rule is presented in [112] and bounds the schedule span necessary to complete the current partial schedule; the rule requires to store, for each visited partial schedule $\bar{S}$ the updated latest finish time $LET(\bar{S}, a_i)$ of a_i, after all possible continuations of $\bar{S}$ have been explored. Then:

Rule 2: Incompletable Tails let $\bar{S}'$ be the partial schedule to be extended at time τ in the current search step, having cutset $C(\bar{S}')$; if a stored partial schedule $\bar{S}''$ exits, with:

- the same cutset, i.e. $C(\bar{S}'') = C(\bar{S}')$,
- higher or equal leftover capacities for all non-renewable resources,
- $\tau + LET(\bar{S}'', a_i) - E(\bar{S}'') + 1 > LET(a_i)$,

then the current partial schedule cannot be completed to a makespan better the the current $LET(a_i)$. Cutset rules are described in [40, 54, 36, 25].

4.2.5 Effectiveness of the Bounding Rules

Some experimental evaluation of the described rules is provided in [113, 112, 54, 40]; additionally, [112] provides some details about rule implementation. An overall thorough comparison is difficult, since different works have considered different bounding rules (and sometimes targeted different instance sets). As a general remark, bounding rules are usually fruitfully combined, i.e. there are not sharp dominance relations.

In general, the non-renewable resource rule is considered to be among the most effective technique and provides the highest speed-up both in [113] and [112]; the reported improvement is less substantial, but still remarkable, in [40]. The effectiveness of the left-shift rule is also well documented; interestingly, the best results are usually obtained through the application of the *local* version of the rule, with the global one providing minor improvements. The single enumeration rule has a fundamental role within precedence tree branching [54]; the multi-mode rule performs nicely in the comparison from [113].

Among the cutset rules, “dominated heads” performs very well, definitely much better than “incompletable tails”. The “immediate selection” rule tends to be effective for small instances, but quickly becomes less likely to fire (and more expensive) on instances with more than 10 activities. Statc bounding rules are very effective for MRCPSP instances with high Resource Strength (see [113, 73]). The order-swap rule introduced in [54] is as effective as the local left shift one.

5 Search

Exact methods for resource allocation and scheduling usually build over search strategies for pure scheduling problems; most of the developed methods are based on tree search, with the exception of Logic Based Benders' Decomposition (see Section 2.4); in particular, Depth First Search (DFS) is the most common choice, in order to restrain memory requirements.

5.1 Search Strategies in Constraint Programming

Exact CP algorithms for mixed resource allocation and scheduling problems are usually based on DFS. The nodes of the search tree represent *partially instantiated schedules*: scheduled activities have fixed start time, while the time window of unscheduled activities is maintained through the domains of start/end variables and updated via propagation. Similarly, bounds on the cost function are updated by the propagation of optimization constraints.

Tree search proceeds by opening choice points and posting different constraints along each branch; distinct strategies differ for the type of posted constraints and for the heuristics used to take non-deterministic decisions.

Schedule Or Postpone This is a tree-search strategy proposed in [87]. At each step an activity a_i is selected; usually, this is the one with the lowest $EST(a_i)$, while $LET(a_i)$ is used to break ties. Then a binary choice point is opened and a_i is scheduled at $EST(a_i)$ along the first branch; along the second branch the activity is marked as non-selectable (i.e. *postponed*) until its earliest start time is modified by propagation.

The strategy results in the production of *active* schedules; hence it requires the objective function to be regular to guarantee optimality; some care must be taken with Generalized Precedence Relations as described in Section 2.5.3. The schedule-or-postpone strategy has been extended to deal with resource assignments in [19], by adding a resource assignment stage before the scheduling decision is taken. Compared to MRCPSP branching with left-shift rules, this strategy provides better support for side constraints, thanks to the use of a CP model.

Successor Or Successor-but-not-next This search strategy is employed in [48] and makes use of binary choice points; let L be the set of the last activities scheduled on each resource; along the left branch, the activity a_i with minimum earliest start time is scheduled to be *next* of some activity a_j in L; on the right branch a_i is forced to be a *successor, but not next*. The actual predecessor activity in L depends on the resource a_i is assigned to, which is chosen *after* the scheduling decision. Note this approach does not require to immediately assign a start time to the selected activity.

Precedence Constraint Posting (PCP) This search method proceeds by resolving possible resource conflicts through the addition of precedence constraints. The idea was successfully applied in a CP framework for pure scheduling problems in [32, 104] (within the Iterative Flattening heuristics) and in [79] (in a tree-search method).

In particular, the latter approach is based on the systematic identification and resolution of so called Minimal Critical Sets (see Section 5.2); MCSs can be identified via enumeration [79], sampling over an earliest start solution [104], by computing resource envelopes [104], or by the solution of min-low problem

[88]. An MCS is *resolved* by posting a precedence constraint between any pair of activities a_i, a_j in the set (i.e. a *resolver*). Branching is done either by enumerating all possible conflict resolution choices, or by opening a binary choice point where a selected precedence constraint (a_i, a_j) is alternatively *posted* or *forbidden* [88]. Some heuristics (e.g. [111, 33]) is used to choose an MCS to branch on and to rank the resolvers.

Ranking heuristics for resolvers have been extended in [14] to deal with alternative activities; the approach exploits the so-called *probability* of execution (PEX). Rather than a measure of the occurrence probability of an uncontrolled event, $PEX(a_i)$ is a theoretical estimate of the likelihood that an activity a_i is *selected for execution*.

Two-stage search Here, we refer as two-stage search to any tree search method taking into account resource assignment and scheduling variables in successive, distinct phases. This is the default search method for alternative resources in ILOG Scheduler [62], where all resource assignment decisions are taken in a first stage, and a branching scheme for pure scheduling is then applied. The approach is also applied to the Disjunctive Temporal Problem with Finite Domain constraints (DTP_{FD}); in such context, however, a least commitment approach partially inverting the two phases is also investigated; in this case some ordering decision are taken on the temporal layer, *before* finite domain variables (e.g. resource assignments) are performed; actual start times are decided in a final step.

Heuristic Commitment Techniques Heuristic Commitment Techniques rely on a probabilistic criticality measure, enabling one to identify the most critical resource r_k and the most critical time point τ; then sequencing decisions are taken on the activities with a chance of overlap at τ and cause a conflict on r_k. Texture-based Heuristics (see [15, 16], or Force Directed Scheduling [103]) can be used as a criticality measure; those are adapted in [14] to take into account alternative activities; in the same work the branching scheme is extended so as to incorporate in the choice point the decision to select (or not) a target activity for execution.

Left-Justified Random This method [101] finds the smallest earliest *finish* time of all the unscheduled activities and then identifies the set of activities which are able to *start* before this time. One of the activities in this set (say a_i) is selected randomly and scheduled at its earliest start time. When backtracking, the alternative commitment is to update the earliest start time of the activity to the minimum earliest finish time of all other activities on the same resource as a_i. In the case of alternative resources, activities are considered only if the corresponding execution variable is not bound to 0 (see [14]); when the activity is scheduled, it is simultaneously selected for execution.

5.2 Branching Schemes for the MRCPSP

All the exact approaches developed for the MRCPSP are based on tree search; unlike in Constraint Programming, where a search node always corresponds to a state of the domain store, branching schemes from the Operations Research literature rely on different types of schedule representation.

Precedence Tree Branching This branching strategy dates back to [117], but received a major improvement by Patterson in [102]. Each node of the search tree corresponds to a resource- and precedence-feasible *partial schedule*, i.e. a schedule of a set S of activities, built in chronological order from time 0. No updated information (e.g. range of possible start times) is maintained for unscheduled activities.

The strategy consists in scheduling at each step of the search tree an activity whose predecessors have all been *scheduled*. Therefore, for each search node with partial schedule S a set of *eligible* activities $E(S)$ is univocally defined. On backtrack, a different activity is chosen, so that each path from the root of the search tree to the leaves corresponds to a possible linearization of the partial order induced on A by the precedence graph. A *mode alternative* within this branching scheme is an assignment of a mode m_h to the target activity a_i, which is performed after the scheduling decision. On backtrack, different modes are tested, so that a scheduling decision on a_i is only undone once all modes for a_i are tested.

The original algorithm by Patterson is improved in [114] and [112], in particular with the introduction of bounding rules. The structure of the approach is well described in [54]. The precedence tree method suffers from symmetry issues; in particular, if two activities a_i, a_j can be independently assigned the same start time, the method will always test both enumeration sequences; this is countered by the application of the single enumeration rule (which in fact provides the highest benefits on this branching scheme).

Mode and Delay Alternatives This branching method is introduced in [34, 43] for the RCPSP and is adapted to the multi-mode case in [113]; it is well described in [54] and recently used in [40]. Each node of the search tree is associated to a feasible partial schedule S and a time instant τ. A clear distinction is then made between completed activities at time τ (say $C(S,\tau)$) and activities in process (say $P(S,\tau)$); eligible activities for scheduling are those whose predecessors have all completed execution.

Then an attempt is made to schedule *all* eligible activities and they are added to the set of activities in process. Of course this may cause a conflict; in such a case the method branches by *withdrawing* from execution so-called delay alternatives. Those are: (1) activities in process, i.e. in $P(S,\tau)$ and (2) such that, if they are removed, no resource conflict occurs. A delay alternative is called minimal if none of its proper subsets is a delay alternative; branching on minimal delay alternatives is sufficient to explore the whole search space. If no resource conflict occurs, the only minimal delay alternative is the empty set.

This method differs from the precedence tree based one in two regards: (1) the process branches on *sets* of activities and (2) scheduled activities may be withdrawn from execution. The applicability of the approach requires constant resource capacities; as a consequence, only finish times of scheduled activities need to be considered for starting unscheduled ones.

Activities are assigned a mode when they are first inserted in the $P(S,\tau)$, hence they retain the mode assignment when they are withdrawn from execution. As a consequence, when in the procedure the simultaneous execution of all eligible activities is probed, some activities already possess a mode, while the remaining ones are modeless. A *mode alternative* in this search strategy is a mapping that assigns a mode to every activity in the modeless eligible set; on backtracking, when the branch corresponding to a delay alternative has been completely explored, a different mode alternative is picked.

Mode and Extension Alternatives This method was proposed in [116] for the RCPSP, while the adaptation to the multi-mode case is discussed in [54]. The approach is similar to mode and delay alternatives: each search node corresponds to a partial schedule S and a a time instant τ, for which sets $C(S, \tau)$ and $P(S, \tau)$ are identified. The activities with all predecessors in $C(s, \tau)$ form the *eligible set* $E(S, \tau)$.

The current partial schedule is extended by starting a *subset* of the eligible activities without violating the renewable resource constraints; conversely, in delay alternatives *all* eligible activity are started, then some are withdrawn from execution.

In order to secure that the algorithm terminates, empty extension alternatives may be disregarded if the $P(s, \tau)$ set is empty (no activity is in process). However, if there are currently activities in process, the empty set is always an extension alternative which must be tested in order to guarantee optimality; in case this is not done, the algorithm only builds so called non-delay schedules [72] and may miss the optimal solution (even in case the objective is regular).

A *mode alternative* is a mapping of modes to activities and occurs as soon as they become *eligible*, before an extension alternative is selected. The backtracking mechanism is the same as for delay alternatives. This branching scheme is proven to be dominated by precedence tree and delay alternative branching in [54], at least when no bounding rule is applied.

Dichotomization A branching scheme based on dichotomization is proposed for the MRCPSP in [127]. The approach operates on a time indexed model and is based on Special Ordered Sets (SOS, [12]); in detail, each considered SOS includes all the binary variables $\mathtt{E}_{\mathtt{i},\mathtt{h},\tau}$ referring to a single activity.

Given a fractional LP solution, branching can be performed by splitting a SOS, based on the average finish time of an activity a_i (let this be τ_b); the first subset contains variables with $\tau \leq \tau_b$, the second with $\tau > \tau_b$. Two branches are defined by respectively setting the variables in each subset to zero. The search method is coupled with a bound-tightening step following each branch and the use of local branching [47] to quickly find an initial good solution; the approach obtains promising results.

Minimal Forbidden Sets The idea of Minimal Forbidden Sets dates back to the early 80s [64, 63]; those are minimal size sets of activities causing a resource over-usage in case they overlap. Minimal Forbidden Sets are known in Constraint Programming as Minimal Critical Sets [32, 81, 104].

Branching on MCS can be performed by using *resolvers* (the same as in Section 5.1), or alternatively by posting *disjunctive precedence constraints*, i.e. by requiring a specific activity a_i in the MCS to execute after *at least one* other activity a_j in the set; the specific a_j causing the delay may be left undecided until all start times are assigned. The method is devised in [110] for the RCPSP and is applied in [55] to the multi-mode case. With this branching strategy, a search node corresponds to *fictitious* rather than to a partial schedule. A fictitious schedule assigns a set of possible modes and a provisional start time to each activity of the project; the resulting representation is very similar to the one obtained in CP via the domain store.

Disjunctive precedence constraints are based on the same idea of delay alternatives, but enable one to consider conflicts in non-chronological order; in particular, the method described in [55] systematically branches on the (estimated) hardest decision. The specifics rule applied depends on the type of this decision; if this is a mode decision, one branches over the possible mode assignments for the corresponding activity. If it is a conflict decision, each branch is a possible conflict resolution.

The use of disjunctive precedence relations allows a remarkable reduction of the search tree compared to binary resolvers; as a drawback, with this type of precedence relations the feasible space becomes a *union* of convex polyhedra and Generalized Arc Consistency on the temporal constraints can no longer be achieved in polynomial time.

5.3 Logic Based Benders' Decomposition

To the best of the authors knowledge, Logic Based Benders' Decomposition (LBD) is the only exact approach non based on tree search in the context of resource allocation and scheduling problems.

LBD solves a problem by enumerating values of the *master problem* variables (so called *primary variables*); let $\mathtt{X}$ be the set of master problem variables (allocation decisions) and let $F(\mathtt{X})$ be the master objective function. For each set $\bar{\mathtt{X}}$ of enumerated values, the method solves the subproblem that results from fixing the $\mathtt{X}$ variables to these values. The solution of the subproblem is used to generate a *Benders' cut* (i.e. a constraint). The next set of values for the primary variables is obtained by solving the master problem, which contains all the Benders cuts so far generated. The process continues until the master problem and subproblem converge in value.

The generated Benders' cut is the solution of the so-called *inference dual*; for a *minimization* problem the inference dual is the problem of finding the largest lower bound β^* which can be inferred from the current assignment $\bar{\mathtt{X}}$ of master problem variables. Then, the key step of the process is to identify a bounding function $\beta_{\bar{\mathtt{X}}}(\mathtt{X})$ such that:

$$\beta_{\bar{\mathtt{X}}}(\mathtt{X}) = \begin{cases} \beta^* \text{ if } \mathtt{X} = \bar{\mathtt{X}} \\ \beta'(\mathtt{X}) \leq F(\mathtt{X}) \text{ otherwise} \end{cases}$$

the subscript $\bar{\mathtt{X}}$ of the bounding function denotes the $\mathtt{X}$ values used for its construction; $\beta_{\bar{\mathtt{X}}}(\mathtt{X})$ equals the tightest lower bound when $\mathtt{X} = \bar{\mathtt{X}}$, otherwise it provides a valid (likely more loose) bound. The Benders' cut consists in forcing the master-problem objective to be greater or equal to $\beta_{\bar{\mathtt{X}}}(\mathtt{X})$. The outlined method is complete, i.e. guaranteed to provide the optimal solution.

The simplest form of Benders' cut is a no-good, forbidding the last $\bar{\mathtt{X}}$ assignment; stronger cuts can be obtained by relying on the problem formulation [61, 57]. Alternatively the cut can be strengthen with explanation minimization algorithms (such as the one in [38], or the faster version proposed in [69]); the approach is based on the repeated solution of a relaxed scheduling problem and is viable provided this can be performed sufficiently fast (e.g. in [30, 57, 18]).

Subproblem Relaxation The fundamental drawback of LBD is the risk of loosing valuable information due to the decomposition (e.g. if master and problem variables are connected by tight constraints); hence, extra care should be put in breaking the problem in a proper manner. As a way to mitigate the issue, a *subproblem relaxation* [60] can be embedded in the master problem in the form of a lower bounding function $G(\mathtt{X})$ on the objective of the master problem;

the actual formulation has to be given case by case. The use of a subproblem relaxation can have a strong impact on the performance [57, 18].

6 Conclusions

We provided an over-view of state of the art approaches for a class of resource allocation and scheduling problems, arising in real world settings in a wide diversity of flavors. Given the amount of problem variants we chose to focus this work on techniques to address individual problem traits, rather than on devising an exhaustive (most likely too complex) classification.

In particular, we mainly drew the presented pool of algorithms and methods from scheduling related OR and CP literature. Constraint Programming is a natural candidate to support the integration of heterogeneous techniques: its typical distinction between model, propagation and search provided the backbone for the work organization. Hybrid methods (in particular Logic based Benders' Decomposition) were given prominent importance, as they proved to be particularly effective on allocation and scheduling problems.

We decided to limit our discussion to exact approaches; however, one should be aware that, given the impressing complexity of this class of problems, a consistent number of works from the literature adopts heuristic solution methods. Genetic Algorithms (GA) such as [120, 92] and Large Neighborhood Search (LNS) [80, 82] are worth to mention, due to the promising results. The use of so-called justification [119] and implicit decomposition (e.g. [120]) seems to be among the most important ingredients of an effective GA. LNS proved extraordinary robust on a wide range of different problem variants and is particularly relevant in the context of this work, since it builds over complete tree search.

Finally, there is a manifest lack of stochastic allocation and scheduling approaches, most likely due to the prohibitive complexity one can expect from such a combination. Nevertheless, there are growing practical motivations to invest efforts in devising viable techniques to deal with uncertainty, e.g. process variation [124] and duration uncertainty [93] in the field of system design.

References

[1] A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, 17(7):57–73, 1993.

[2] K. R. Baker. *Introduction to sequencing and scheduling*. John Wiley & Sons, 1974.

[3] P. Baptiste, P. Laborie, C. Le Pape, and W. Nuijten. Constraint-based scheduling and planning. *Foundations of Artificial Intelligence*, 2:761–799, 2006.

[4] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based scheduling*. Kluwer Academic Publishers, 2001.

[5] R. Barták and O. Cepek. Temporal networks with alternatives: Complexity and model. In *Proc. of FLAIRS*, pages 641–646. Citeseer, 2007.

[6] R. Barták and O. Cepek. Nested Precedence Networks with Alternatives: Recognprecedenceition, Tractability, and Models. In *Proc. of AIMSA*, pages 235–246, 2008.

[7] R. Barták, O. Cepek, and P. Surynek. Modelling Alternatives in Temporal Networks. In *Proc. of IEEE SCIS*, pages 129–136, April 2007.

[8] R. Barták and O. Čepek. Nested temporal networks with alternatives. *Proc. of SAC*, page 156, 2008.

[9] R. Barták, O. Čepek, and M. Hejna. Temporal Reasoning in Nested Temporal Networks with Alternatives. *Recent Advances in Constraints*, pages 17–31, 2008.

[10] R. Barták, O. Čepek, and P. Surynek. Discovering implied constraints in precedence graphs with alternatives. *Annals of Operations Research*, 180(1):233–263, December 2008.

[11] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240, 1988.

[12] E. M. L. Beale and J. J. H. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10(1):52–69, 1976.

[13] J. C. Beck and M. S Fox. Scheduling Alternative Activities. In *Proc. of AAAI/IAAI*, pages 680–687, 1999.

[14] J. C. Beck and M. S. Fox. Constraint-directed techniques for scheduling alternative activities. *Artificial Intelligence*, 121(1-2):211–250, 2000.

[15] J. C. Beck and M. S. Fox. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. *Artificial Intelligence*, 117(1):31–81, 2000.

[16] J.C. Beck. *Texture measurements as a basis for heuristic commitment techniques in constraint-directed scheduling*. PhD thesis, University of Toronto, 1999.

[17] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, (4):238–252, 1962.

[18] L. Benini, M. Lombardi, M. Mantovani, M. Milano, and M. Ruggiero. Multi-stage Benders Decomposition for Optimizing Multicore Architectures. In *Proc. of CPAIOR*, pages 36–50. Springer, 2008.

[19] L. Benini, M. Lombardi, M. Milano, and M. Ruggiero. A Constraint Programming Approach for Allocation and Scheduling on the CELL Broadband Engine. In *Proc. of CP*, pages 21–35. Springer, 2008.

[20] L. Benini, M. Lombardi, M. Milano, and M. Ruggiero. Optimal Allocation and Scheduling for the Cell BE Platform. *to appear on: Annals of Operations Research*, 2010.

[21] L. Bianco and M. Caramia. A new lower bound for the resource-constrained project scheduling problem with generalized precedence relations. *Computers & Operations Research*, 2009.

[22] J. Bidot, T. Vidal, P. Laborie, and J. C. Beck. A General Framework for Scheduling in a Stochastic Environment. In *Proc. of IJCAI*, pages 56–61, 2007.

[23] J. Böttcher, A. Drexl, R. Kolisch, and F. Salewski. Project Scheduling Resource Under Partially Renewable Constraints. *Management Science*, 45(4):543–559, 1999.

[24] R.I. Brafman. A simplifier for propositional formulas with many binary clauses. *IEEE Transactions on Cybernetics*, 34(1):52–59, 2004.

[25] P. Brucker, A. Drexl, R. H. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.

[26] P. Brucker and S. Knust. A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, 127(2):355–362, 2000.

[27] P. Brucker and S. Knust. Lower bounds for resource-constrained project scheduling problems. *European Journal of Operational Research*, 149(2):302–313, 2003.

[28] J. Buddhakulsomsiri and D. Kim. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, 175(1):279–295, November 2006.

[29] J. Buddhakulsomsiri and D. S. Kim. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, 178(2):374–390, April 2007.

[30] H. Cambazard, P.E. Hladik, A.M. Déplanche, N. Jussien, and Y. Trinquet. Decomposition and learning for a hard real time task allocation problem. *Proc. of CP*, 3258:153–167, 2004.

[31] H. Cambazard and N. Jussien. Integrating Benders Decomposition Within Constraint Programming. In *Proc. of CP*, pages 752–756. Springer, 2005.

[32] A. Cesta, A. Oddi, and S. F. Smith. Iterative Flattening: A Scalable Method for Solving Multi-Capacity Scheduling Problems. In *Proc. of AAAI/IAAI*, pages 742–747, 2000.

[33] C. C. Cheng and S. F. Smith. Applying constraint satisfaction techniques to job shop scheduling. *Annals of Operations Research*, 70:327–357, 1997.

[34] N. Christofides, R. Alvarez-Valdes, and J. M. Tamarit. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273, 1987.

[35] B. De Reyck. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2):538–556, December 1999.

[36] B. De Reyck, E. Demeulemeester, and W. Herroelen. Local search methods for the discrete time/resource trade-off problem in project networks. *Naval Research Logistics*, 45(6):553–578, 1998.

[37] B. De Reyck and W. Herroelen. The Multi-Mode Resource-Constrained Project Scheduling Problem With Generalized Precedence Relations, 1997.

[38] N. J. L. de Siqueira and J. F. Puget. Explanation-Based Generalisation of Failures. In *Proc. of ECAI*, pages 339–344, 1988.

[39] F. Deblaere, E. Demeulemeester, and W. Herroelen. Exact and heuristic reactive planning procedures for multimode resource-constrained projects. *Open Access publications from Katholieke Universiteit Leuven*, 2008.

[40] F. Deblaere, E. Demeulemeester, and W. Herroelen. Reactive scheduling in the multi-mode RCPSP. *Computers & Operations Research*, pages 1–12, January 2010.

[41] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[42] E. Demeulemeester, B. De Reyck, and W. Herroelen. The discrete time/resource trade-off problem in project networks: a branch-and-bound approach. *IIE transactions*, 32(11):1059–1069, 2000.

[43] E. L. Demeulemeester and W. Herroelen. A branch-and-bound procedure for multiple resource-constrained project scheduling problem. *Management science*, 38(12):1803–1818, 1992.

[44] A. Drexl and J. Gruenewald. Nonpreemptive multi-mode resource-constrained project scheduling. *IIE transactions*, 25(5):74–81, 1993.

[45] S. E. Elmaghraby and J. Kamburowski. The Analysis of Activity Networks Under Generalized Precedence Relations (GPRs). *Management Science*, 38(9):1245–1263, September 1992.

[46] S.E. Elmaghraby. *Activity networks: Project planning and control by network models.* Wiley New York, 1977.

[47] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1):23–47, 2003.

[48] F. Focacci, P. Laborie, and W. Nuijten. Solving scheduling problems with setup times and alternative resources. In *Proc. of ICAPS*, 2000.

[49] F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In *Proc. of CP*, pages 189–203. Springer, 1999.

[50] D. Godard, P. Laborie, and W. Nuijten. Randomized Large Neighborhood Search for Cumulative Scheduling. In *Proc. of ICAPS*, pages 81–89, 2005.

[51] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.

[52] I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26(11):1533–1552, 2002.

[53] S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1—-14, 2010.

[54] S. Hartmann and A. Drexl. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32(4):283–297, 1998.

[55] R. Heilmann. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. *European Journal of Operational Research*, 144(2):348–365, January 2003.

[56] T. J. Hindelang and J. F. Muth. A dynamic programming algorithm for decision CPM networks. *Operations Research*, 27(2):225–241, 1979.

[57] J. N. Hooker. A Hybrid Method for Planning and Scheduling. *Constraints*, 10(4):385–401, 2005.

[58] J. N. Hooker. An Integrated Method for Planning and Scheduling to Minimize Tardiness. *Constraints*, 11(2-3):139–157, 2006.

[59] J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[60] J.N. Hooker. Planning and scheduling to minimize tardiness. *Proc. of CP*, 3709:314–327, 2005.

[61] J.N. Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55(3):588, 2007.

[62] IBM. *IBM-ILOG Scheduler User Manual (CP1.5)*. 2010.

[63] G. Igelmund and F. J. Radermacher. Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, 13(1):29–48, January 1983.

[64] G. Igelmund and F. J. Radermacher. Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, 13(1):1–28, January 1983.

[65] S. A. ILOG. *ILOG Scheduler Reference Manual*. 1997.

[66] V. Jain and I. E. Grossmann. Algorithms for Hybrid MILP / CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing*, pages 258–276, 1999.

[67] V. Jain and I. E. Grossmann. Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing*, 13(4):258–276, 2001.

[68] B. Jarboui, N. Damak, P. Siarry, and A. Rebai. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308, January 2008.

[69] U. Junker. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In *Proc. of AAAI*, pages 167–172. AAAI Press / The MIT Press, 2004.

[70] J.E. Kelley and M. R. Walker. Critical-path planning and scheduling. In *Proc. of eastern joint IRE-AIEE-ACM conference*, pages 160–173, New York, USA, 1959. ACM.

[71] R. Klein and A. Scholl. Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research*, 112(2):322–346, 1999.

[72] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[73] R. Kolisch. PSPLIB - A project scheduling problem library. *European Journal of Operational Research*, 96(1):205–216, January 1997.

[74] R. Kolisch and A. Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29(11):987–999, November 1997.

[75] R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science*, 41(10):1693–1703, 1995.

[76] Y. K. Kwok. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *Parallel and Distributed Systems, IEEE*, 7(5):506—-521, 2002.

[77] Y. K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys (CSUR)*, 31(4):406–471, 1999.

[78] P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, 143(2):151–188, February 2003.

[79] P. Laborie. Complete MCS-Based Search: Application to Resource Constrained Project Scheduling. In *Proc. of IJCAI*, pages 181–186. Professional Book Center, 2005.

[80] P. Laborie. IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems. In *Proc. of CPAIOR*, pages 148–162, 2009.

[81] P. Laborie and M. Ghallab. Planning with sharable resource constraints. In *Proc.of IJCAI*, pages 1643–1649. Citeseer, 1995.

[82] P. Laborie and D. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. In *Proceedings MISTA-07, Paris*, 2007.

[83] P. Laborie and J. Rogerie. Reasoning with conditional time-intervals. In *Proc. of FLAIRS*, pages 555–560, 2008.

[84] P. Laborie, J. Rogerie, P. Shaw, and P. Vilìm. Reasoning with Conditional Time-Intervals Part II: An Algebraical Model for Resources. In *Proc. of FLAIRS*, pages 201–206, 2009.

[85] P. Laborie, J. Rogerie, P. Shaw, P. Vilìm, and F. Wagner. ILOG CP Optimizer: Detailed Scheduling Model and OPL Formulation. Technical report, 2008.

[86] C. Le Pape. Using a constraint-based scheduling library to solve a specific scheduling problem. In *Proc. of AAAI-SIGMAN*, 1994.

[87] C. Le Pape, P. Couronné, D. Vergamini, and V. Gosselin. Time-versus-capacity compromises in project scheduling. *AISB QUARTERLY*, page 19, 1995.

[88] M. Lombardi and M. Milano. A Precedence Constraint Posting Approach for the RCPSP with Time Lags and Variable Durations. In *Proc. of CP*, pages 569–583. Springer, 2009.

[89] M. Lombardi and M. Milano. Allocation and scheduling of Conditional Task Graphs. *Artificial Intelligence*, In Press,, 2010.

[90] M. Lombardi, M. Milano, M. Ruggiero, and L. Benini. Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip. *Journal of Scheduling*, 13(4):315–345, June 2010.

[91] A. Lova, P. Tormos, and F. Barber. Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial*, 30(30):69–86, December 2006.

[92] A. Lova, P. Tormos, M. Cervantes, and F. Barber. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117(2):302–316, February 2009.

[93] S. Manolache, P. Eles, and Z. Peng. Task mapping and priority assignment for soft real-time applications under deadline miss ratio constraints. *ACM Transactions on Embedded Computing Systems*, 7(2):1–35, February 2008.

[94] M. Mika, G. Waligora, and J. Weglarz. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research*, 187(3):1238–1250, June 2008.

[95] M. Mika, G. Waligora, and J. Wglarz. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. *European Journal of Operational Research*, 164(3):639–668, August 2005.

[96] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, pages 714–729, 1998.

[97] M. D. Moffitt, B. Peintner, and M. E. Pollack. Augmenting disjunctive temporal problems with finite-domain constraints. In *Proc. of AAAI*, pages 1–6, 2005.

[98] K. Neumann and C. Schwindt. Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56(3):513–533, 2003.

[99] W. Nuijten. *Time and resource constrained scheduling : a constraint satisfaction approach*. PhD thesis, Technische Universiteit Eindhoven, 1994.

[100] W. Nuijten, T. Bousonville, F. Focacci, D. Godard, and C. Le Pape. Towards an industrial manufacturing scheduling problem and test bed. In *Proc. of PMS*, 2004.

[101] W. P. M. Nuijten, E. H. L. Aarts, D.A.A. van Arp Talmaan Kip, and K.M. van Hee. Randomized constraint satisfaction for job shop scheduling. In *Proc. of IJCAI*, pages 251–262, Chambery, France, 1993.

[102] JH Patterson, R. Slowinski, FB Talbot, and J. Weglarz. An algorithm for a general class of precedence and resource constrained scheduling problems. *Advances in project scheduling*, pages 3–28, 1989.

[103] P. G. Paulin and J. P. Knight. Force-directed scheduling for the behavioral synthesis of ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(6):661–679, June 1989.

[104] N. Policella, A. Cesta, A. Oddi, and S. F. Smith. From precedence constraint posting to partial order schedules: A CSP approach to Robust Scheduling. *AI Communications*, 20(3):163–180, 2007.

[105] F. J. Radermacher. Scheduling of project networks. *Annals of Operations Research*, 4(1):227–252, 1985.

[106] M. Ranjbar, B. De Reyck, and F. Kianfar. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, 193(1):35–48, February 2009.

[107] M. Ruggiero, P. Gioia, G. Alessio, L. Benini, M. Michela, Davide Bertozzi, and Alexandru Andrei. A cooperative, accurate solving framework for optimal allocation, scheduling and frequency selection on energy-efficient mpsocs. In *Proc. of SoC*, pages 1–4. IEEE, 2007.

[108] M. Sabzehparvar and S. M. Seyed-Hosseini. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *The Journal of Supercomputing*, 44(3):257–273, November 2007.

[109] A. Schutt, T. Feydy, P. J. Stuckey, and M. Wallace. Why Cumulative Decomposition Is Not as Bad as It Sounds. In *Proc. of CP*, pages 746–761, 2009.

[110] C. Schwindt. *Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern.* Shaker, 1998.

[111] S. F. Smith and C. C. Cheng. Slack-based heuristics for constraint satisfaction scheduling. In *Proc.of AAAI*, pages 139–139. Wiley, 1993.

[112] A. Sprecher and A. Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107(2):431–450, 1998.

[113] A. Sprecher, S. Hartmann, and A. Drexl. An exact algorithm for project scheduling with multiple modes. *OR Spectrum*, 19(3):195–203, 1997.

[114] A. Sprecher and C. L. Hwang. *Resource-constrained project scheduling: exact methods for the multi-mode case.* Springer, 1994.

[115] A. Sprecher, R. Kolisch, and A. Drexl. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 80(1):94–102, 1995.

[116] J. P. Stinson, E. W. Davis, and B. M. Khumawala. Multiple Resource-Constrained Scheduling Using Branch and Bound. *IIE Transactions*, 10(3):252–259, 1978.

[117] F. B. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, 28(10):1197–1210, 1982.

[118] I. Tsamardinos, T. Vidal, and M. E. Pollack. CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning. *Constraints*, 8(4):365–388, 2003.

[119] Vicente Valls, Francisco Ballestín, and S Quintanilla. Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, 165(2):375–386, September 2005.

[120] V. Van Peteghem and M. Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418, March 2010.

[121] P. Vilìm. Max energy filtering algorithm for discrete cumulative resources. In *Proc. of CPAIOR*, page 294. Springer, 2009.

[122] P. Vilim, R. Barták, and O. Cepek. Extension of (n log n) Filtering Algorithms for the Unary Resource Constraint to Optional Activities. *Constraints*, 10(4):403–425, 2005.

[123] S. Vo and A. Witt. Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: a real-world application. *International Journal of Production Economics*, 105(2), 2007.

[124] F. Wang and Y. Xie. Embedded Multi-Processor System-on-chip (MP-SoC) design considering process variations. In *Proc. of IEEE IPDPS*, pages 1–5. Ieee, April 2008.

[125] P. Wuliang and W. Chengen. A multi-mode resource-constrained discrete timecost tradeoff problem and its genetic algorithm based solution. *International Journal of Project Management*, 27(6):600–609, August 2009.

[126] J. C. Zapata, B. M. Hodge, and G. V. Reklaitis. The Multimode Resource Constrained Multiproject Scheduling Problem : Alternative Formulations. *AIChE Journal*, 54(8), 2008.

[127] G. Zhu, J. F. Bard, and G. Yu. A Branch-and-Cut Procedure for the Multimode Resource-Constrained Project-Scheduling Problem. *INFORMS Journal on Computing*, 18(3):377–390, January 2006.