Benders decomposition of the resource constrained project scheduling problem

SE Terblanche† and JH van Vuuren‡

†Centre for Business Mathematics and Informatics, North-West University, South Africa,
‡Department of Industrial Engineering, Stellenbosch University, South Africa

Abstract

Problem instances found in the literature that are used in computational studies of the resource constrained project scheduling problem, typically include only a few resources. In some practical applications, however, the number of resources may be significantly higher. In this paper, problem instances with a large number of resources are considered and a Benders decomposition approach is followed to deal with the additional computational effort required. Details of a separation routine for deriving the feasibility cuts and a heuristic for computing primal feasible solutions within a branch-and-cut framework, are provided. Randomly generated problem instances are introduced, which are based on existing problem instances from the literature. The empirical results reported in this paper demonstrate the scalability of the Benders decomposition approach when considering problem instances with a large resource set.

1 Introduction

Finding solutions to the resource constrained project scheduling problem (RCPSP), whether approximate or optimal, requires an explicit mathematical model. The use of mixed integer linear programming (MILP) as a modelling approach is well suited for the formulation of the RCPSP due to the logical decision-making nature of the problem. In the literature, three main classes of RCPSP formulations can be found, namely time-indexed formulations, resource flow-based formulations, and event-based formulations.

Time-indexed formulations are based on the discretisation of time. Binary variables, indexed by both an activity and a time period, are used to indicate at which time period each activity will start ([23], [21]). An obvious drawback of this approach is that an increase in the total schedule duration will result in a significant increase in the number of variables. Event-based formulations rely on the fact that the end time of an activity, or a set of activities, coincides with the start time of another activity, or set of activities. This point in time is called an event. In [3] and [16], continuous variables are introduced to keep track of the time of each event and binary variables are used to indicate at which event each activity will start and end, respectively. In a resource flow formulation, continuous variables are defined that represent the flow of a resource from one activity to the next [5]. In [4], the start time of an activity is modelled as a continuous variable, while binary decision variables are required to fix the ordering of the activities.

The computational results provided in the literature that involve MILP modelling approaches are all based on problem instances containing only a small set of resources. For instance, the project scheduling problem library (PSPLIB) [1] is a repository of RCPSP problem instances that has been
related extensively over the years. The PSPLIB comprises the data sets J30, J60, J90 and J120, which are sets of RCPSP instances each containing 30, 60, 90 and 120 activities, respectively. Each data set consists of 480 different problem instances, except for the J120 data set which contains 600 problem instances. Each of these problem instances include, however, a set of only four resources. There are practical applications for which the problem instances may have a much larger resource set. As an example, consider the application of RCPSP approaches for solving underground mine scheduling problem instances. These problem instances may have, in addition to a large number of activities, a large number of resources that need to be considered in order to allow the generation of feasible solutions [27].

The model proposed in this paper for solving the resource constrained project scheduling problem with a large number of resource requirements, is based on the resource flow formulation. The attractiveness of this formulation stems from the ability to apply a Benders decomposition approach and that the resulting Benders sub-problem is a special instance of the well known multi-commodity network flow problem. For the purpose of facilitating the maximisation of NPV, which is a non-linear function of the activity start times, a piece-wise linear approximation approach is followed. The computational results presented in this paper demonstrate the scalability of the Benders decomposition approach when considering RCPSP problem instances with a large resource set.

A literature review of related work is presented in the section below, followed by Section 3 which provides a formulation of the resource constrained project scheduling problem. Details of preprocessing approaches for the purpose of reducing the number of variables and constraints in the original resource flow formulation, are provided in Section 4. Section 5 provides details of the proposed Benders decomposition approach, which include the separation of feasibility cuts and the generation of primal feasible solutions. Finally, computational results are presented in Section 7, followed by a summary and conclusion.

2 Related work

Due to the large number of practical applications of the RCPSP, the literature available on the topic is vast. Many variants of the RCPSP have been developed over the years as a result of having to solve scheduling problems with application-specific requirements. The surveys by [11] and [14] are good sources to help navigate through the many variants and extensions to the basic RCPSP.

Algorithmic approaches towards solving the RCPSP are in general either exact or heuristic. The philosophy adopted in this paper is that, although proven optimality may be seen as a luxury in practice, the continuous efforts in devising exact algorithms are important in terms of understanding problem-specific properties that may lead to the development of improved approaches, whether exact or heuristic. Furthermore, from a practical point of view, the solvers used for solving general MILP problems provide solution bounds when prematurely terminated, which serve as a quality certificate for the incumbent feasible solution. The literature discussion to follow is, therefore, limited to exact approaches.

Most of the exact methods reported for the RCPSP are based on either the general branch-and-bound method, proposed by [18], for solving MILP problems, or on implicit enumeration methods incorporating customised branch-and-bound schemes ([7], [9], [17], [21]). The early popularity of customised branch-and-bound schemes over the more general MILP branch-and-bound approach was a result of the inefficiency of the latter [22]. Technological advances and the growing maturity of mathematical programming methodologies have renewed efforts of solving the RCPSP as a MILP problem [16]. A clear advantage over customised branch-and-bound schemes when solving the RCPSP as a MILP problem is that other “business-related” side constraints may easily be incorporated into the model. For instance, binary programming formulations have been proposed for maximising NPV while taking constraints on capital expenditure [10] and material use [26] into account.
More recent advances within an exact framework include the use of constraint programming approaches ([12], [24], [25]) which resulted in solving many of the open PSPLIB instances [1]. A custom branch-and-bound scheme based on constraint propagation was proposed by [19] which makes use of the resource flow formulation of [5].

3 A resource flow-based formulation

The earliest MILP formulation of the flow-based RCPSP is by Artigues et al. [4], in which a polynomial insertion algorithm is proposed for solving the RCPSP. This formulation is, however, driven by an algorithmic approach and is not formulated for the purpose of solving it by means of a MILP solver. The work by [16] is the first to provide numerical results for a resource flow RCPSP formulation solved using an off-the-shelf commercial MILP solver.

Let \( \mathcal{N} \) denote the index set of all activities and let \( d_i \) be the duration of an activity \( i \in \mathcal{N} \). The order in which the activities have to be scheduled is specified by means of a directed acyclic graph \( H(\mathcal{N}, \mathcal{Z}) \), with vertex set \( \mathcal{N} \) and arc set \( \mathcal{Z} \). Each arc \((i, j) \in \mathcal{Z}\) corresponds to the precedence relation dictating that activity \( j \in \mathcal{N} \) should be preceded by activity \( i \in \mathcal{N} \). The execution of an activity implies that one or more resources will be consumed. For this purpose, \( \mathcal{R} \) is defined as the index set of all resources and \( v_{ir} \) as the quantity of resource \( r \in \mathcal{R} \) being consumed by activity \( i \in \mathcal{N} \), per time period. The availability of resources is in most cases restricted and, therefore, \( U_r \) is defined as an upper bound on the consumption of resource \( r \in \mathcal{R} \), per time period.

A source and a sink activity are introduced in order to facilitate the resource flow formulation. If the index set of activities is defined as \( \mathcal{N} = \{0, 1, \ldots, N\} \), then the index 0 is used to denote the source activity and the index \( N \) is used to denote the sink activity. For all the activities \( j \in \mathcal{N} \setminus \{0\} \) that do not have a predecessor, according the precedence graph \( H(\mathcal{N}, \mathcal{Z}) \), the arcs \((0, j)\) are added to the set \( \mathcal{Z} \). For all the activities \( i \in \mathcal{N} \setminus \{N\} \) that do not have a successor, according the precedence graph \( H(\mathcal{N}, \mathcal{Z}) \), the arcs \((i, N)\) are added to the set \( \mathcal{Z} \).

In order to represent the flow of resources, the complete graph \( G(\mathcal{N}, \mathcal{A}) \) is introduced, with the set of arcs \( \mathcal{A} \) representing the flow of resources among the nodes in \( \mathcal{N} \). Let \( \mathcal{A}(i) \subset \mathcal{A} \) be the set of arcs adjacent to the node \( i \in \mathcal{N} \). The notation \((i, j) \in \mathcal{A}(i)\) denotes an arc for which node \( i \) is the source and the notation \((i, j) \in \mathcal{A}(j)\) denotes an arc for which node \( j \) is the target. The resource requirements of the source and sink activities are set equal to the availability of the resources, i.e. \( v_{0r} = v_{Nr} = U_r \), for all \( r \in \mathcal{R} \).

The primary decision variables are the starting times \( s_i \geq 0 \) for each of the activities \( i \in \mathcal{N} \). The resource flow variables \( f_{rij} \geq 0 \) are introduced to denote the flow of a resource \( r \in \mathcal{R} \) from activity \( i \in \mathcal{N} \) to \( j \in \mathcal{N} \). The variables \( z_{ij} \in \{0, 1\} \), called the linear ordering variables, are used to indicate the ordering of activities based on the flow of resources. That is, if \( z_{ij} = 1 \), then activity \( j \) is allowed to start after completion of activity \( i \), since resources will then be allowed to be transferred from activity \( i \) to \( j \).

For the purpose of maximising NPV, a future cash-flow of \( c_i \) is associated with each activity \( i \in \mathcal{N} \). Additional variables and constraints are introduced to facilitate the formulation of the following non-linear function that maximises NPV based on a discount factor \( \alpha \):

\[
\text{maximise } \sum_{i \in \mathcal{N}} f_i(s_i),
\]

where

\[
f_i(s_i) = c_i e^{-\alpha s_i}.
\]

A piece-wise linear approximation of the objective function, which is based on a SOS type 2 formulation ([8, 20]), is suggested to deal with each non-linear function \( f_i(s_i) \). Let the points
\((s_{iv}, f_{iv}), v \in V = \{0, 1, 2, \ldots, V - 1\}\) be the vertices of the piece-wise linear approximation of the function \(f_i(s_i)\). The decision variable \(y_i \in \mathbb{R}\) is introduced to capture the approximate value of \(f_i(s_i)\) and the auxiliary variables \(\lambda_{iv} \geq 0, v \in V\) and \(\ell_{iv} \in \{0, 1\}\), with \(v \in V \setminus \{0\}\) are introduced in order to facilitate the SOS type 2 formulation.

The objective of the resource flow-based RCSP (RF) is to

\[
\text{maximise } \sum_{i \in \mathcal{N}} y_i, \quad (3)
\]

subject to the constraints

\[
z_{ij} = 1, \quad (i, j) \in \mathcal{Z},
\]

\[
s_j - s_i - (d_i + M)z_{ij} \geq -M, \quad (i, j) \in \mathcal{A},
\]

\[
s_i - \sum_{v \in V} \lambda_{iv} s_{iv} = 0, \quad i \in \mathcal{N},
\]

\[
y_i - \sum_{v \in V} \lambda_{iv} y_{iv} = 0, \quad i \in \mathcal{N},
\]

\[
\sum_{v \in V} \lambda_{iv} = 1, \quad i \in \mathcal{N},
\]

\[
\lambda_{iv} - \ell_{iv} \leq 0, \quad i \in \mathcal{N},
\]

\[
\lambda_{iv} - \ell_{iv} - \ell_{i(v+1)} \leq 0, \quad i \in \mathcal{N}, \ v \in V \setminus \{0, V - 1\},
\]

\[
\lambda_{i(V-1)} - \ell_{i(V-1)} \leq 0, \quad i \in \mathcal{N},
\]

\[
\sum_{(i, j) \in \mathcal{A}(i)} f_{ijr} = v_{ir}, \quad i \in \mathcal{N}, \ r \in \mathcal{R},
\]

\[
\sum_{(i, j) \in \mathcal{A}(j)} f_{ijr} = v_{jr}, \quad j \in \mathcal{N}, \ r \in \mathcal{R},
\]

\[
f_{ijr} - \min\{v_{ir}, v_{jr}\} z_{ij} \leq 0, \quad (i, j) \in \mathcal{A}, \ r \in \mathcal{R}. \quad (14)
\]

The objective function (3) expresses the NPV as the sum of the linear piece-wise approximations \(y_i \approx f_i(s_i)\), for all activities \(i \in \mathcal{N}\), while constraint set (4) is required to ensure feasibility in terms of activity precedence. Constraint set (5) is collectively called the linear ordering constraints which determine the linear ordering variables \(z_{ij}\) based on the starting time \(s_j\) of activity \(j\) and the completion time of its predecessor \(i\), given by \(s_i + d_i\). A reasonable choice for the large number \(M\) in (5) would be the latest possible finishing time of the schedule, estimated as \(M = \sum_{i \in \mathcal{N}} d_i\).

Constraint sets (6) and (7) express \(s_i\) and \(y_i\) as convex combinations of the piece-wise linearisation knots of \(f_i(s_i)\), for all \(i \in \mathcal{N}\). Convexity conditions are maintained by (8), while constraint sets (9), (10) and (11) are responsible for enabling the convexity variable \(\lambda_{iv}\) to take on an appropriate value based on the selection of a specific line segment \(\ell_{iv}\).

The resource flow requirements are imposed by constraint sets (12) and (13), stating that all the flow of resources into an activity (12) and all the flow of resources out of an activity (13) should match the resource requirement \(v_{ir}\) of an activity \(i\). According to constraint set (14), the flow of resources from activity \(i\) to \(j\) is permitted only if activity \(j\) is scheduled to start after the completion of activity \(i\), that is when \(z_{ij} = 1\).

4 Problem reformulation and preprocessing

As a first attempt to improve computational efficiency when solving the RF formulation with a large resource set \(\mathcal{R}\), a variable and constraint reduction approach are suggested. Variable reduction is
achieved by reducing the number of arcs in the resource flow graph $G(N, A)$ and constraint reduction
is achieved by aggregating some of the constraints in the RF formulation, without removing feasible
solutions from the problem space. Although the resulting reformulation may possibly be weaker
than the original, it may be computationally more efficient since it employs fewer variables and
constraints to describe the same feasible region. The anticipated advantage is that there is less of
a burden on the LP solver when solving the sub-problems of the branch-and-bound tree.

4.1 Graph reduction (GR)

The approach proposed in this section to reduce the resource flow graph is inspired by the ob-
servation that resource flows are consistent with precedence relationships. More specifically, it is
evident from the linear ordering constraints that for an activity $i$, which is a predecessor of $j$ (i.e.
$s_i + d_i \leq s_j$), resources are not allowed to flow from $j$ to $i$. Applying this recursively, no flows
are allowed from $j$ to any of its ancestors, as defined by the precedence graph $G(N, Z)$. Based on
this observation, some of the variables and constraints in the original problem formulations may
be ignored by considering a subgraph $G'(N, A')$, with $A' \subset A$ constructed in such a manner that
there are no arcs from a node $i$ to any of its ancestors.

The set of predecessor activities of activity $i \in N$ is defined as $P(i) \subseteq N$. These predecessor
activities are the immediate predecessors of activity $i$. For each of the predecessors $j \in P(i)$,
a predecessor list $P(j)$ exists. Continuing in a recursive manner, all of the activities along each
possible path from node $i$ to a root node of the precedence graph is obtained. This set of activities
is denoted by $P^+(i) \subseteq N$ and denotes the set of ancestors for activity $i$.

The set of arcs of the reduced graph $G'(N, A')$ is defined as $A' = \{(i, j) : j \notin P^+(i)\}$. By
replacing the set $A$ in the RF formulation (3)–(14) with the set $A'$, graph reduction is applied
resulting in fewer variables and constraints.

4.2 Constraint aggregation (CA)

Consider constraint set (14) in the RF formulation. These constraints are responsible for allowing
the flow of resources whenever the linear ordering variable $z_{ij}$ has been set to one. There are in total
$|A| \times |R|$ of these constraints. An increase in the number of resources in the problem instance being
solved may, therefore, have a deteriorating effect on computing times. Since the linear ordering
variables $z_{ij}$ are not indexed by a resource it is, however, possible to rewrite these constraints in
the following aggregated form

$$\sum_{r \in R} f_{ijr} - \left( \sum_{r \in F} U_r \right) z_{ij} \leq 0, \ (i, j) \in A. \quad (15)$$

Although the constraints (15) are based on a big-M formulation, which is considered to be an
undesirable property, they do, however, comprises fewer constraints since the individual resource
flows $f_{ijr}$ are aggregated over all the resources $r \in R$. Some benefit is expected in terms of
computing times by replacing (14) with (15), especially in the case where a large number of resources
is considered.

5 Benders decomposition

The suggested approach for solving the resource flow RCPSP within a branch-and-cut framework
is to make use of Benders decomposition [6]. The resource flow variables and their associated con-
straints are projected out of the original MILP formulation and a separation algorithm is presented
below which is responsible for generating cuts so as to ensure feasibility.
5.1 The RF master problem (RFM)

Consider the RF formulation (3)–(14). The variables in the RF formulation are the start time variables \( s \in \mathbb{R}_{+}^{|N|} \), the linear ordering variables \( z \in \{0, 1\}^{|A|} \), the resource flow variables \( f \in \mathbb{R}_{+}^{|A| \times |R|} \) and the variables required to facilitate the maximisation of NPV, that is \( y \in \mathbb{R}_{+}^{|N|} \), \( \lambda \in [0, 1]^{|N| \times |V|} \) and \( \ell \in \{0, 1\}^{(|V|-1)} \). For illustration purposes, let \( x \) denote the adjoint vectors \((s, z, y, \lambda, \ell)\) and let \( X \) denote the polyhedron induced by the constraints (4)–(11). Let \( A \) and \( B \) be appropriate coefficient matrices associated with \( x \) and \( f \), respectively, which correspond to the coefficients in the constraints of the RF formulation. Furthermore, let \( b \) correspond to the right-hand side of the same constraints and let \( c \) and \( d \) be the cost vectors associated with \( x \) and \( f \), respectively. It is evident from the RF objective function (3) that all entries of the cost vector \( d \) are zero since there are no costs associated with the flow variables \( f \) in the objective function.

The objective of the RF, in matrix form, is to

\[
\text{maximise } c^T x + d^T f, \quad (16)
\]
\[
s.t. \quad Ax + Bf \leq b, \quad (17)
\]
\[
x \in X. \quad (18)
\]

Since no flow variables are present in the polyhedron \( X \), the above problem can be decomposed into a master problem

\[
\text{min } c^T x, \quad (19)
\]
\[
x \in X \quad (20)
\]

and a sub-problem

\[
\text{min } d^T f, \quad (21)
\]
\[
s.t. Bf \leq b - Ax^*, \quad (22)
\]

where \( x^* \) is the solution obtained by solving the master problem.

The master problem (19)–(20) is an MILP problem and may be solved using the branch-and-bound method. The sub-problem (21)–(22) is an LP problem, given a fixed \( x^* \). The outcome of solving the LP sub-problem is that either an optimal solution \( f^* \) is obtained, or the problem is infeasible as a result of \( x^* \). In the case where the sub-problem is infeasible, a cut \((b - Ax^*)^T w \geq 0\) can be derived where \( w \) is an extreme direction. By adding the feasibility cut to the master problem, the infeasible point \( x^* \) is “separated” from the feasible region of the master problem.

The objective of the resource flow master problem (RFM) is to

\[
\text{maximise } \sum_{i \in N} y_i, \quad (23)
\]

subject to the constraints

\[
\begin{align*}
z_{ij} &= 1, & (i, j) \in Z, \quad (24) \\
s_j - s_i - (d_i + M)z_{ij} &\geq -M, & (i, j) \in A, \quad (25) \\
s_i - \sum_{v \in V} \lambda_{iv} s_{iv} &= 0, & i \in N, \quad (26) \\
y_i - \sum_{v \in V} \lambda_{iv} y_{iv} &= 0, & i \in N, \quad (27) \\
\sum_{v \in V} \lambda_{iv} &= 1, & i \in N, \quad (28) \\
\lambda_{ii} - \ell_{ii} &\leq 0, & i \in N, \quad (29) \\
\lambda_{iv} - \ell_{iv} - \ell_{(v+1)} &\leq 0, & i \in N, \quad v \in V \setminus \{0, V - 1\}, \quad (30) \\
\lambda_{i(V-1)} - \ell_{i(V-1)} &\leq 0, & i \in N. \quad (31)
\end{align*}
\]
5.2 The resource flow separation problem (RFSEP)

A separation problem is suggested based on sub-problem (21)–(22), which produces a Benders cut 
\((b - Ax^*)^T w \geq 0\) if infeasibility is detected in the sub-problem. Also note that the sub-problem 
(21)–(22) is separable with respect to the set of resources \(\mathcal{R}\). A slack variable \(\alpha_r \geq 0\) is introduced 
and for each \(r \in \mathcal{R}\), the objective of the resource flow separation problem (RFSEP), is to 
\[
\min \alpha_r,
\]
subject to the constraints
\[
\begin{align*}
\sum_{(i,j) \in A(i)} f_{ijr} & = v_{ir}, \quad i \in \mathcal{N}, \quad (33) \\
\sum_{(i,j) \in A(j)} f_{ijr} & = v_{jr}, \quad j \in \mathcal{N}, \quad (34) \\
f_{ijr} + \alpha_r & \leq \min\{v_{ir}, v_{jr}\} z^*_{ij}, \quad (i,j) \in A. \quad (35)
\end{align*}
\]
If \(\alpha_r > 0\) for any \(r \in \mathcal{R}\), then the current vector \(z^*\) is infeasible.

**Theorem 1.** For a given \(r \in \mathcal{R}\) and a vector \(z^* \in \{0,1\}^{|A|}\), the cut 
\[
\sum_{(i,j) \in A} \min\{v_{ir}, v_{jr}\} z^*_ij \mu_{ij} \leq -\sum_{i \in \mathcal{N}} v_{ir} (\pi^1_i + \pi^2_i), 
\]
where \(\pi^1 \in \mathbb{R}^{|\mathcal{N}|}\), \(\pi^2 \in \mathbb{R}^{|\mathcal{N}|}\) and \(\mu \in \mathbb{R}^{\leq 0}\) are the dual vectors associated with (33), (34) and (35) 
respectively, separates the infeasible point \(z^*\) if \(\alpha_r > 0\).

**Proof.** The dual objective function of the RFSEP for a given \(r \in \mathcal{R}\), is to 
\[
\max \left\{ \sum_{i \in \mathcal{N}} (v_{ir} \pi^1_i + v_{ir} \pi^2_i) + \sum_{(i,j) \in A} \min\{v_{ir}, v_{jr}\} z^*_ij \mu_{ij} \right\}.
\]
In the case of infeasibility, \(\alpha > 0\) is an optimal solution to the RFSEP and, therefore,
\[
\sum_{i \in \mathcal{N}} v_{ir} (\pi^1_i + \pi^2_i) + \sum_{(i,j) \in A} \min\{v_{ir}, v_{jr}\} z^*_ij \mu_{ij} > 0.
\]
In order to separate the infeasible point \(z^*\), the violated cut 
\[
\sum_{(i,j) \in A} \min\{v_{ir}, v_{jr}\} z^*_ij \mu_{ij} \leq -\sum_{i \in \mathcal{N}} v_{ir} (\pi^1_i + \pi^2_i)
\]
is added to the master problem. \qed

5.3 The right-shift primal heuristic (RSPH)

The Benders decomposition solution approach outlined above involves the resource flow master 
problem RFM and the separation routine RFSEP. The application of Benders decomposition within 
a branch-and-cut framework allows for heuristics to be applied for the purpose of generating primal 
solutions to the Benders master problem. The proposed right-shift primal heuristic (RSPH) relies 
on current information obtained from the LP relaxation of the Benders master problem.
Algorithm 1 The right-shift primal heuristic (RSPH)

\[ \mathcal{N}^B = \emptyset, \]
\[ \mathcal{N}^P = \{0\}. \]
\[ s_0 = 0. \]
\[ \text{for } (0, j) \in \mathcal{Z} \text{ do} \]
\[ \mathcal{N}^B = \mathcal{N}^B \cup \{j\}. \]
\[ \text{end for} \]
\[ t = 0 \]
\[ \text{while } \mathcal{N}^B \not= \emptyset \text{ do} \]
\[ \text{period\_feasible} = \text{false}. \]
\[ \text{for } i \in \mathcal{N}^B \text{ do} \]
\[ \text{if } v_{ir} \leq U_r - \sum_{i' \in \mathcal{N}^P} v_{i'r} \forall r \in \mathcal{R} \text{ then} \]
\[ \text{period\_feasible} = \text{true}. \]
\[ \text{end if} \]
\[ \text{end for} \]
\[ \text{if } \text{period\_feasible} = \text{true} \text{ then} \]
\[ \text{Sort } \mathcal{N}^B \text{ in ascending order according to } s^*_{i,LP}. \]
\[ \text{for } i \in \mathcal{N}^B \text{ do} \]
\[ \text{if } v_{ir} \leq U_r - \sum_{i' \in \mathcal{N}^P} v_{i'r} \forall r \in \mathcal{R} \text{ then} \]
\[ \text{if } s^*_i + d_j \leq t \forall (i, j) \in \mathcal{Z} \text{ then} \]
\[ s^*_i = t \]
\[ \mathcal{N}^P = \mathcal{N}^P \cup \{i\} \]
\[ \mathcal{N}^B = \mathcal{N}^B \setminus \{i\} \]
\[ \text{for } (i, j) \in \mathcal{Z} \text{ do} \]
\[ \mathcal{N}^B = \mathcal{N}^B \cup \{j\}. \]
\[ \text{end for} \]
\[ \text{end if} \]
\[ \text{end for} \]
\[ \text{end if} \]
\[ t = t + 1 \]
\[ \text{end while} \]
\[ \text{for } (i, j) \in \mathcal{A} \text{ do} \]
\[ \text{if } s^*_j \geq s^*_i + d_i \text{ then} \]
\[ z^*_ij = 1 \]
\[ \text{end if} \]
\[ \text{end for} \]

Solve RFM(z*).

Consider the RFM formulation (23)–(31). During the solution of the RFM at a node of the branch-and-bound tree, solution values are obtained for the start time variables s, which may not necessarily constitute a feasible solution. That is, some of the linear ordering variables z may be fractional. The motivation for the suggested RSPH is that the LP relaxation solutions of the start time variables, denoted by \( s^*_{i,LP} \), may give a good indication of an optimal solution, since they are directly influenced by the objective function. More specifically, the sequencing order of the activities implied by the start time solutions \( s^*_{i,LP} \) may possibly correspond to the same order as in an optimal solution. The approach of the RSPH is, therefore, to right-shift activities by maintaining the sequencing order implied by the start time solutions.

A set \( \mathcal{N}^B \) is introduced for activities that are eligible for execution in the current time period.
until the resource capacity for the current time period $t$ and a set $N^P$ is introduced for activities already processed and for which start time values $s^*_i$ have been determined. Once an activity $i$ has been selected for execution at a time period $t$, its start time is set to $s^*_i = t$, it is added to $N^P$ and it is removed from $N^B$. An activity $j$ may only be considered eligible to start during a time period $t$ if all of its predecessors $i$ have been completed prior to $t$. More specifically, $N^B = N^B \cup \{j\}$ only if $i \in N^P$ for all $(i,j) \in Z$, and only if $s_j \geq s_i + d_i$.

Selecting activities from the eligibility set $N^B$ for execution during time period $t$ is based on the activity ranking provided by the LP relaxation solution $s^*_{LP}$. That is, the activity with the smallest start time LP solution is selected first. Activities are selected from the eligibility set $N^B$ until the resource capacity for the current time period $t$ has been depleted.

It is important to note that the start time solutions $s^*$ generated by the RSPH heuristic are not the final solutions utilised by the branch-and-cut process. The solution vector $s^*$ is only used to derive the proposed sequencing order of a potential feasible solution. More specifically, the solution $z^*_j = 1$ is constructed for all resource flow arcs $(i,j) \in A$, provided that $s^*_j \geq s^*_i + d_i$. The solution vector $z^*$ is then used to fix the ordering variables $z$ in a copy of the RFM problem. The notation $RFM(z^*)$ is used to indicate the solution of the RFM problem with the ordering variables $z$ fixed to the values provided by the vector $z^*$. The solution obtained by solving $RFM(z^*)$ is used as a primal solution of the Benders master problem $RFM$ within the branch-and-cut framework. The complete algorithm for the RSPH is provided in pseudocode form in Algorithm 1.

## 6 Test data instances

Several data sets for the RCPSP and its variants are available in the academic research community for the purpose of testing model efficiency and algorithmic ideas. For instance, the project scheduling problem library (PSPLIB) [1] is a repository of RCPSP problem instances that has been referenced extensively over the years. The PSPLIB comprises the data sets J30, J60, J90 and J120, which are sets of RCPSP instances each containing 30, 60, 90 and 120 activities, respectively. Details on how these problem instances were created can be found in [15].

The empirical study for this paper involves newly created data sets which are based on the PSPLIB instances. For the purpose of testing the scalability of the newly proposed formulations and decomposition approach, additional data sets were created that include problem instances with a large number of resources. Creating problem instances with more resources required the generation of new resource upper limits $U_r$ and resource requirements $v_{ir}$, based on existing problem instances. For example, a new resource $r \in R$ was created from an existing resource $r \in R$, by letting $U_r = U_r(1 + u)$ with $u \sim U(-0.5, 0.5)$ and by letting $v_{ir} = v_{ir}(1 + n_i)$ with $n_i \sim N(0,1)$.

The newly created data sets with randomly generated resources are called the J30Rx, J60Rx, and J90Rx, with the place holder $x$ indicating the resource multiplier applied. For instance, the problem instances of the J30 data set have a total of 4 resources each, whereas the J30R10 data set have a total of 40 resources per problem instance. Multiplier values of 10 and 20 were used for testing purposes.

An important collective contribution by the research community has been the characterisation of problem instances according to various indicators. Some of the indicators may be used to distinguish between “easy” and “hard” instances and are briefly described below (see [2] for details):

**Order strength (OS)** is a measure of parallelism of the underlying precedence graph. That is, a problem instance for which $OS = 0$ indicates that all activities occur in parallel, whereas $OS = 1$ indicates that all activities are ordered in series. The hardness of problem instances increases with a decrease in the value of $OS$.

**Network complexity (NC)** is the average number of incident arcs per node in the precedence graph. Higher levels of $NC$ are associated with harder problem instances.
Resource strength (RS) is a measure which combines resource requirements per activity and peak resource demand due to a precedence-feasible schedule based on the earliest start times of activities. Problem instances for RS close to zero are considered much harder than problem instances for which RS is close to one.

Disjunction ratio (DR) provides an indication of how many activities may be scheduled in parallel by taking resource requirements and precedence relations into account. Highly disjunctive problem instances are considered to be easier than cumulative instances that have a lower disjunction ratio.

<table>
<thead>
<tr>
<th>Data set</th>
<th>OS avg</th>
<th>OS σ</th>
<th>NC avg</th>
<th>NC σ</th>
<th>RS avg</th>
<th>RS σ</th>
<th>DR avg</th>
<th>DR σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>J30</td>
<td>0.52</td>
<td>0.09</td>
<td>1.81</td>
<td>0.26</td>
<td>0.62</td>
<td>0.29</td>
<td>0.56</td>
<td>0.11</td>
</tr>
<tr>
<td>J30R10</td>
<td>0.52</td>
<td>0.09</td>
<td>1.81</td>
<td>0.26</td>
<td>0.52</td>
<td>0.14</td>
<td>0.76</td>
<td>0.14</td>
</tr>
<tr>
<td>J30R20</td>
<td>0.52</td>
<td>0.09</td>
<td>1.81</td>
<td>0.26</td>
<td>0.17</td>
<td>0.1</td>
<td>0.89</td>
<td>0.12</td>
</tr>
<tr>
<td>J60</td>
<td>0.40</td>
<td>0.08</td>
<td>1.81</td>
<td>0.25</td>
<td>0.60</td>
<td>0.29</td>
<td>0.41</td>
<td>0.09</td>
</tr>
<tr>
<td>J60R10</td>
<td>0.40</td>
<td>0.08</td>
<td>1.81</td>
<td>0.25</td>
<td>0.28</td>
<td>0.19</td>
<td>0.63</td>
<td>0.15</td>
</tr>
<tr>
<td>J60R20</td>
<td>0.40</td>
<td>0.08</td>
<td>1.81</td>
<td>0.25</td>
<td>0.13</td>
<td>0.07</td>
<td>0.86</td>
<td>0.16</td>
</tr>
<tr>
<td>J90</td>
<td>0.34</td>
<td>0.08</td>
<td>1.80</td>
<td>0.25</td>
<td>0.60</td>
<td>0.29</td>
<td>0.34</td>
<td>0.08</td>
</tr>
<tr>
<td>J90R10</td>
<td>0.34</td>
<td>0.08</td>
<td>1.80</td>
<td>0.25</td>
<td>0.22</td>
<td>0.14</td>
<td>0.56</td>
<td>0.16</td>
</tr>
<tr>
<td>J90R20</td>
<td>0.34</td>
<td>0.08</td>
<td>1.80</td>
<td>0.25</td>
<td>0.12</td>
<td>0.07</td>
<td>0.84</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Table 1: Tractability indicators.

The tractability indicators for the newly generated problem instances, compared to the original instances, are provided in Table 1. The OS and NC values remained the same for the newly generated problem instances, since these indicators are not calculated in terms of resource requirements. The RS values of the newly generated problem instances are lower compared to the original problem instances, indicating that they may be “harder” than the original problem instances. The DR values, on the other hand, are higher compared to the original problem instances, indicating that they may be “easier” than the original problem instances. Although a verdict on the tractability of the newly created problem instances is unclear at this point, the computational results to follow clearly show, however, that an increase in computing times is observed with an increase in the size of the resource set. For some of the problem instances, an increase in the number of resources even prevent the computation of feasible solutions within the specified time limit. This is especially true for the J60 and J90 data sets.

7 Computational results

All of the empirical tests reported in this section were performed on an HP Compaq Elite 8300, with eight cores and 32GB of RAM. SuSE Linux was used as operating system and the IBM product CPLEX v12.6 [13] was used as MILP solver. The CPLEX Concert Technology library was employed within a C++ program for implementing the model. Call-back functions were implemented to allow the execution of the separation sub-problems and primal heuristic as part of the branch-and-cut.

The time limit for each computational run was determined as a function of the size of the problem instance being solved. More specifically, 10 seconds of computing time was allowed for each activity in the problem instance. As an example, for the J30 data set in which each problem instance comprises 30 activities, the run time was limited to 300 seconds.

Two performance indicators are used for the purpose of reporting results. The first indicator is the percentage of instances in the data set for which at least one feasible solution could be computed
in the allocated time. The average gap is also reported for all of the problem instances for which feasible solutions could be computed. The second indicator is the percentage of instances in the data set for which an optimal solution could be computed within the specified time limit. The average computing times recorded to obtain optimal solutions are also provided.

Results are presented for the plain RF formulation, the RF formulation when considering both constraint aggregation and graph reduction (RF+CA+GR), and also the Benders decomposition approach implemented within a branch-and-cut framework.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Model</th>
<th>Feasible Solutions</th>
<th>Optimal Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Instances (%)</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>J30</td>
<td>RF</td>
<td>99</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>99</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>99.4</td>
<td>0.9</td>
</tr>
<tr>
<td>J60</td>
<td>RF</td>
<td>99.6</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>100</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>99.8</td>
<td>1.5</td>
</tr>
<tr>
<td>J90</td>
<td>RF</td>
<td>99</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>99.8</td>
<td>7.6</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>100</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 2: Computational results obtained for original PSPLIB problem instances.

The results obtained for the standard PSPLIB data sets from literature are presented in Table 2. Although no real differences are observed in terms of the percentage of feasible solutions computed, the application of Benders decomposition resulted in an improvement in the average gap obtained within the specified time limit. For the J60 data set, for instance, the average gap obtained by applying Benders decomposition is 1.5% compared to the 3.5% and 3.6% obtained by both the standard RF formulation and the RF+CA+GR formulation, respectively. For the J90 data set, even better results were obtained. An average gap of 0.9% was achieved by means of Benders decomposition, compared to the 6.8% and 7.6% obtained by the standard RF formulation and the RF+CA+GR formulation, respectively. It should be noted, however, that no real benefit is realised when employing RF+CA+GR or Benders over the standard RF formulation when considering the percentage of optimal solutions computed. The percentage of J30 problem instances solved to optimality through the application of Benders decomposition is only 76.6%, compared to the 85.2% and the 84.4% of instances solved to optimality by applying the standard RF formulation and the RF+CA+GR formulation, respectively. A similar trend is observed for the J60 and the J90 data sets, showing that the percentage of problem instances solved to optimality is much lower for Benders decomposition than for the standard RF and RF+CA+GR formulations.

Table 3 shows the results obtained for the PSPLIB problem instances which were augmented with 10 times more the number of resources. The computational burden imposed by increasing the number of resources is reflected by the lower percentage of problem instance for which at least on feasible solution is computed. The only exception is for the cases where Benders decomposition are applied. For the J30R10, the J60R10 and the J90R10 data sets, the percentage feasible solutions computed by applying Benders decomposition are 98.3%, 99.4% and 100%, respectively. And even more impressive are the low gap percentages obtained. For the J60R10 problem instances an average gap of 9.7% was achieved by means of Benders decomposition, compared to the 42% and 34.4% obtained by the standard RF formulation and the RF+CA+GR formulation, respectively. The average percentage gap computed by applying Benders decomposition for the J90R10 data set is 7.1%, compared to the 26.2% and 28.4% computed by applying the standard RF formulation and the RF+CA+GR formulation, respectively.

Once again, no real benefit is realised when employing RF+CA+GR or Benders over the standard RF formulation when considering the percentage of optimal solutions computed. It should be
Table 3: Computational results for the PSPLIB problem instances augmented with additional resources (×10).

<table>
<thead>
<tr>
<th>Data set</th>
<th>Model</th>
<th>Feasible Solutions</th>
<th>Solved to optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Instances (%)</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>J30R10</td>
<td>RF</td>
<td>95.2</td>
<td>3.3</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>97.5</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>98.3</td>
<td>4.2</td>
</tr>
<tr>
<td>J60R10</td>
<td>RF</td>
<td>92.7</td>
<td>42.0</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>92.9</td>
<td>34.4</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>99.4</td>
<td>9.7</td>
</tr>
<tr>
<td>J90R10</td>
<td>RF</td>
<td>52.7</td>
<td>26.2</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>68.3</td>
<td>28.4</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>100</td>
<td>7.1</td>
</tr>
</tbody>
</table>

Table 4: Computational results for the PSPLIB problem instances augmented with additional resources (×20).

<table>
<thead>
<tr>
<th>Data set</th>
<th>Model</th>
<th>Feasible Solutions</th>
<th>Solved to optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Instances (%)</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>J30R20</td>
<td>RF</td>
<td>95</td>
<td>22.3</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>94.4</td>
<td>27.6</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>99.8</td>
<td>25.0</td>
</tr>
<tr>
<td>J60R20</td>
<td>RF</td>
<td>39.8</td>
<td>168.0</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>55.2</td>
<td>54.5</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>75.4</td>
<td>19.2</td>
</tr>
<tr>
<td>J90R20</td>
<td>RF</td>
<td>23.5</td>
<td>44.9</td>
</tr>
<tr>
<td></td>
<td>RF+CA+GR</td>
<td>26.9</td>
<td>40.1</td>
</tr>
<tr>
<td></td>
<td>Benders</td>
<td>78.3</td>
<td>14.2</td>
</tr>
</tbody>
</table>

noted, however, that with a larger number of resources, it is almost impossible to compute optimal solutions for the J90R10 problem instances with any of the approaches. The application of the standard RF formulation and the RF+CA+GR formulation resulted in only 1.9% and 1% of the problem instances being solved to optimality, respectively.

Table 4 shows the results obtained for the PSPLIB problem instances which were augmented with 20 times more the number of resources. The most notable result is the inability of the standard RF formulation, the RF+CA+GR formulation, and the Benders decomposition approach, to produce optimal solutions to any of the J60R20 and J90R20 problem instances. The real benefit of applying Benders decomposition is, however, demonstrated through the results which show that at least one feasible solution could be computed for 75% and 78.3% of the J90R10 and J90R20 problem instances, respectively. In addition, the corresponding average gap percentages are 19.2% for the J90R10 data set and 14.2% for the J90R20 data set. The percentage of feasible solutions of the J90R10 and J90R20 problem instances obtained through the application of the standard RF formulation is 39.8% and 23.5%, respectively, and the corresponding percentages for the RF+CA+GR formulation are 55.2% and 26.9%, respectively.

Figures 1 to 3 serve as a final summary of the computational results obtained. The graphs show the scalability of the proposed Benders decomposition approach with respect to the various data sets, when the number of resources for each problem instance is increased by a factor of one, ten and twenty, respectively. As an example, to clarify the horizontal axis labels used in the graphs, the results for the “×1” resource multiplier case in Figure 1 corresponds to the problem instances of the J30 data set, the results for the “×10” resource multiplier case corresponds to the problem instances of the J30R10 data set, and the results for the “×20” resource multiplier case
Figure 1: Algorithmic scalability for the J30(50)D data set.

(a) Percentage feasible solutions
(b) Integrality gap

Figure 2: Algorithmic scalability for the J60(50)D data set.

(a) Percentage feasible solutions
(b) Integrality gap

Figure 3: Algorithmic scalability for the J90(50)D data set.

(a) Percentage feasible solutions
(b) Integrality gap

corresponds to the problem instances of the J30R20 data set. Figure (a) shows the percentage of problem instances of the J30 data set for which feasible solutions could be computed by the plain RF problem formulation, the RF+CA+GR and Benders decomposition, respectively. The average integrality gap obtained for the J30 data set, when considering an increase in the number of resources, is reported by the graph in Figure (b).

The graphs in Figures 1 to 3 that report the percentage feasible solutions computed (all the graphs on the left) show that the highest percentage of feasible solutions was computed by applying Benders decomposition, even with an increase in the number of resources. By comparing these results to those obtained by the plain RF formulation, the scalability of the Benders approach is clearly demonstrated. This is especially true for the larger and more difficult problem instances like the J90 data set. Although the RF+CA+GR formulation appears to be an improvement over the RF formulation, it is still not as scalable as Benders decomposition.

A significant reduction in average integrality gaps were observed for most of the data sets by employing Benders decomposition. The scalability of the Benders decomposition approach,
measured in terms of average integrality gap with an increase in the number of resources, is clearly demonstrated by the results for the larger and more difficult problem instances of the J60 and J90 data sets. The performance of the plain RF formulation, on the other hand, degraded significantly. Although improvements are reported for the use of the variable and constraint reduction approaches, they are still dominated by the superior performance of Benders decomposition.

8 Summary and conclusion

The primary objective of this paper is to propose algorithmic approaches for improving computing times when solving RCPSP problem instances with a large number of resources. Constraint aggregation and graph reduction are presented for reducing the number of variables and constraints in the resource flow formulation. In addition, a Benders decomposition approach is suggested which involves a sub-problem for the separation of Benders feasibility cuts and a heuristic for the generation of primal feasible solutions.

Computational tests were based on randomly generated data sets with a large number of resources. The results obtained for the various data sets demonstrated the superior scalability of the Benders decomposition approach over adopting the plain resource flow formulation.

References


