Volume 33 (1), pp. 21–34
http://orion.journals.ac.za

ORiON
ISSN 0259–191X (print)
ISSN 2224–0004 (online)

A note on flow-based formulations for solving resource constrained scheduling problems

SE Terblanche* JH van Vuuren†

Received: 22 March 2016; Revised: 15 August 2016; Accepted: 9 February 2017

Abstract

The resource constrained scheduling problem involves the scheduling of a number of activities over time, where each activity consumes one or more resources per time period. For a feasible solution to exist, the total resource consumption per time period must not exceed the available resources. In addition, the order in which activities may be scheduled is determined by a precedence graph. In this paper, valid inequalities proposed for the resource flow-based formulation in previous studies are investigated to determine what effect they may have on computing times. It is shown empirically that improved computing times may be obtained if these valid inequalities are, in fact, omitted from the resource flow-based formulation. In addition, a heuristic is proposed for the generation of initial starting solutions and for estimating the extent of the scheduling horizon which, in turn, is required to calculate the latest starting times of activities. The computational results are based on well-known problem test instances as well as new randomly generated problem instances.

Key words: Scheduling, mixed integer linear programming, valid inequalities.

1 Introduction

A solution to the *resource constrained scheduling problem* (RCSP) prescribes starting times of activities such that the total resource usage by the activities per time period is within a pre-specified resource capacity. Furthermore, the activities have to be scheduled such that all of the precedence constraints are satisfied.

The use of *mixed integer linear programming* (MILP) as a modelling approach is well suited for the formulation of the RCSP due to the logical decision-making nature of the problem. Several different mathematical formulations may, however, exist for the same problem.

*Corresponding author: Centre for Business Mathematics and Informatics, North-West University, Potchefstroom, South Africa, email: fanie.terblanche@nwu.ac.za

†**(Fellow of the Operations Research Society of South Africa)**, Stellenbosch Unit for Operations Research in Engineering, Department of Industrial Engineering, Stellenbosch University, Stellenbosch, South Africa, email: vuuren@sun.ac.za

http://dx.doi.org/10.5784/33-1-555

These different formulations may be equivalent in terms of representing the feasible region and the objective function of the RCSP, but they may differ in the number of variables and constraints. This may, in turn, have an impact on the efficiency with which the underlying algorithm finds solutions to these models. In the literature, three main classes of RCSP formulations can be found, namely time-indexed formulations [10, 12], resource flow-based formulations [3, 4] and event-based formulations [9, 16].

In a resource flow formulation, the resource consumption of activities is modelled as a network flow problem. That is, continuous variables represent the flow of a resource from one activity to the next. 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. In this paper it is shown that valid inequalities applied to the resource flow formulation by Koné *et al.* [9] may actually have a detrimental effect on computing times. Empirical results provided below show that improved computing times are obtained if these valid inequalities are, in fact, excluded from the RCSP formulation. In addition, a *resource graph expansion* (RGE) heuristic is proposed for the purpose of estimating the scheduling horizon which, in turn, is required to calculate the latest starting times of activities. The resulting solution from the RGE heuristic may also be applied as a starting solution of the overall RCSP.

In the following section, a resource flow-based formulation of the RCSP is provided. The valid inequalities of interest are also identified. These inequalitites may be omitted from the formulation of the RCSP in order to speed up computing times. Details of the proposed RGE heuristic are presented in Section 3, and this is followed by a description of an iterative linear programming approach for calculating the earliest and latest allowable starting times of activities. Computational results are presented in Section 4 based on well-known RCSP problem instances and randomly generated instances. The paper closes in Section 5 with a brief summary and some ideas for follow-up future work.

2 A resource flow formulation of the RCSP

The earliest resource flow MILP formulation of the RCSP is due to Artigues *et al.* [3], who proposed a polynomial insertion algorithm for solving the RCSP. This formulation is driven by an algorithmic approach and is not formulated for the purpose of solving it with an MILP solver. Koné *et al.* [9] were the first to provide numerical results for a resource flow RCSP formulation solved using an off-the-shelf, commercial MILP solver.

In order to facilitate a formulation of the RCSP, the following notation is required. Let $\mathcal{R}$ denote the set of resources and let $\mathcal{A}$ denote the index set of all activities. Furthermore, let d_i be the duration of activity $i \in \mathcal{A}$, measured in days, and let v_{ri} be the quantity of resource $r \in \mathcal{R}$ being consumed by activity $i \in \mathcal{A}$ per day. Also, let E_i be the earliest start time and let L_i be the latest start time of activity $i \in \mathcal{A}$. The earliest and latest start times of an activity are functions of a so-called precedence graph (of which the nodes represent the various activities and each directed edge a precedence relationship) and the planning horizon. An approach toward calculating E_i and L_i is provided in Section 3. Moreover, let $\mathcal{P}(i) \subseteq \mathcal{A}$ denote the set of immediate predecessor activities of activity $i \in \mathcal{A}$ (that is, all incident predecessor activities according to the precedence graph). Finally, let $\mathcal{S}(i) \subseteq \mathcal{A}$ denote the set of immediate successor activities of activity $i \in \mathcal{A}$ (that is, all incident

successor activities according to the precedence graph), and let U_r be the upper limit on the quantity of resource $r \in \mathcal{R}$ that may be consumed per day.

In order to facilitate the formulation of resource flow constraints below, two artificial activities are introduced, both with a duration of zero. A source activity i^+ is introduced with $\mathcal{P}(i^+) = \emptyset$ and $\mathcal{S}(i^+) = \{i \in \mathcal{A} : \mathcal{P}(i) = \emptyset\}$, and a sink activity i^- is introduced with $\mathcal{S}(i^-) = \emptyset$ and $\mathcal{P}(i^-) = \{i \in \mathcal{A} : \mathcal{S}(i) = \emptyset\}$. Furthermore, for the source and sink activities, $v_{ri^+} = U_r$ and $v_{ri^-} = U_r$, respectively, for all $r \in \mathcal{R}$.

The primary decision variables are the starting times $s_i \geq 0$ for each of the activities $i \in \mathcal{A}$. In order to formulate the resource requirement constraints, resource flow variables $f_{rij} \geq 0$ are introduced to denote the flow of a resource $r \in \mathcal{R}$ from activity $i \in \mathcal{A}$ to $j \in \mathcal{A}$. Binary variables $z_{ij} \in \{0, 1\}$, called the linear ordering variables, are used to indicate the ordering of activities. That is, if $z_{ij} = 1$ it indicates that activity j is scheduled to start only after completion of activity i. Consequently, the linear ordering variables also indicate whether the transfer of a resource is permitted from activity $i \in \mathcal{A}$ to $j \in \mathcal{A}$.

The objective of the resource flow RCSP is to

$$\text{minimise} \quad s_{i^-} \tag{1}$$

subject to the constraints

$$z_{ij} = 1, \qquad j \in \mathcal{A},\ i \in \mathcal{P}(j), \tag{2}$$

$$s_j - s_i - (d_i + M)z_{ij} \geq -M, \qquad (i,j) \in \mathcal{A}^2,\ i \neq j, \tag{3}$$

$$f_{rij} - \min\{v_{ri}, v_{rj}\}/d_i z_{ij} \leq 0, \qquad (i,j) \in \mathcal{A}^2,\ i \neq j,\ r \in \mathcal{R}, \tag{4}$$

$$\sum_{i \in \mathcal{A}\backslash\{j\}} f_{rij} = v_{rj}, \qquad j \in \mathcal{A}\backslash\{i^+\},\ r \in \mathcal{R}, \tag{5}$$

$$\sum_{j \in \mathcal{A}\backslash\{i\}} f_{rij} = v_{ri}, \qquad i \in \mathcal{A}\backslash\{i^-\},\ r \in \mathcal{R}, \tag{6}$$

$$z_{ij} + z_{ji} \leq 1, \qquad (i,j) \in \mathcal{A}^2,\ i < j, \tag{7}$$

$$z_{ij} + z_{jk} - z_{ik} \leq 1, \qquad (i,j,k) \in \mathcal{A}^3,\ i \neq j,\ i \neq k,\ j \neq k. \tag{8}$$

The objective function (1) minimises the makespan of the schedule by minimising the starting time of the sink activity i^-, while constraint set (2) is required to ensure feasibility in terms of activity precedence.

Constraint set (3) is collectively called the linear ordering constraints. These constraints determine the value of 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 of the large number M in (3) would be the latest possible finishing time of the schedule, *i.e.* $M = L_{i^-} + d_{i^-}$. According to constraint set (4), the flow of resources from activity i to activity j is permitted only if activity j is scheduled to start after the completion of activity i, that is when $z_{ij} = 1$.

The resource requirements are imposed by constraint sets (5) and (6), stating that all the flow of resources into an activity (5) and all the flow of resources out of an activity (6) should match the daily resource requirement v_{ri} of an activity i, for any resource $r \in \mathcal{R}$.

According to Koné *et al.* [9], constraint set (7), which is collectively referred to as directional constraints, ensures that resource flow is either in one direction or the other, or that activities i and j are being processed in parallel, *i.e.* $z_{ij} = 0$ and $z_{ji} = 0$. Constraints (8) are called transitivity constraints which, according to Artigues *et al.* [1], are responsible for ensuring that there are no cycles in the permutations.

Constraint sets (7) and (8) are redundant valid inequalities [11]. No evidence of improvement in computing times are, however, provided in any of the computational results where these valid inequalities have been included in the flow-based RCSP formulation [2, 9].

The computational results reported in this paper show that constraint sets (7) and (8) may, in fact, have a detrimental effect on computing times for some problem instances when included in the formulation of resource flow RCSP models.

3 The resource graph expansion (RGE) heuristic

In addition to the precedence graph, implicity defined by the predecessor and successor sets $\mathcal{P}(i)$ and $\mathcal{S}(i)$, a resource flow graph is implied by the resource flow variables f_{rij} and the linear ordering variables z_{ij}. The basic idea behind the newly proposed RGE heuristic is to incrementally add activities to the resource flow graph while successively generating partial solutions. Once a starting solution to an activity $i \in \mathcal{A}$ has been calculated, the corresponding start-time variable s_i is fixed to this solution in subsequent iterations.

The initial RCSP formulation of the RGE heuristic comprises the constraint sets (2)–(4), which are formulated by taking the entire set of activities into account. The resource flow requirement constraint sets (5)–(6) are initially formulated for a subset of activities $\mathcal{A}' \subseteq \mathcal{A}$, which include only the source activity i^+ and its set of immediate successors $\mathcal{S}(i^+)$. That is, $\mathcal{A}' = \{i^+\} \cup \mathcal{S}(i^+)$. Solving this relaxed version of the RCSP yields a solution that is feasible with respect to the subset of activities $\mathcal{A}'$.

In the following iteration of the heuristic, the start-time variables s_i are fixed to the solutions s_i^* obtained during the previous step for all $i \in \mathcal{A}'$. It should be noted that the fixing of a variable $s_i = s_i^*$ is only allowed if the variables of its predecessors have already been fixed. Next, the subset of activities $\mathcal{A}'$ is augmented with the successors of all of the activities in $\mathcal{A}'$, that is, $\mathcal{A}' = \mathcal{A}' \cup \left(\bigcup_{i \in \mathcal{A}'} \mathcal{S}(i)\right)$. The resource flow requirement constraint sets (5)–(6) are updated each time the subset $\mathcal{A}'$ is augmented. A formal outline of the RGE heuristic is provided in Algorithm 1.

The purpose of solving the RGE heuristic is two-fold. First, it provides a starting solution for the RCSP which may result in a speed-up of the MILP solver, and secondly, it provides an estimate of the scheduling horizon which, in turn, is required to calculate the latest starting times of activities. Specifying MILP-specific stopping criteria provides several variations on the RGE heuristic. For instance, by specifying a gap limit when solving the relaxed RCSP problem during each iteration, a speed-up of the RGE heuristic may be achieved since the branch-and-bound process will be terminated once the current optimality gap is less than the gap limit. This may, of course, be to the detriment of the quality of the final solution obtained by the heuristic. On the other hand, this may result in the successful computation of feasible solutions within the overall time limit specified

Algorithm 1: RGE heuristic

```
1  F ← ∅;
2  Formulate constraints (2)–(4) based on the entire set A;
3  A' ← i⁺ ∪ S(i⁺);
4  Formulate constraints (5)–(6) based on the subset A';
5  Solve RCSP and obtain solutions s_i*, for all i ∈ A';
6  while A' ≠ A do
7      B ← ∅;
8      for i ∈ A' \ F do
9          fixable ← true;
10         for j ∈ P(i) do
11             if j ∉ F then
12                 fixable ← false;
13         if fixable = true then
14             Fix variable s_i = s_i*;
15             F ← F ∪ {i};
16             for j ∈ S(i) do
17                 B ← B ∪ {i};
18     A' ← A' ∪ B;
19     Update constraints (5)–(6) based on the subset A';
20     Solve RCSP and obtain solutions s_i*, for all i ∈ A';
```

for solving the RCSP. The notation RGE(γ) is used in the remainder of this paper to refer to the RGE heuristic where a gap limit of γ is applied during each successive solution of the relaxed RCSP. A gap limit of $\gamma = 0$ implies that the relaxed RCSP is solved to optimality.

Conceptually, the approach toward determining E_i involves solving an optimisation problem in which the objective is to minimise the start time s_i of activity i, subject to the precedence constraints of the RCSP. Similarly, an optimisation problem that maximises the start time s_i of an activity i is solved to determine L_i. It should be noted that some upper bound, say T, is required on s_i in order to prevent an unbounded solution in the case of solving the maximisation problem. An estimate of T is provided by the solution of the RGE heuristic.

The optimisation problem for determining E_i and L_i involves

$$\text{minimising / maximising} \quad s_i \tag{9}$$

subject to the constraints

$$s_j - s_i \leq d_i, \quad (i,j) \in \mathcal{A}^2, \tag{10}$$

$$s_i \leq T, \quad i \in \mathcal{A}, \tag{11}$$

for each activity $i \in \mathcal{A}$.

4 Computational results

All of the empirical tests reported in this section were performed on an HP Compaq Elite 8300 processor, with four cores, 32GB of RAM and a CPU speed of 3.4 GHz. SuSE Linux was used as operating system and the IBM product, CPLEX v12.6 [7], was used as MILP solver.

Several data sets for the RCSP and its variants are available in the research community for the purpose of testing algorithmic ideas. For instance, the *project scheduling problem library* (PSPLIB) [13] is a repository of RCSP problem instances which has been referenced extensively over the years. The PSPLIB comprises the data sets J30, J60, J90 and J120, which are sets of RCSP instances with respectively 30, 60, 90 and 120 activities. Each data set consists of 480 different problem instances, except for the J120 data set which has 600 problem instances. Details on how these problem instances were generated can be found in [8]. Other well-known data sets are the 39 problem instances of Baptiste and Le Pape [5], henceforth referred to as the BL instances, and the Pack instances by Carlier and Néron [6]. In order to test the efficiency of event-based formulations, Koné *et al.* [9] generated the problem instances KSD15_d and Pack_d, which are based on the J30 and Pack instances, respectively. These newly generated instances are characterised by activities having longer durations.

The main objective in this section is to reproduce some of the results reported by Koné *et al.* [9] and to evaluate the effect that the valid inequalities (7) and (8) have on computing times. For this purpose the same data sets used by Koné *et al.* [9] are considered in this paper, with the addition of the J60 data set. Further data sets were generated randomly using the software RanGen2 [14]. Details on the design of RanGen2 can be found in [15]. The major benefit of using RanGen2 is that it allows for the specification of several input parameters which influence the properties of the randomly generated problem instances. For instance, one of the parameters in RanGen2, called I_2, is used to specify the level of serialisation that the resulting precedence graph of the generated problem instance should possess. More specifically, if the value $I_2 = 1$ is specified by the user, a random problem instance is created for which all the activities are serial according to the precedence graph. On the other hand, if $I_2 = 0$, all the activities are in parallel. For the purpose of this study, instances containing 50 or 100 activities were generated randomly using RanGen2. The problem instances in the data sets RG50_L and RG100_L were generated by specifying a low degree of serialisation, that is $I_2 = 0.1$, while the problem instances in RG50_H and RG100_H were generated by specifying a high degree of serialisation, that is $I_2 = 0.5$. The data sets RG50_L, RG50_H, RG100_L and RG100_H each contains 50 problem instances.

An important collective contribution by the research community has been the characterisation of RCSP instances according to various indicators. Some of the indicators used to distinguish between "easy" and "hard" RCSP instances are briefly described:

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 are in parallel, whereas $OS = 1$ indicates that all activities are ordered in series. The hardness of problem instances increases with a decrease in 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 factor (RF) measures the average number of resources required per activity. It has been observed empirically that RF increases with an increase in the hardness of 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 1 contains a summary of the statistics for the above indicators calculated for all of the problem instances considered in this paper. The hardest set of instances, according to the DR indicator, are the BL instances followed by the Pack_d, J60, Pack, KSD15_d and J30 instances. The DR values for the randomly generated data sets RG50_L, RG50_H, RG100_L and RG100_H are much higher and it may appear that all of these instances are easy. The RG50_L and RG100_L data sets, which were generated according to a low degree of serialisation, however, exhibit relatively low OS values and very low RS values, which may suggest a higher degree of difficulty.

		OS		*NC*		*RF*		*RS*		*DR*	
Data set	# activities	avg	σ	avg	σ	avg	σ	avg	σ	avg	σ
J30	30	0.52	0.09	1.81	0.26	0.63	0.28	0.62	0.29	0.56	0.11
J60	60	0.4	0.08	1.81	0.25	0.63	0.28	0.6	0.29	0.41	0.09
BL	22-27	0.34	0.07	1.67	0.13	0.66	0.06	0.34	0.09	0.34	0.07
Pack	17-35	0.23	0.07	1.61	0.06	1	0	0.17	0.08	0.44	0.19
KSD15_d	15	0.47	0.06	1.79	0.2	0.63	0.28	1	0.77	0.51	0.1
Pack_d	17-35	0.23	0.07	1.84	0.27	1	0	0.17	0.07	0.45	0.2
RG50_H	50	0.92	0.01	1.91	0.07	1	0	0.26	0.69	0.99	0
RG50_L	50	0.38	0.01	4.57	0.12	1	0	0.03	0.02	0.93	0.01
RG100_H	100	0.96	0	1.89	0.04	1	0	0.1	0.35	0.99	0
RG100_L	100	0.65	0	5.84	0.04	1	0	0.01	0.016	0.96	0

Table 1: Tractability indicators for the RCSP instances considered in this paper.

For the purpose of reporting computational results, the resource flow formulation of the RCSP given by (1)–(8) is denoted by RF. The abreviation RFX is used to refer to the resource flow formulation that excludes the valid inequalities, that is, the formulation given by (1)–(6). In order to measure the effect of employing the RGE heuristic, the notation RFX+RGE(γ) and RF+RGE(γ) are used to refer to the combination of the RFX formulation and the RF formulation with the use of the RGE heuristic, respectively. Gap limits of $\gamma = 0\%$, $\gamma = 50\%$ and $\gamma = 100\%$ are considered.

The first set of results is provided in Table 2 for the same problem instances that were considered in [9]. For the J30 data set, both the RFX and the RF formulations were successful in solving 75% of the problem instances to optimality. The average time it took to solve the RFX formulation is 19.1 seconds compared to the 27.8 seconds that it took the RF formulation. In the case of the BL problem instances, the RF formulation facilitated the solution of 15% of the instances to optimality, compared to the 13% solved by the RFX formulation. The average gap obtained for the RFX formulation, however, is 18.6% compared to the 19.6% achieved by the RF formulation. The results obtained for the "easier" KSD15_d instances show that the RF formulation performs better than the RFX formulation overall. For both the Pack and Pack_d problem instances the RFX performed better again. It allowed for the computation of solutions exhibiting, on average, a gap of 59.7% and 44.6%, respectively, compared to the RF obtaining average gaps of 62.6% and 49.3%, respectively.

Some improvements were observed when applying the RGE heuristic. For instance, in the case of the J30 problem instances, the average time required by the RFX+RGE(100%) to solve the same number of instances to optimality is 14.9 seconds *vs.* 19.1 seconds when applying only the RFX formulation. In the case of BL, 18% of the problem instances were solved to optimality according to the RFX+RGE(100%) formulation, compared to only 13% when applying the RFX. No improvements were, however, observed for the Pack and the KSD_15 problem instances and a marginal improvement in the average gap was obtained for the Pack_d instances.

The main conclusion drawn from the first set of results provided in Table 2 is that there is merit to excluding the directional and transitivity inequalities (7)–(8) from the resource flow-based RCSP formulation. This may have an effect on the conclusions made by, for instance, Koné *et al.* [9] with respect to the success of event-based formulations over resource-flow formulations which include the directional and transitivity inequalities (7)–(8). The results that follow for the remaining problem instances considered in this paper are even more convincing in this regard.

The significance of excluding the directional and transitivity inequalities (7)–(8) from the RCSP problem formulation is clearly demonstrated by the results for the J60 data set in Table 3. Adopting the RFX formulation, a total of 67% of the J60 problem instances were solved to optimality, compared to 23% for the RF formulation. Further improvements were achieved by employing RFX+RGE(50%) in which an average gap of 9.4% was achieved by generating feasible solutions to all of the problem instances, compared to 9.7% in the case of the plain RFX formulation.

Although the RFX formulation is outperformed by the RF formulation in the case of the "easier" RG50_H and RG100_H problem instances, very promising results are obtained for the "harder" RG50_L and RG100_L problem instances. The RFX allowed for the solution of 46% of the RG50_L problem instances, whereas the RF could not facilitate the solution of any of the problem instances to optimality. Improvements were once again achieved by RFX+RGE(50%) in which case an average gap of 5.2% was obtained by generating feasible solutions to all of the problem instances, compared to 6.9% in the case of the plain RFX formulation. Although none of the RG100_L problem instances could be solved to optimality, the benefits of applying RFX are still clearly visible by considering that an

		Solved to optimality		Feasible solution	
Data set	Formulation	# Instances (%)	Time (s)	# Instances (%)	Gap (%)
J30	RFX	75	19.1	100	5.8
	RFX+RGE(0%)	75	13.8	98	5
	RFX+RGE(50%)	75	18.7	100	5.7
	RFX+RGE(100%)	75	14.9	100	5.8
	RF	75	27.8	100	6.8
	RF+RGE(0%)	75	19.1	98	5.7
	RF+RGE(50%)	74	25	100	6.8
	RF+RGE(100%)	75	29.2	100	6.8
BL	RFX	13	118.8	100	18.6
	RFX+RGE(0%)	13	121.7	13	0
	RFX+RGE(50%)	15	93.5	100	19.1
	RFX+RGE(100%)	18	124.5	100	18.6
	RF	15	220.4	100	19.6
	RF+RGE(0%)	13	126.8	13	0
	RF+RGE(50%)	15	212.8	100	20.2
	RF+RGE(100%)	15	200.4	100	19.8
Pack	RFX	0	—	100	59.7
	RFX+RGE(0%)	0	—	18	31.2
	RFX+RGE(50%)	0	—	41	43.9
	RFX+RGE(100%)	0	—	100	59.8
	RF	0	—	100	62.6
	RF+RGE(0%)	0	—	18	30.7
	RF+RGE(50%)	0	—	41	44.5
	RF+RGE(100%)	0	—	100	62.7
KSD_15	RFX	96	6.7	100	0.9
	RFX+RGE(0%)	96	5.8	100	0.9
	RFX+RGE(50%)	96	5	100	0.9
	RFX+RGE(100%)	20	3.2	21	0.3
	RF	99	6.9	100	0.1
	RF+RGE(0%)	99	8	100	0.1
	RF+RGE(50%)	99	7	100	0.2
	RF+RGE(100%)	21	5.5	21	0
Pack_d	RFX	9	0.3	96	44.6
	RFX+RGE(0%)	7	0.2	16	12.6
	RFX+RGE(50%)	7	0.3	43	28.2
	RFX+RGE(100%)	9	0.4	96	44.5
	RF	9	7.7	96	49.3
	RF+RGE(0%)	7	1.8	16	12.2
	RF+RGE(50%)	7	3.2	43	30
	RF+RGE(100%)	9	7.7	96	49.7

Table 2: The effect of the directional and transitivity inequalities (7)–(8) on computing times for the problem instances considered in [9].

average gap of 32.6% was obtained for all of the cases for which at least one feasible solution could be computed. Feasible solutions could be computed for only 96% of the RG100_L problem instances according to the RF formulation.

The distinction between "easy" and "hard" instances for the RG50 and RG100 data sets are only based on the level of OS. That is, the easier problem instances RG50_H and RG100_H are associated with higher OS levels, while the harder instances RG50_L and

Data set	Formulation	Solved to optimality		Feasible solution	
		# Instances (%)	Time (s)	# Instances (%)	Gap (%)
J60	RFX	67	50	100	9.7
	RFX+RGE(0%)	66	21.6	78	1.4
	RFX+RGE(50%)	66	35.9	100	9.4
	RFX+RGE(100%)	66	37.9	100	9.6
	RF	23	146.9	73	37.1
	RF+RGE(0%)	45	30.5	78	5.2
	RF+RGE(50%)	30	128	100	29.4
	RF+RGE(100%)	27	144.2	92	31.6
RG50_H	RFX	100	1	100	0
	RFX+RGE(0%)	100	4.1	100	0
	RFX+RGE(50%)	100	3.9	100	0
	RFX+RGE(100%)	100	3.5	100	0
	RF	100	0.5	100	0
	RF+RGE(0%)	100	4.4	100	0
	RF+RGE(50%)	100	3.9	100	0
	RF+RGE(100%)	100	3.5	100	0
RG50_L	RFX	46	201.6	100	6.9
	RFX+RGE(0%)	34	140.5	42	1.1
	RFX+RGE(50%)	38	180.5	100	5.2
	RFX+RGE(100%)	42	132.6	90	7.6
	RF	0	—	96	62.5
	RF+RGE(0%)	12	136.9	42	8.9
	RF+RGE(50%)	0	—	100	45.4
	RF+RGE(100%)	0	—	88	57.4
RG100_H	RFX	100	8.9	100	0
	RFX+RGE(0%)	76	93.3	76	0.7
	RFX+RGE(50%)	80	79.2	80	2.1
	RFX+RGE(100%)	80	75.5	80	1.9
	RF	100	2.9	100	0
	RF+RGE(0%)	76	94.7	76	1.3
	RF+RGE(50%)	80	79.1	80	2.4
	RF+RGE(100%)	80	75.1	80	1.7
RG100_L	RFX	0	—	100	32.6
	RFX+RGE(0%)	0	—	30	9
	RFX+RGE(50%)	0	—	100	27.8
	RFX+RGE(100%)	0	—	90	27.8
	RF	0	—	96	87
	RF+RGE(0%)	0	—	30	10.2
	RF+RGE(50%)	0	—	100	55.7
	RF+RGE(100%)	0	—	90	69.1

Table 3: The effect of the directional and transitivity inequalities (7)–(8) on computing times for the remaining problem instances.

RG100_L are associated with lower OS levels. From the above results it is therefore reasonable to assume that improvements in computing times may be expected for problem instances characterised by low OS values if the directional and transitivity inequalities (7)–(8) are excluded. As indicated in Table 1, all of the RG50 and RG100 instances are considered easy when measured according to the DR tractability indicator. The question raised here is whether the RFX is effective for problem instances characterised by a low

DR value. The results of Table 2 may hint at the contrary considering that the RF outperformed the RFX in the case of the BL data set, which is considered to be the hardest since it has the lowest average DR value. In order to explore this further, attention is drawn to the positive results obtained for the J60 problem instances. The effectiveness of the RFX in respect of the J60 problem instances is analysed with respect to different levels of DR. More specifically, the ranges $[0, \mu - \sigma]$ and $[\mu + \sigma, 1]$ are applied as filters on DR values in order to create the segments J60_L and J60_H, which correspond to J60 instances with a low DR value (hard) and J60 instances with a high DR value (easy), respectively. The resulting number of activities in the J60_L data set is 106 with an average DR value of 0.29. For the J60_H data set, the resulting number of activities is 88 with an average DR value of 0.54.

Positive results are reported in Table 4 for the J60_L and J60_H problem instances. By making use of the RFX formulation, which excludes the directional and transitivity inequalities (7)–(8), 73% of the total number of J60_L instances are solved to optimality. For the RF formulation, which includes the directional and transitivity inequalities (7)–(8), only 44% of the instances are solved to optimality. It is also encouraging to note that the application of RFX+GRE(50%) improves the average gap, managing to generate feasible solutions for all of the problem instances, with an average gap of 6.2% *vs.* the 6.8% gap achieved according to the plain RFX formulation without the heuristic. Results for the J60_H data set are also positive, showing that 49% of the instances are solved to optimality according to the RFX *vs.* 30% when applying the RF.

		Solved to optimality		Feasible solution	
Data set	Formulation	# Instances (%)	Time (s)	# Instances (%)	Gap (%)
J60_L	RFX	73	21.3	100	6.8
$\|$J60_L$\| = 106$	RFX+RGE(0%)	74	39	88	1.2
$\mu(DR) = 0.29$	RFX+RGE(50%)	73	34.6	100	6.2
$\sigma(DR) = 0.02$	RFX+RGE(100%)	19	172	100	6.6
	RF	44	11.3	77	50.5
	RF+RGE(0%)	25	155.1	88	6
	RF+RGE(50%)	19	114.7	100	29.8
	RF+RGE(100%)	0	—	97	38.6
J60_H	RFX	49	54.2	100	18.1
$\|$J60_H$\| = 88$	RFX+RGE(0%)	49	26.1	59	2
$\mu(DR) = 0.54$	RFX+RGE(50%)	48	33.3	100	17.6
$\sigma(DR) = 0.03$	RFX+RGE(100%)	47	24.8	100	17.8
	RF	30	136.6	77	28.6
	RF+RGE(0%)	36	32.7	59	5.5
	RF+RGE(50%)	31	101.2	100	33.8
	RF+RGE(100%)	27	90.3	84	30.4

Table 4: The effect of the directional and transitivity inequalities (7)–(8) on computing times for the J60_L and J60_H segments based on the DR ranges $[0, \mu - \sigma]$ and $[\mu + \sigma, 1]$, respectively.

Although the average DR values for the two segments J60_L and J60_H are lower than that of the BL data set, the average RS values are higher. A further analysis involves refining the segments J60_L and J60_H in such a way that they represent problem instances with both low DR and low RS values. For this purpose, the ranges $[0, \mu - \frac{1}{3}\sigma]$ and $[\mu + \frac{1}{3}\sigma, 1]$

were applied as filters on RS values for each of the J60 problem instances, in addition to the DR filters $[0, \mu - \sigma]$ and $[\mu + \sigma, 1]$. The motivation for the choice of filters is to obtain a reasonably sized J60_L segment for which both the average DR and average RS are lower than that of the BL data set. The resulting number of activities in the refined J60_L is 19, with an average DR value of 0.29 and an average RS value of 0.3. For the J60_H data set the resulting number of activities is 25, with an average DR value of 0.52 and an average RS value of 0.84.

Results for the refined J60_L and J60_H problem instances are reported in Table 5. The results show that by adopting the RFX formulation, 11% of the total number of J60_L instances are solved to optimality. The RF formulation, which includes the directional and transitivity inequalities (7)–(8), does not produce any optimal solution for any of the J60_L problem instances. Although feasible solutions are obtained for all of the J60_L problem instances by making use of the RF formulation, the average gap is more than double the average gap obtained by the RFX. Results for the J60_H data set are also positive, showing that 96% of the instances are solved to optimality according to the RFX *vs.* 72% when applying RF.

		Solved to optimality		Feasible solution	
Data set	Formulation	# Instances (%)	Time (s)	# Instances (%)	Gap (%)
J60_L	RFX	11	59.7	100	32.9
$\|$J60_L$\| = 19$	RFX+RGE(0%)	11	24	100	8.5
$\mu(DR) = 0.29$	RFX+RGE(50%)	11	41.5	42	31.7
$\mu(RS) = 0.3$	RFX+RGE(100%)	11	76.6	100	32.6
	RF	0	—	100	76.6
	RF+RGE(0%)	0	—	79	21.5
	RF+RGE(50%)	0	—	42	58
	RF+RGE(100%)	0	—	100	68
J60_H	RFX	96	35.8	100	0
$\|$J60_H$\| = 25$	RFX+RGE(0%)	100	14.9	100	0
$\mu(DR) = 0.52$	RFX+RGE(50%)	96	27.7	100	0
$\mu(RS) = 0.84$	RFX+RGE(100%)	96	18.9	100	0
	RF	72	116.3	100	9.8
	RF+RGE(0%)	84	6.2	92	1.1
	RF+RGE(50%)	76	99.3	100	7.1
	RF+RGE(100%)	68	101.4	100	9.3

Table 5: The effect of the directional and transitivity inequalities (7)–(8) on computing times for the refined J60_L and J60_H segments based on the DR ranges $[0, \mu - \sigma]$ and $[\mu + \sigma, 1]$, and the RS ranges $[0, \mu - \frac{1}{3}\sigma]$ and $[\mu + \frac{1}{3}\sigma, 1]$, respectively.

As a final analysis, Table 6 provides the average performance measures for the RFX and heuristic combinations over all of the problem instances. At a first glance, use of the GRE heuristic does not appear to be beneficial since the RFX formulation without the heuristic achieves the highest percentage of instances solved to optimality. Note that these are averages over all of the problem instances. On closer inspection it is observed that the application of the heuristic is especially useful when considering harder problem instances. For instance, the application of RFX+RGE(50%) results in improved average gaps for the harder instances, such as J60, RG50_L and RG100_L, but not necessarily for the easier problem instances. Table 6 also indicates that, on average, $\gamma = 50\%$ may be

a good parameter choice since RFX+RGE(50%) solves the most instances to optimality, compared to other choices of γ.

	Solved to optimality		Feasible solution	
Formulation	# Instances (%)	Time (s)	# Instances (%)	Gap (%)
RFX	51.6	44.6	99.7	17.3
RFX+RGE(0%)	48.1	37.7	59.1	5.5
RFX+RGE(50%)	49.0	41.4	88.7	14.0
RFX+RGE(100%)	43.3	38.7	89.8	16.9

Table 6: Computational results for evaluating the effectiveness of the GRE heuristic over all problem instances.

5 Summary and conclusion

The primary concern of this paper has been an investigation into how directional and transitivity valid inequalities may influence computing times when included in the formulation of the flow-based RCSP. The trend emerging from the computational results shows that improved computing times may be expected when excluding these valid inequalities, especially when considering "harder" problem instances. This warrants a re-evaluation of the results presented by others which suggested that event-based RCSP formulations may perform better than resourc flow-based formulations.

As a secondary contribution, a heuristic was proposed for the purpose of generating initial feasible solutions and estimating the scheduling horison necessary for the computation of the latest start dates of the activities. Positive results were reported for specifically harder problem instances.

References

[1] Artigues C, Demassey S & Nèron E (Eds) 2008, *Resource-constrained project scheduling: Models, algorithms, extensions and applications*, John Wiley & Sons Inc., Hoboken (NJ).

[2] Artigues C, Leus R & Nobibon FT, 2013, *Robust optimization for resource-constrained project scheduling with uncertain activity durations*, Flexible Services and Manufacturing Journal, **25(1)**, pp. 175–205.

[3] Artigues C, Michelon P & Reusser S, 2003, *Insertion techniques for static and dynamic resource-constrained project scheduling*, European Journal of Operational Research, **149(2)**, pp. 249–267.

[4] Artigues C & Roubellat F, 2000, *A polynomial insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes*, European Journal of Operational Research, **127(2)**, pp. 297–316.

[5] Baptiste P & Pape CL, 2000, *Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems*, Constraints, **5(1)**, pp. 119–139.

[6] Carlier J & Néron E, 2003, *On linear lower bounds for the resource constrained project scheduling problem*, European Journal of Operational Research, **149(2)**, pp. 314–324.

[7] IBM, *IBM ILOG CPLEX*, [Online], Available from http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/, [Accessed July 9th, 2015].

[8] Kolish R & Sprecher A, 1996, *PSPLIB — A project scheduling problem library*, European Journal of Operational Research, **96(1)**, pp. 205–216.

[9] Koné O, Artigues C, Lopez P & Mongeau M, 2011, *Event-based MILP models for resource-constrained project scheduling problems*, Computers and Operations Research, **38(1)**, pp. 3–13.

[10] Mingozzi A, Maniezzo V, Ricciardelli S & Bianco L, 1998, *An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation*, Management Science, **44(5)**, pp. 714–729.

[11] Olaguíbel R & Goerlich J, 1993, *The project scheduling polyhedron: Dimension, facets and lifting theorems*, European Journal of Operational Research, **67(2)**, pp. 204–220.

[12] Pritsker A, Watters L & Wolfe P, 1969, *Multi-project scheduling with limited resources: A zero-one programming approach*, Management Science, **16(1)**, pp. 93–108.

[13] PSPLIB, *A project scheduling problem library*, [Online], Available from http://www.om-db.wi.tum.de/psplib/main.html, [Accessed July 9th, 2015].

[14] RanGen2, *A random network generator for RCSP instances*, [Online], Available from http://www.projectmanagement.ugent.be/?q=research/data/RanGen, [Accessed July 9th, 2015].

[15] Vanhoucke M, Coelho J, Debels D, Maenhout B & Tavares L, 2008, *An evaluation of the adequacy of project network generators with systematically sampled networks*, European Journal of Operational Research, **187(2)**, pp. 511–524.

[16] Zapata J, Hodge B & Reklaitis G, 2008, *The multimode resource constrained multiproject scheduling problem*, AIChE Journal, **54(8)**, pp. 2101–2119.