



Resource-constrained multi-project scheduling: benchmark datasets and decoupled scheduling

Rob Van Eynde¹ · Mario Vanhoucke^{1,2,3}

Published online: 10 April 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

In this paper, we propose a new dataset for the resource-constrained multi-project scheduling problem and evaluate the performance of multi-project extensions of the single-project schedule generation schemes. This manuscript contributes to the existing research in three ways. First, we provide an overview of existing benchmark datasets and classify the multi-project literature based on the type of datasets that are used in these studies. Furthermore, we evaluate the existing summary measures that are used to classify instances and provide adaptations to the data generation procedure of Browning and Yassine (J Scheduling 13(2):143-161, 2010a). With this adapted generator we propose a new dataset that is complimentary to the existing ones. Second, we propose decoupled versions of the single-project scheduling schemes, building on insights from the existing literature. A computational experiment shows that the decoupled variants outperform the existing priority rule heuristics and that the best priority rules differ for the two objective functions under study. Furthermore, we analyse the effect of the different parameters on the performance of the heuristics. Third, we implement a genetic algorithm that incorporates specific multi-project operators and test it on all datasets. The experiment shows that the new datasets are challenging and provide opportunities for future research.

Keywords Multi-project scheduling · Portfolio scheduling · Summary measures · Decoupled scheduling · Benchmark data

1 Introduction

The resource-constrained project scheduling problem (RCPSp) is a well-known scheduling problem where a set of activities needs to be scheduled subject to precedence and resource constraints in order to optimise an objective function (e.g. minimise makespan). This problem has a wide variety of applications and extensions (Hartmann and Briskorn 2010). The resource-constrained multi-project scheduling problem (RCMPSP) extension deals with a *portfolio* of multiple projects that require the same resources. The problem is to construct a schedule, where activities of different projects

can be executed in parallel as long as the resource constraints are not violated. In the last decades, more attention has been devoted to this multi-project extension because most companies have multiple active projects in their portfolio and most of the value is situated in this multi-project setting (Payne 1995; Maroto et al. 1999; Liberatore and Pollack-Johnson 2003). Even though most companies operate in a multi-project environment, the sharing of resources over projects might not be feasible (e.g. due to geographical constraints) or not desirable. In this case, resources are dedicated to the individual projects, such that each project can be scheduled individually and the problem can be treated as a single-project scheduling problem (Beşikci et al. 2013).

As the RCMPSP has received less attention from researchers than its single-project variant, the literature lags behind with regard to both summary measures to classify data and standardised datasets to benchmark algorithms. In this paper, we will review the incumbent summary measures and make some adaptations to them. Furthermore, we will evaluate the existing datasets and propose an alternative dataset which includes parameter combinations that are absent in the incumbent sets. We introduce two sched-

✉ Mario Vanhoucke
mario.vanhoucke@ugent.be

Rob Van Eynde
rob.vaneynde@ugent.be

¹ Faculty of Economics and Business Administration,
Ghent University, Tweeckerkenstraat 2, 9000 Ghent, Belgium

² Vlerick Business School, Reep 1, 9000 Ghent, Belgium

³ UCL School of Management, University College London,
1 Canada Square, London E14 5AA, UK

ule generation schemes that take the specific multi-project structure into account. Some previous research suggested that incorporating this multi-project information could lead to well-performing algorithms (Asta et al. 2016; Mittal and Kanda 2009). We evaluate the performance of these schemes in a computational experiment on both existing and new datasets. Furthermore, we implement a genetic algorithm and test it on all datasets.

The outline of the paper is as follows. Section 2 first discusses the basic problem formulation and extensions for the RCMPSP. Then, the existing summary measures and datasets are reviewed. In Sect. 3, we explain our multi-project extensions of the single-project schedule generation schemes (SGS). In Sect. 4, we adapt the generation procedure of Browning and Yassine (2010b) and use it to generate a new benchmark dataset. We evaluate the new SGS' and the genetic algorithm on the existing and new datasets in Sect. 5, and we conclude the manuscript in Sect. 6.

2 Problem Formulation

In this section, we will first give a formal description of the basic RCMPSP with according notation. Then, we will review the literature with regard to extensions of the RCMPSP. Third, we will discuss the existing multi-project measures and we conclude this section by reviewing the existing benchmark datasets.

2.1 Description

A multi-project portfolio consists of a set of projects J , each with a set of activities I_j . The total number of activities in the multi-project is denoted by s . Activity i of project j is denoted by a_{ij} and has a duration of d_{ij} . Precedence relations between activities may exist. The set of predecessors of a_{ij} is denoted by P_{ij} . An activity a_{ij} can only be started when all $a_{kj} \in P_{ij}$ are finished. A project can be graphically represented by a directed acyclic graph (also called a project network) with a node for each activity and an arc for each precedence relation. A project network also has a dummy start and dummy end activity with a duration of zero to indicate when the project is started and completed. Each project also has a set of renewable resources types K that are shared among all projects. Each resource type has a per period availability of R_k . Activity a_{ij} has a constant per period resource requirement of r_{ijk} for each $k \in K$. Since it is not necessary that all activities require each resource type, H_{ij} denotes the set of resource types for which a_{ij} has a non-negative demand.

A schedule for a multi-project is an assignment of a start time to each activity. When a project has a release date r_j , it cannot be started before this date. Furthermore, a project can have a due date \bar{d}_j , i.e. the desired finish time for the project.

Based on the start times of the activities, the values of the project start and finish times can be derived. The start time (S_j) of project j is the start time of the first non-dummy activity of that project. The finish time of project j is denoted by F_j and is equal to the finish time of its dummy end activity. The makespan of a project (M_j) is the difference between F_j and S_j . The portfolio makespan (M) is the difference between the latest finish time and the first start time over all projects in the portfolio. The critical path duration is calculated by constructing the earliest start schedule where each activity starts as soon as possible, respecting the precedence relations but neglecting resource constraints. The makespan of a project in this schedule is the critical path duration CP_j , and the longest critical path in the portfolio is denoted by CP_{\max} . The portfolio makespan of the earliest start schedule is indicated by CP .

In this paper, we will address two objective functions that are commonly used in the literature: the average per cent delay (APD) and the portfolio delay (PDEL) (Browning and Yassine 2010b). The APD is defined as follows:

$$APD = \frac{1}{|J|} \sum_{j \in J} \frac{M_j - CP_j}{CP_j}. \quad (1)$$

APD normalises the makespan of a project over its critical path duration and averages it over all projects in the portfolio. As it takes all projects into consideration, this objective function can be classified under the multi-project approach. The PDEL is defined by

$$PDEL = \frac{M - CP_{\max}}{CP_{\max}}. \quad (2)$$

This objective function only evaluates the portfolio makespan and normalises it by CP_{\max} . As such, it is equivalent to combining all projects into one single-project network and minimising the single-project makespan. Nevertheless, we chose to include this objective in our study to resolve conflicting results from the literature regarding which priority rules perform best. We come back to this in Sect. 3.

2.2 Extensions

In the previous subsection, we defined the basic RCMPSP, but researchers have extended this problem in six ways. First of all, in the multi-mode extension (MRCMPSP), each activity has several execution modes where each mode has a corresponding cost and duration and there is an available budget. The objective is to minimise a time-related objective while respecting the budget constraint. This extension was the subject of the MISTA 2013 challenge (Wauters et al. 2016), and several of these submitted algorithms have been published (Asta et al. 2016; Geiger 2017; Toffolo et al. 2016). Ver-

cellis (1994) also addresses the multi-mode extension, but with the objective of maximising the net present value of the project. A second extension that has received considerable attention is the decentralised RCMPSP (DRCMPSP), which relaxes the assumption that all information is centrally available (Confessore et al. 2007; Homberger 2007). Instead, a schedule is the result of a negotiation process between different project managers and possibly a mediating agent, each having access to only a part of all available information. Researchers mainly solve this problem with multi-agent systems, where virtual agents negotiate with each other about allocated resources, start times, etc. Third, the dynamic RCMPSP relaxes the assumption that all projects in the portfolio are known at the decision time and assume new projects can arrive during project execution. In this context, previous scheduling decisions may have to be revised and activities may be preempted when new projects arrive into the system (Yang and Sum 1997; Wang et al. 2015). Fourth, the stochastic RCMPSP relaxes the assumption that all information is deterministic. In the study of Wang et al. (2015), the resource availability is subject to uncertainties, while Zheng et al. (2013) and Wang et al. (2017) address the problem where activity durations are stochastic. Fifth, Beşikci et al. (2013) address the resource dedication problem (RDP), where amounts of resources need to be allocated to the different projects in the portfolio. Once allocated, the resources are not shared among the projects, which reduces the problem to separate single-project scheduling problems. This research is extended in Beşikci et al. (2015) to a multi-project environment where first a budget is allocated to each resource type, then the resource dedication problem is solved, and finally, the individual projects are scheduled. The last extension to the RCMPSP incorporates resource transfer times, where transferring a resource from one project or activity to another may incur a delay. This problem is studied in Krüger and Scholl (2009, 2010). Resource transfers have been investigated in a static decentralised environment (Adhau et al. 2013) and in a dynamic decentralised environment (Yang and Sum 1993, 1997). This paper treats the basic RCMPSP with two time-related objectives, but further research could extend the insights towards other objective functions or towards extensions of the basic problem (Table 1).

2.3 Existing multi-project measures

Similar to the single-project literature, summary measures have been introduced to classify multi-project instances. The existing measures are reviewed in Browning and Yassine (2010b) (BY10), the authors use one network measure and three resource measures to generate their data. C_j describes the network structure of the projects in the portfolio. The resource-related measures are the *normalised*

Table 1 Summary of symbols

<i>Basic parameters</i>	
J	The set of projects, indexed by j
I_j	The set of activities of project j , indexed by i
s	$\sum_j I_j $
a_{ij}	Activity i of project j
d_{ij}	The duration of a_{ij}
P_{ij}	The set of predecessors of a_{ij}
<i>Resource parameters</i>	
K	The set of renewable resource types, indexed by k
R_k	The per period availability of resource type k
r_{ijk}	The resource requirement of a_{ij} for resource type k
H_{ij}	The set of resource types for which the demand of a_{ij} is nonzero
<i>Schedule parameters</i>	
r_j	The release date of project j
\bar{d}_j	The due date of project j
CP_j	The critical path duration of project j
CP_{\max}	$\max_{j \in J} CP_j$
CP	$\max_{j \in J} (r_j + CP_j) - \min_{j \in J} r_j$
<i>Decision variables</i>	
S_j	The start time of project j
F_j	The finish time of project j
M_j	$F_j - S_j$
M	$\max_{j \in J} F_j - \min_{j \in J} S_j$

average resource loading factor, the *modified average utilisation factor* and its *variance*. In the remainder of this subsection, we will discuss these measures and propose some extensions or adaptations.

Network complexity (C_j): This measure compares the number of non-redundant arcs to the theoretical maximum number of arcs. An arc (i, k) is redundant or transitive when (i, j) and (j, k) are both arcs in the network. C_j is calculated as follows:

$$C_j = \frac{A' - A'_{\min}}{A'_{\max*} - A'_{\min}} \quad (3)$$

where A' is the number of non-redundant arcs, $A'_{\min} = |I_j| - 1$ and $A'_{\max*}$ is the maximum of non-redundant arcs, which depends on the length L_j of the network. L_j is defined as the length of the longest path of non-dummy nodes between the dummy start to the dummy end node. A' can have values between A'_{\min} and $A'_{\max*}$, so C_j lies in the interval $[0, 1]$. BY10 generated a dataset of project networks with a low ($C_j = 0.14$) and a high ($C_j = 0.69$) number of arcs. They report that the length L_j , averaged over the three networks in one portfolio lies in the range $[2, 11.67]$.

Table 2 Network structure of the data from BY10

	Min	Avg	Max
L_j	2	2.41	5
SP	0.05	0.07	0.21
OS	0.03	0.05	0.21

The structure of a project network has been described in the single-project literature by different measures. An overview of the measures is given by Vanhoucke et al. (2008); a review on the strengths and weaknesses of the measures can be found in Demeulemeester et al. (2003). In the following discussion, the progressive level \mathcal{P}_{ij} of an activity will be used, which is defined as the longest path from the dummy start to the activity a_{ij} (Elmaghraby 1977). We will now evaluate the network structure of the instances in BY10 by calculating their values for two well-known metrics: the order strength (OS) and the serial–parallel (SP) indicator. The OS compares the total of all redundant and non-redundant arcs in the network with the theoretical maximum for a network of that size (independent of the length of the network), leading to values in the range $[0,1]$. SP divides $L_j - 1$ by $|I_j| - 1$, its value lies in the range $[0,1]$. For both measures, 0 corresponds to the completely parallel network and 1 to the completely serial network. Table 2 reports the minimum, average and maximum value over all instances for L_j , SP and OS. Even instances with high C_j values of 0.69 score low on SP and OS. With an average length of 2.41, the instances in the BY10 dataset are rather parallel. The most serial network in the BY10 set has a length of 5, while each network consists of 20 nodes.

The discrepancy between the values of C_j and OS/SP values can be attributed to the discriminative power of C_j and the generation procedure. First, C_j has a lower discriminative power between serial and parallel instances than OS and SP. While the extremes of OS and SP correspond to the completely parallel and serial networks, this is not the case for C_j . As a consequence, it is not easy to interpret C_j unambiguously. Browning and Yassine (2010a) state that the minimum number of arcs ($C_j = 0$) is obtained for both the completely serial and parallel network. The maximum number of arcs ($C_j = 1$) is obtained by assigning $(|I_j| - L_j + 2)/2$ activities to two consecutive progressive levels and 1 activity to the others. Additionally, C_j depends on L_j because L_j determines the value of A'_{\max} . Two networks with different lengths may have the same value for C_j , even though the one with the higher L_j is more serial than the other.

The second cause originates in the generation procedure. The authors specify the length of a network (i.e. the number of progressive levels) and a value for C_j . The procedure then assigns each activity to a progressive level. In order to enforce this progressive level \mathcal{P}_{ij} of an activity a_{ij} , it should

have at least one predecessor $a_{ij'}$ with $\mathcal{P}_{ij'} = \mathcal{P}_{ij} - 1$. In the dataset, this condition is not always satisfied, leading to more parallel instances than originally specified because activities may have a lower progressive level. As a consequence, the critical path durations of the instances may also differ, which impacts the values for the resource measures which are discussed hereafter.

The Normalised Average Resource Loading Factor (NARLF): Kurtulus and Davis (1982) defined the average resource loading factor (ARLF) to indicate whether the total resource requirement of a project mainly occurs in the first or second half of its critical path duration. To compute it, they define two auxiliary variables:

$$Z_{ijt} = \begin{cases} -1 & \text{if } t \leq r_j + (CP_j/2), \\ 1 & \text{if } t > r_j + (CP_j/2) \end{cases} \tag{4}$$

$$X_{ijt} = \begin{cases} 1 & \text{if activity } a_{ij} \text{ is active at time } t, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Z_{ijt} indicates whether time t falls in the first or second half of CP_j , taking into account the release date of the project. X_{ijt} is equal to one if a_{ij} is being executed or active at time t according to the resource-unconstrained earliest start schedule. Now, the $ARLF_j$ of project j can be calculated with

$$ARLF_j = \frac{1}{CP_j} \sum_{t=r_j}^{r_j+CP_j} \sum_{i \in I_j} \sum_{k \in H_{ij}} Z_{ijt} X_{ijt} \left(\frac{r_{ijk}}{|H_{ij}|} \right) \tag{6}$$

In this formula, the resource demand of activities that fall in the first (second) half of the critical path of project j has a negative (positive) contribution to the $ARLF_j$. When the largest fraction of the demand occurs in the first (second) half, the $ARLF_j$ will be negative (positive) and the project is called *front loaded* (*back loaded*). The ARLF of the portfolio is obtained by averaging over all projects:

$$ARLF = \frac{1}{|J|} \sum_{j \in J} ARLF_j \tag{7}$$

BY10, however, state that this measure may produce biased results and propose the NARLF. Equation 6 calculates the $ARLF_j$ of every individual project relative to its own critical path duration. BY10 argue that when the durations of the projects differ significantly, ARLF provides misleading results.

$$NARLF = \frac{1}{|J| * CP_{\max}} \sum_{j \in J} \sum_{t=r_j}^{r_j+CP_j} \sum_{i \in I_j} \sum_{k \in H_{ij}} Z_{ijt} X_{ijt} \left(\frac{r_{ijk}}{|H_{ij}|} \right) \tag{8}$$

Equation 8 partially solves this issue, as the resource distribution of every project is normalised over the duration of

Fig. 1 NARLF vs NARLF': an illustration

		Project 1	Project 2	Portfolio
Portfolio 1	Demand profile			
	NARLF	0	0	0
	NARLF'	0	0	0
Portfolio 2	Demand profile			
	NARLF	0	0	0
	NARLF'	0	-1.5	-0.75

the whole portfolio (CP_{max}). However, we believe the measure can be further improved since Z_{ijt} is still calculated with regard to the critical path duration of the individual projects. In order to resolve this, we propose a new definition Z'_t :

$$Z'_t = \begin{cases} -1 & \text{if } t \leq \min_{j \in J} (r_j) + \lceil CP/2 \rceil, \\ 1 & \text{if } t > \min_{j \in J} (r_j) + \lceil CP/2 \rceil \end{cases} \quad (9)$$

This new variable indicates whether a time point falls in the first or second half of the critical path duration of the portfolio rather than the critical path durations of the individual projects. NARLF' is calculated with Eq. 8 where Z_{ijt} is replaced by Z'_t . We demonstrate the difference between the two measures with the example in Fig. 1. There are two portfolios, each consisting of two projects. For each of the individual projects, the demand is equally distributed over its own critical path duration. The half of the critical path duration is indicated by the bold line. In the right hand side of the table, the demand profiles of the portfolios are shown. These are obtained by aggregating the demand profiles of the constituent projects. Portfolio 1 has a constant resource demand over its duration, leading to a value of 0 for both NARLF and NARLF'. However, in portfolio 2 the biggest part of the resource demand clearly occurs in the first half of CP_{max} (the first two periods). According to NARLF, there is no difference in the loading of the resource profile of portfolio 1 and 2, even though the latter is more front loaded. NARLF', on the other hand, correctly indicates that the second portfolio is front loaded.

The Modified Average Utilisation Factor (MAUF): MAUF gives an indication of the resource constrainedness. In single-project literature, several measures exist to describe resource constrainedness (Resource Strength, Resource Constrainedness,...). Demeulemeester et al. (2003) provide an overview and discuss their strengths and weaknesses. Davis (1975) was the first to introduce a multi-project measure for resource

constrainedness. He created the utilisation factor UF, which calculates the resource requirement per time unit based on the unconstrained earliest start schedule and divides it by the available resources per time unit. Kurtulus and Davis (1982) proposed to average the utilisation factor over time intervals in order to reduce the computational effort. Browning and Yassine (2010b) noted that this measure may be biased because the intervals could be of different length. To resolve this issue, they proposed to use unit intervals for the calculations. This approach is equivalent to the original definition of Davis (1975), as it measures the demand per time unit and compares it to the available resources. The original argumentation for averaging over time intervals becomes obsolete with modern computing power. As MAUF and UF are equivalent, we will abbreviate it for the remainder of the document to the measure UF.

The Variance of UF (σ^2_{UF}): As each resource type k may have a different UF_k , Browning and Yassine (2010b) proposed to measure the variation of UF over the different resource types. They define UF as the maximum utilisation factor over all resource types and σ^2_{UF} as the variance from this maximum:

$$\sigma^2_{UF} = \frac{\sum_{k \in K} (UF - UF_k)^2}{|K|} \quad (10)$$

If $\sigma^2_{UF} > 0$, BY10 set $UF_1 = UF$ and

$$UF_k = UF - \sqrt{\sigma^2_{UF}}, \quad \forall k > 1 \quad (11)$$

This results in resource type 1 being the most constrained and all others less but equally constrained. However, a multitude of different utilisation factors may lead to the same σ^2_{UF} , so only a small subset of all possible resource availabilities is generated when Equation 11 is used. Therefore, we propose an adapted approach in which resource type $k = 1$ is again the most constrained but $UF_k, \forall k > 1$ can have values that differ from those in Equation 11.

2.4 Existing benchmark sets

In the multi-project literature, four datasets have been used for benchmarking algorithms. First, Homberger (2007) created the multi-project scheduling problem library (MPSPLIB), consisting of 80 instances for the decentralised RCMPSP with completely global resources, i.e. each project has access to each resource type. The author combined single-project instances from PSPLIB (Kolisch and Sprecher 1997) into portfolios and assigned a release date to the constituent projects. For each of the instances, the portfolio resource measure AUF is also reported. This dataset was extended with 60 instances similar to the original ones, but having a combination of global and local resources (Homberger 2012). A local resource type is only accessible by one of the projects in the portfolio, while a global resource is accessible by all projects. Second, Browning and Yassine (2010a) proposed a new generation procedure for the RCMPSP and a corresponding dataset of 12,320 instances. The procedure generates single-project networks with a desired value for the network complexity measure C_j and then combines them into one project portfolio. Next, the resource profile of the portfolio is set such that it meets the target values for three resource-related portfolio measures (NARLF, MAUF and σ_{MAUF}^2). We will call this dataset BY10. Third, an open library called RCMPSPLIB was created by Vázquez et al. (2015), containing 14 instances constructed from single projects in PSPLIB, 7 instances from the generator of BY10 and 5 instances generated by the authors (without specified parameter values). The fourth and last dataset was created for the MISTA 2013 challenge, addressing the multi-mode RCMPSP (Wauters et al. 2016). The authors combined single-project multi-mode instances from PSPLIB to create 30 multi-project instances, included release dates and made the distinction between global and local resources.

Tables 3 and 4 provide an overview of the previously discussed benchmark datasets and the datasets used in the literature, respectively. The column ‘#Inst.’ indicates the number of instances in the dataset, while column $|J|$ reports the minimum and maximum number of projects per portfolio and $|I_j|$ denotes the minimum and maximum number of activities per project. ‘Origin’ denotes how the single-project networks were obtained, while ‘Network’ and ‘Resources’

indicate which network and resource measures were used to generate these projects. In Table 3, we separate these measures on the portfolio and project level, while in Table 4 the portfolio measures are bold-faced. Note that not every dataset explicitly reports the values of the single-project measures for each instance. The column ‘Local’ indicates whether local resources are included in the instances (Y) or not (N). In Table 4, the last column indicates which extension of the RCMPSP is addressed. All network and resource measures that occur in this paper are explained in “Appendix”.

In Table 4, we have summarised the literature of solution procedures for the RCMPSP. We classify the papers in three classes, based on two criteria labelled as reproducibility and comparability. We define a dataset to be *reproducible* when it is (1) created with a multi-project generator, (2) created by combining instances from single-project libraries or generators or (3) generated by the authors with a custom procedure and both network and resource measures are reported. A dataset is *comparable* when it is freely available and as such it is possible to benchmark the results of different algorithms on this dataset. The data in the first class are neither reproducible nor comparable as the algorithms are tested on non-standard instances (which we label NS in the table). These data are either created without in-depth documentation on their generation or obtained from real life cases. Although these datasets were relevant in the research studies mentioned in the table, they cannot be used in future studies making comparisons harder or even impossible. The second class contains the literature where the procedures are tested on reproducible data. For data in this class, it is possible for the reader to obtain instances that are similar, but not exactly the same, to those proposed in the paper. The third class contains papers that use comparable data. The literature in this class uses one of the libraries from Table 3. This approach is the most favourable because the instances are specifically designed for the multi-project context, and the results of different authors can be compared on the same benchmark. Note that for the papers that use comparable data, we only report which dataset is used, as the other information can be found in Table 3. This summary table exhibits that there is a trend towards reproducibility and comparability of the data, facilitating the comparison of results. 58% and 32% of the literature published since 2010 use a com-

Table 3 Summary information datasets

Dataset	#Inst.	$ J $	$ I_j $	Project level			Portfolio level	
				Origin	Network	Resources	Resources	Local
MPSPLIB	140	[2,20]	[30, 120]	PSPLIB	NC	RF	AUF	N(80), Y(60)
BY10	12,320	[3,3]	[20,20]	New	C_j	–	NARLF, MAUF, σ_{MAUF}^2	N
RCMPSPLIB	26	[2,10]	[10, 120]	PSPLIB, BY10, New	NC, C_j	RF	NARLF, MAUF, σ_{MAUF}^2	N
MISTA	30	[2,20]	[10, 30]	PSPLIB	NC	RF	–	Y

Table 4 Overview of data in the literature

Reference	#Inst.	J	J _i	Origin	Network	Resources	RCMSPSP ^a
<i>Non-standard</i>							
Chen and Shahandashiti (2009)	2	3	[21,46]	NS	–	–	Basic
Chiu and Tsai (2002)	8938	[2,4]	–	NS	–	–	DCF
Deckro et al. (1991)	1	8	[11,20]	NS	–	–	Basic
Kumanan et al. (2006)	1	5	[6,12]	NS	–	–	Basic
Kurtulus and Davis (1982)	77	[3,5]	–	NS	–	ARLF, AUF	Basic
Kurtulus (1985)	77	[3,5]	–	NS	–	ARLF, AUF	Basic
Lawrence and Morton (1993)	160	5	[25,50]	NS	OS	–	Basic
Singh (2014)	1	5	8	NS	–	–	Basic
Vercellis (1994)	–	[5,10]	[10,20]	NS	–	–	MM, NPV
Wang et al. (2015)	18	[5,30]	10	NS	–	Load	STO, DYN
<i>Reproducible</i>							
Beşikci et al. (2013)	60	6	[20,30]	PSPPLIB	NC	MUF	MM, RD
Beşikci et al. (2015)	80	6	[20,30]	PSPPLIB	NC	MUF	MM, RD
Chakraborty et al. (2017)	77	3	20	Generator BY10	C _j	NARLF, MAUF, σ ² _{MAUF}	Basic
Confessore et al. (2007)	240	[3,5]	[10,18]	ProGen	NC	RF	DEC
Dumond and Mabert (1988)	1	250	[7,50]	Patterson	–	–	DYN
Gonçalves et al. (2008)	120	[10,50]	120	PSPPLIB	NC	–	Basic
Kröger and Scholl (2009)	100	5	[7,50]	Patterson	–	–	RT
Kröger and Scholl (2010)	100	5	[7,50]	Patterson	–	–	RT
Lee et al. (2003)	100	3	[6,15]	ProGen	NC	RF	DEC
Lova et al. (2000)	32	[4,8]	[30,60]	NS	NC, NPL	RF, AUF	Basic
Lova and Tormos (2001)	32	[4,8]	[30,60]	NS	NC, NPL	RF, AUF	Basic
Mittal and Kanda (2009)	64	[3,6]	[30,60]	PSPPLIB	NC	RF, AUF	Basic
Wang et al. (2017)	420	3	30	RanGen2	SP	NARLF, MAUF, σ ² _{MAUF}	STO
Yang and Sum (1993)	16	500	[6,49]	Patterson	–	–	DEC, DYN, RT
Yang and Sum (1997)	10	500	[6,49]	Patterson	–	–	DEC, DYN, RT
Zheng et al. (2013)	10000	3	30	RanGen	OS	RC	STO
<i>Comparable</i>							
Adhau et al. (2012)				MPSPLIB			DEC
Adhau et al. (2013)				MPSPLIB			DEC, RT
Hombberger (2007)				MPSPLIB			DEC
Hombberger (2012)				MPSPLIB			DEC
Song et al. (2017)				MPSPLIB			DEC
Wauters et al. (2015)				MPSPLIB			DEC
Browning and Yassine (2010b)				BY10			Basic
Vázquez et al. (2015)				RCMPSPLIB			Basic
Pérez et al. (2016)				RCMPSPLIB			Basic
Asta et al. (2016)				MISTA			MM
Geiger (2017)				MISTA			MM
Toffolo et al. (2016)				MISTA			MM

a NPV: net present value, DCF: discounted cash flows, MM: multi-mode, STO: stochastic, RT: resource dedication, DEC: decentralised, DYN: dynamic, RT: resource transfers

parable or reproducible dataset, respectively, which indicates that the practice of using standardised data is becoming more widespread in multi-project literature.

Despite these fragmentary efforts to create multi-project benchmark data, not many authors have focused on a standardised generation approach with summary measures that are specific to the RCMPSP. However, the random generator and benchmark dataset of Browning and Yassine (2010a) are very important steps in this direction. Their generation procedure has been our main inspiration to initially look at all the data for the RCMPSP, which resulted in some ideas for improvements. In Sect. 4, we adapt this procedure and propose a new dataset as a complementary alternative to the incumbent sets.

3 Decoupled schedule generation schemes

In multi-project scheduling literature, one of the most prevalent scheduling techniques is priority rule-based heuristics, because they have a number of advantages (Browning and Yassine 2010b). First, they are relevant for practitioners due to their ease of implementation. Second, they can be used as building blocks for more advanced solution methods. Third, as their computation speed is very high, they are also applicable to large instances. For these reasons, several authors have used priority rules to solve the RCMPSP. For the description of priority rules used in this paper, we refer to Table 6. Kurtulus and Davis (1982) analysed the performance of 9 different priority rules for the RCMPSP. The authors distinguish between two ways to solve the RCMPSP: the multi-project or single-project approach. The former treats the projects individually (e.g. each project has its own critical path), while the latter combines all networks into one super-project, losing information on its constituent projects. They concluded that priority rules that incorporate both project and activity information (i.e. the multi-project approach) give better results. Kurtulus (1985) extended this study to multi-projects with unequal delay penalties, incorporating due date and penalty information in some priority rules. Dumond and Mabert (1988) investigated the extension to due dates with varying degrees of negotiating power of the portfolio manager. Lawrence and Morton (1993) propose a weighted, slack-based priority rule that incorporates due dates and compare their heuristic with 20 priority rules from the literature. Lova et al. (2000) propose a heuristic method that lexicographically optimises time-related and non-time-related objectives. For the time-related objective, they use two priority rules: MINLFT and SASP. Lova and Tormos (2001) propose an objective function for both the multi-project (mean project delay) as the single-project approach (multi-project duration increase, which is equivalent to PDEL) and analyse the performance of 7 pri-

ority rules on both objectives. They conclude that for a multi-project objective, priority rules that use activity and project information (i.e. the multi-project approach) give the best results. Browning and Yassine (2010b) were the first to perform a systematic experiment with 20 priority rules from the literature on a dataset that is generated to cover a range of different parameters. They conclude that TWK-LST and SASP perform well for the multi-project objective APD, while MINWCS performs best for the single-project objective PDEL and that these results are robust over different parameter values. Vázquez et al. (2015) propose a learning algorithm that starts with an initial set of 34 priority rules and generates a set of project priority lists. From this set, the algorithm selects the best priority rule and a tie breaker for the chosen instance. The best priority rule and corresponding project priority lists are then used as input for a genetic algorithm.

The previous paragraph shows that several researchers have advocated the usefulness of the multi-project approach and implemented this with new priority rules that take into account both project and activity information. However, these new rules are still often applied with the single-project serial and parallel schedule generation scheme. Only a few authors so far have separated the project and activity selection decisions. In our manuscript, we will call this separation of the project and the activity selection decision *decoupled scheduling*. Note that for decoupled priority rules, we use the notation $X-Y$, where X refers to the project priority rule and Y to the activity priority rule.

To the best of our knowledge, Lova and Tormos (2001) were the first to introduce decoupled priority rules. They use TWK (called MAXWK in our study) as project rule and test 4 activity rules. These decoupled rules all outperformed the original priority rules. Two of them (TWK-EST and TWK-LST) were used in other studies (Browning and Yassine 2010b; Wang et al. 2017), where they also performed well. Mittal and Kanda (2009) created a decoupled serial schedule generation scheme and compared it to existing priority rules on 64 instances. They conclude that the decoupled variant, which uses multi-project information, outperforms the existing heuristics for both APD and PDEL. This contradicts the findings of Lova and Tormos (2001) and Browning and Yassine (2010b) who found that for PDEL, the single-project rules outperform rules that incorporate multi-project information. In our computational experiment (Sect. 5), we will reconcile these conflicting observations. Zheng et al. (2013) created a genetic algorithm with an activity list where all activities of a project need to form a contiguous block in the list. Vázquez et al. (2015) create a serial schedule generation scheme where the priority list is filled with project indices. At each iteration, the project index determines which project will be considered. From this project, the eligible activity with the highest priority is scheduled. Asta et al.

Table 5 Additional notation

Notation	Meaning
ES_{ij}, LS_{ij}	The earliest and latest start time of activity a_{ij} according to the critical path schedule.
LF_{ij}	The latest finish time of a_{ij} according to the critical path schedule.
E_{ij}	The earliest precedence and resource feasible start time for a_{ij} in the current partial schedule.
SLK_{ij}	Slack of a_{ij} in the current partial schedule: $LS_{ij} - \max(ES_{ij}, E_{ij})$.
W_{ij}	The work content of a_{ij} : $\sum_{k \in K} r_{ijk} d_{ij}$.
AS_j	The set of activities of project j that are already added to the partial schedule.
US_j	The set of unscheduled activities of project j .
EP	The set containing all pairs of eligible activities for the current partial schedule.
$E_{ij,kl}$	The earliest precedence and resource feasible start time for a_{ij} if first a_{kl} and then a_{ij} are scheduled.
S_{ij}	The set of direct successors of a_{ij} .
RCP_j	The remaining critical path length of project j where $d_{ij} = 0$ for all completed activities, $d_{ij} = d_{ij} - (t - s_{ij})$ for activities that are in execution at schedule time t and d_{ij} is unchanged for $a_{ij} \in US_j$. s_{ij} is the start time of a_{ij} .
EDL_j	The expected delay: $MINDUR_j - DUE_j$. $MINDUR_j$ is the finish time of project j when all activities $a_{ij} \in US_j$ are started as soon as possible in the partial schedule, neglecting resource constraints. DUE_j is the due date of project j , in this study the finish time according to the critical path calculations

(2016) created a hyper-heuristic for the multi-mode resource-constrained multi-project scheduling problem (MRCMPSP), which won the MISTA 2013 challenge (Wauters et al. 2016). They keenly observe that when the objective is to minimise the sum or average of project completion times, good solutions often exhibit an approximate ordering of projects. To fully exploit this insight, they created neighbourhood operators that act on a complete project, in addition to operators that act on individual activities. The previous examples show that decoupling the activity and project selection may be a promising direction when the objective function is a sum over projects. To extend this research, we create decoupled extensions of the single-project schedule generation schemes and introduce new decoupled scheduling rules.

We divide priority rule heuristics for the RCMPSP into three classes: single-project, coupled and decoupled heuristics. *Single-project heuristics* originate from RCPSP research; these priority rules only use information that is related to the activities. The rules are combined with the single-project serial or parallel schedule generation scheme to create a schedule. This approach reduces a multi-project instance to a single project and does not take into account any project-specific information.

Coupled heuristics consist of a single-project SGS with a priority rule that combines activity and project information. Previous research has shown that incorporating both activity and project information in priority rules can outperform single-project priority rules, certainly when the objective function incorporates project-specific information (Browning and Yassine 2010b). Although these heuristics improve upon their single-project variants, the SGS still treats the multi-project as one super-project network. The eligible

activities of all projects are collected in one set, and no distinction is made between activities from different projects.

Decoupled heuristics resolve this issue by separating the eligible activities in different sets, one per project. The resulting SGS' decouple the project and activity prioritisation decisions. In the first step, the project with the highest priority is selected and in the second step the activity with the highest priority from the selected project. In our literature review, we established that previous research has indicated the value of this approach (Lova and Tormos 2001; Mittal and Kanda 2009). In our computational experiment, we extend the existing body of knowledge by comparing the performance of the decoupled schemes with the single-project and coupled scheduling schemes. The additional notation required for the calculation of the priority rules is provided in Table 5. Table 6 lists the priority rules used in our experiments, classified in the three categories. Note that for the heuristics in the third category, both activity and project priority rules are provided as the scheduling scheme makes two selection decisions. In the remainder of this section, we provide a description of the parallel and serial decoupled schedule generation schemes.

3.1 The parallel decoupled SGS

The parallel decoupled SGS (PSGSd) is a time-incrementing scheme; its pseudocode is shown in Algorithm 1. The difference with the single-project parallel SGS is that in every iteration it first prioritises the projects according to their project priority and then iterates over their eligible sets in that order. The eligible set of a project is denoted by \mathcal{E}_j . First, the time is initialised to $t = 0$. At the beginning of every iteration, the procedure schedules the dummy start activities of

Table 6 Priority rules used in this study

Code	Name		Description	
<i>Single-project rules</i>				
MINEST	Minimum earliest start time		$\min_{i,j}(ES_{ij})$	
MAXEST	Maximum earliest start time		$\max_{i,j}(ES_{ij})$	
MINLST	Minimum latest start time		$\min_{i,j}(LS_{ij})$	
MAXLST	Maximum latest start time		$\max_{i,j}(LS_{ij})$	
MINSLKd	Minimum slack (dynamic)		$\min_{i,j}(SLK_{ij})$	
MAXSLKd	Maximum slack (dynamic)		$\max_{i,j}(SLK_{ij})$	
SPT	Shortest processing time		$\min_{i,j}(d_{ij})$	
LPT	Longest processing time		$\max_{i,j}(d_{ij})$	
MINWCS	Minimum worst case slack		$\min_{i,j}(LS_{ij} - \max_{akl:(a_{ij},a_{kl}) \in EP} E_{ij,kl})$	
MAXSUC	Maximum direct successors		$\max_{i,j}(S_{ij})$	
FCFS	First come first served		The activity that arrived first in the eligible set.	
<i>Coupled rules</i>				
SASP	Shortest activity of shortest project		$\min_{i,j}(d_{ij} + CP_j)$	
LALP	Longest activity of longest project		$\max_{i,j}(d_{ij} + CP_j)$	
MINTWK	Minimum total work content		$\min_{i,j}(W_{ij} + \sum_{i \in AS_j} W_{ij})$	
MAXTWK	Maximum total work content		$\max_{i,j}(W_{ij} + \sum_{i \in AS_j} W_{ij})$	
<i>Decoupled rules</i>				
Project selection rules			Activity selection rules	
MAXWK ^{a,b}	Work content	$\max_j(\sum_{i \in AS_j} W_{ij})$	MINEST ^{a,b} Earliest start	$\min_i(ES_{ij})$
MINWK ^b	Work content	$\min_j(\sum_{i \in AS_j} W_{ij})$	MAXEST Earliest start	$\max_i(ES_{ij})$
MAXTWK	Total work content	$\max_j(\sum_{i \in I_j} W_{ij})$	MINLST ^{a,b} Latest start	$\min_i(LS_{ij})$
MINTWK	Total work content	$\min_j(\sum_{i \in I_j} W_{ij})$	MAXLST Latest start	$\max_i(LS_{ij})$
MINCP ^b	CP-length	$\min_j(CP_j)$	MINLFT ^b Latest finish	$\min_i(LF_{ij})$
MAXCP ^b	CP-length	$\max_j(CP_j)$	MINSLKd ^b Slack (dynamic)	$\min_i(SLK_{ij})$
MAXTWR ^b	Total work remaining	$\max_j(\sum_{i \in US_j} W_{ij})$	MAXSLKd Slack (dynamic)	$\max_i(SLK_{ij})$
MINTWR ^b	Total work remaining	$\min_j(\sum_{i \in US_j} W_{ij})$	MINSLKs Slack (static)	$\min_i(LS_{ij} - ES_{ij})$
MAXSP	SP value	$\max_j(SP_j)$	MAXSLKs Slack (static)	$\max_i(LS_{ij} - ES_{ij})$
MINSP	SP value	$\min_j(SP_j)$	SPT ^b Processing time	$\min_i(d_{ij})$
MAXACT	Number of activities	$\max_j(I_j)$	LPT Processing time	$\max_i(d_{ij})$

Table 6 continued

Code	Name		Description		
MINACT	Number of activities	$\min_j(I_j)$	MINWK ^b	Work content	$\min_i(W_{ij})$
MINRCP ^b	Remaining CP-length	$\min_j(RCP_j)$	MAXWK ^b	Work content	$\max_i(W_{ij})$
MAXRC ^b	Remaining CP-length	$\max_j(RCP_j)$	MAXSUC	Number of successors	$\max_i(S_{ij})$
MINEDL	Expected delay	$\min_j(EDL_j)$	FCFS ^{a,b}	First come, first served	
MAXEDL	Expected delay	$\max_j(EDL_j)$	MINWCS	Worst case slack	$\min_i(LS_{ij} - \max_{a_{kl}:(a_{ij},a_{kl}) \in EP} E_{ij,kl})$

^a Occurs in Lova and Tormos (2001)

^b Occurs in Mittal and Kanda (2009)

all projects that are not started yet and for which the release date is equal to t . Then, it runs through the sorted project list and per project runs through the sorted eligible activity list. It schedules the activities for which there are enough resources available at the current time. At the end of the iteration, it updates the eligible set if necessary. Then, it increments t to the minimum of the earliest release date of the unstarted projects and the earliest finish time greater than t and frees the resources of all activities that finish at time t . The algorithm repeats these steps until the counter c is equal to s , the total number of activities in the multi-project.

Algorithm 1: Decoupled parallel SGS

```

1  $c \leftarrow 0; t \leftarrow 0$ 
2 for  $j \in J$  do
3    $\mathcal{E}_j \leftarrow \{a_{0j}\}$ 
4 while  $c < s$  do
5   for  $j \in J$  do
6     if  $a_{0j} \in \mathcal{E}_j$  and  $ES_{0j} \geq t$  then
7        $Schedule(a_{0j}); c \leftarrow c + 1$ 
8    $P \leftarrow Prioritise(J)$ 
9   for  $j \in P$  do
10     $A_j \leftarrow Prioritise(\mathcal{E}_j)$ 
11    for  $a_{ij} \in A_j$  do
12      if  $CheckResources(a_{ij})$  then
13         $Schedule(a_{ij}); c \leftarrow c + 1$ 
14   $Release()$ 
15   $t \leftarrow IncrementTime()$ 

```

3.2 The serial decoupled SGS

The serial decoupled SGS (SSGSd) is an activity-incrementing scheme; the general outline is shown in Algorithm 2. Again, it differs from its single-project variant because at every iteration it first selects the highest priority project and from that project the eligible activity with the highest priority. The algorithm first schedules the dummy start activities of all

projects at their earliest start times, i.e. the release date of their respective projects. Then, it iteratively schedules the activity with the highest priority from the project with the highest priority to its earliest precedence and resource feasible start time. Once the activity is scheduled, the eligible set is updated if necessary. Again, this is repeated until all activities are scheduled (i.e. $c = s$).

Algorithm 2: Decoupled serial SGS

```

1  $c \leftarrow 0; t \leftarrow 0$ 
2 for  $j \in J$  do
3    $\mathcal{E}_j \leftarrow \{a_{0j}\}$ 
4    $Schedule(a_{0j}, ES_{0j}); c \leftarrow c + 1$ 
5 while  $c < s$  do
6    $j \leftarrow Prioritise(J)$ 
7    $a_{ij} \leftarrow Prioritise(\mathcal{E}_j)$ 
8    $t \leftarrow FindWindow(a_{ij})$ 
9    $Schedule(a_{ij}, t); c \leftarrow c + 1$ 

```

4 Data generation

In this section, we will generate new datasets for our computational experiments. In Sect. 4.1, we reimplement the generation procedure of BY10 with the minor adaptations of Sect. 2. In Sect. 4.2, we evaluate the procedure and compare it with the existing datasets. After this analysis, we will propose a new benchmark dataset using the modified generation procedure.

4.1 Reimplementation of generation procedure

We have reimplemented the generation algorithm from BY10 with three adaptations. First, Sect. 2 shows that the networks in the dataset BY10 were all rather parallel. In order to obtain a more diverse set of networks, we used RanGen2 because it was shown that this generator is able to generate a wide variety of networks with different properties (Vanhoucke et al. 2008). We systematically created networks with SP values

in the interval of [0.1, 0.9] in order to cover the most important range of network types. The networks are divided in 3 classes of seriality: L (low: $SP \in \{0.1, 0.2, 0.3\}$), M (medium: $SP \in \{0.4, 0.5, 0.6\}$) and H (high: $SP \in \{0.7, 0.8, 0.9\}$). Networks with SP values outside this interval are rather extreme cases, as they are close to the completely serial or parallel network. Similar to Browning and Yassine (2010b), we denote the network structure of a portfolio by a vector of the three categories. For instance, 8L-2M-2H denotes that the portfolio contains 8 projects in the low category, 2 in the medium category and 2 in the high category. Second, we generated instances with the adapted definition of $NARLF'$. The algorithm to obtain this value is the same as reported in BY10, but now works with $NARLF'$ instead of $NARLF$. Third, σ_{UF}^2 was generated in a more general manner. With this adaptation, the algorithm can generate different UF_k values for each resource type, while still meeting the target σ_{UF}^2 . More precisely, UF_1 is set to the maximum value UF , while UF_k for $k > 1$ is sampled from the range $[LB_\sigma, UF]$. LB_σ is the lower bound on UF_k for all resource types given a desired σ_{UF}^2 , while UF is the upper bound by definition. LB_σ can be calculated as follows. Take the extreme case where $UF_k = UF_1$ for all resource types except one arbitrary type, we take type 2 for illustrative purposes. It follows that only UF_2 contributes to σ_{UF}^2 . Isolating UF_2 in Equation 10 results in:

$$UF_2 = UF_1 \pm \sqrt{\sigma_{UF}^2 \cdot |K|}. \tag{12}$$

σ_{UF}^2 is a variance from the maximum, so $UF_k \leq UF_1, \forall k$. As a consequence, Equation 12 reduces to

$$UF_2 = LB_\sigma = UF_1 - \sqrt{\sigma_{UF}^2 \cdot |K|}. \tag{13}$$

If UF_2 would be lower than this LB_σ , σ_{UF}^2 would exceed the desired variance, independent of the other UF_k values. Because Eq. 13 may result in negative values, we set a minimum value for $UF_k > 0.1$.

The generation algorithm consists of three steps, which are shown in Algorithms 3, 4 and 5. First, the single projects with the desired $SP_j, |I_j|$ and $|K|$ are generated (Algorithm 3). For each activity a_{ij} , the demand for each resource type is randomly sampled from the interval $[1, R_k]$. In the second step (Algorithm 4), the algorithm iteratively changes the resource demand of random activities until the desired $NARLF'$ is obtained. This means that when the target $NARLF'$ is higher than the current value, the algorithm will decrease (increase) the demand of activities that lie in the first (second) half

of CP according to the earliest start schedule, where ES_{ij} indicates the earliest start time of activity a_{ij} . The algorithm returns a failure if the target $NARLF'$ was not obtained after $maxIt$ iterations. In the third step (Algorithm 5), the resource availabilities R_k are generated such that the resulting UF and σ_{UF}^2 meet the desired values. Resource type 1 is set to be the most constrained, and the others are randomly sampled from $[LB_\sigma, UF_{des}]$. As long as the difference between the resulting σ_{UF}^2 and the target exceeds a small value (β), the algorithm increments or decrements UF_k values, within the interval $[\max(LB_\sigma, 0.1), UF]$. Once a feasible set of UF values is obtained, the R_k values are set such that the desired UF is obtained. There are two situations that cause the algorithm to return a failure. First, for higher UF_k it is possible that the resulting R_k value for a resource type is smaller than the requirement for that type of at least one activity, resulting in an infeasible instance. Second, due to rounding, the difference between the actual UF_k and the desired UF can exceed a threshold value γ .

Algorithm 3: Read single projects

```

1 Create an empty multi-project
2 for j ∈ J do
3   Add a single project with desired SPj, |Ij| and |K|
4   Randomise resource demand

```

Algorithm 4: tuneNARLF($NARLF'_{des}$)

```

1 c ← 0
2 while |NARLF' - NARLF'_{des}| > α do
3   if c ≥ maxIt then
4     return failure
5   else
6     Select random activity aij
7     Select random resource type k ∈ Hij
8     Calculate NARLF'
9     if NARLF' < NARLF'_{des} then
10      if ESij + ⌈ $\frac{d_{ij}}{2}$ ⌉ ≤ min rj + ⌈ $\frac{C_j^p}{2}$ ⌉ then
11        rijk ← rijk - 1
12      else
13        rijk ← rijk + 1
14    else
15      if ESij + ⌈ $\frac{d_{ij}}{2}$ ⌉ ≤ min rj + ⌈ $\frac{C_j^p}{2}$ ⌉ then
16        rijk ← rijk + 1
17      else
18        rijk ← rijk - 1
19    c ← c + 1
20 return success

```

Algorithm 5: tuneUF($UF_{des}, \sigma_{UF,des}^2$)

```

1 if  $\sigma_{UF,des}^2 = 0$  then
2   Set all  $UF_k \leftarrow UF_{des}$ 
3 else
4   Set  $UF_1 \leftarrow UF_{des}$ 
5   for  $k > 1$  do
6      $UF_k \leftarrow \text{rand}(LB_\sigma, UF_{des})$ 
7     while  $|\sigma_{UF}^2 - \sigma_{UF,des}^2| > \beta$  do
8       Calculate  $\sigma_{UF}^2$ 
9       Select a random resource type  $k > 1$ 
10      if  $\sigma_{UF}^2 > \sigma_{UF,des}^2$  then
11         $UF_k \leftarrow UF_k + 0.05$ 
12      else
13         $UF_k \leftarrow UF_k - 0.05$ 
14 for  $k \in K$  do
15    $R_k = \text{round}(\frac{\sum_{j \in J} \sum_{i \in I_j} r_{ijk} d_{ij}}{CP_{\max} \cdot UF_k})$ 
16   if  $R_k < \max_{i,j} (r_{ijk})$  or  $|\frac{W_k}{CP_{\max} \cdot R_k} - UF_k| > \gamma$  then
17     return failure
18 return success

```

4.2 Evaluation of the generation procedure

In order to evaluate the modified generation procedure, we have generated a large set of instances and compared them with the existing datasets (we exclude the MISTA set because it is designed for the multi-mode variant). In order to select appropriate values for NARLF', UF and σ_{UF}^2 we used the extreme values found in the existing datasets. This resulted in the intervals $[-30, 2]$, $[0.25, 9.25]$ and $[0, 6]$ for NARLF', UF and σ_{UF}^2 , respectively. Because most of the instances in the incumbent datasets are rather parallel, we chose 7 different combinations of L, M and H such that three types of portfolios are obtained. In type 1, all projects are in the same category, in type 2 one category is predominantly present, and in type 3 each category is equally represented in the portfolio. We generated instances with 6, 12 and 24 projects per instance. Figure 2 shows the two-dimensional parameter plots for the three existing datasets and the result of the generation procedure for the three portfolio sizes.

The plots allow us to make several observations. First of all, we observe that the parameter space obtained by the generation procedure encloses the three existing datasets, i.e. it can create instances that are similar to those in the datasets. Second, the generator obtains parameter combinations that are not present in any of the datasets. RCMPSPLIB and MPSPLIB approximately cover the complete NARLF' range, but most of their instances have a low σ_{UF}^2 . The BY10 dataset, on the other hand, covers a wider variety of σ_{UF}^2 values, but does not contain instances with NARLF' higher than -7.5. The generator can create instances on the whole

spectrum for the three measures. Third, the range of feasible NARLF' and UF values depends on the number of projects in the portfolio. When the number of projects in the portfolio $|J|$ increases, the minimal obtainable NARLF' value increases. When the portfolio size increases, the effect of a project with an extreme NARLF' is averaged over a larger amount of projects. This reduces the effect of individual projects, leading to less extreme values. Moreover, higher UF values are more likely to be generated with increasing $|J|$ because adding new projects to the portfolio increases the total resource demand and thus the UF.

Although Fig. 2 suggests that the procedure can generate instances over the whole spectrum, this is not the case for every parameter combination. We detected two parameter dependencies. First, the range of feasible NARLF' values depends on the SP vector. Only for instances where all projects are in the low-SP category, the whole range of NARLF' values can be obtained. The more projects in the high category, the smaller the feasible NARLF' range. A second dependency was found between the SP vector and UF. For the case with 6 projects, UF values higher than 4.25 are practically only obtainable for portfolios where all projects have a low SP value. These instances have on average a lower CP_{\max} than those with a higher SP value. As CP_{\max} is in the denominator of the UF formula, this explains why more parallel instances can have a higher UF.

Furthermore, we observed that the procedure may generate instances with unrealistic resource profiles. For instance, for high NARLF' values, the procedure may shift a large amount of demand to activities in the second half of the portfolio, while reducing the demand of activities in the first half to very low values. This leads to distorted demand profiles as activities may have extremely high or low demands. In order to obtain a quantifiable measurement of this distortion, we rely on the Gini coefficient, which is used in economics to measure the inequality of e.g. wealth distribution. In this context, it measures how the total work content ($\sum_{j \in J} \sum_{i \in I_j} \sum_{k \in K} r_{ijk} \cdot d_{ij}$) is distributed among all activities. A value of 0 means that each activity has the same work content, while a value of 1 means that one activity accounts for the total resource demand and the others have no resource demand. To benchmark this distortion of resource profiles, we calculated the distribution of Gini values for instances that were randomly generated without any parameter specifications. This distribution is shown in Fig. 3 by the grey full line. The black line shows the distribution of Gini values for instances generated by our adapted procedure over the whole feasible range of parameter combinations. This second type of multi-project instances has a wider range of possible Gini values, which is in line with the existing datasets where more than 95% of the Gini values range from 0.3 to 0.55. These distributions show that by just combining single-project instances, only a subset of all possible multi-

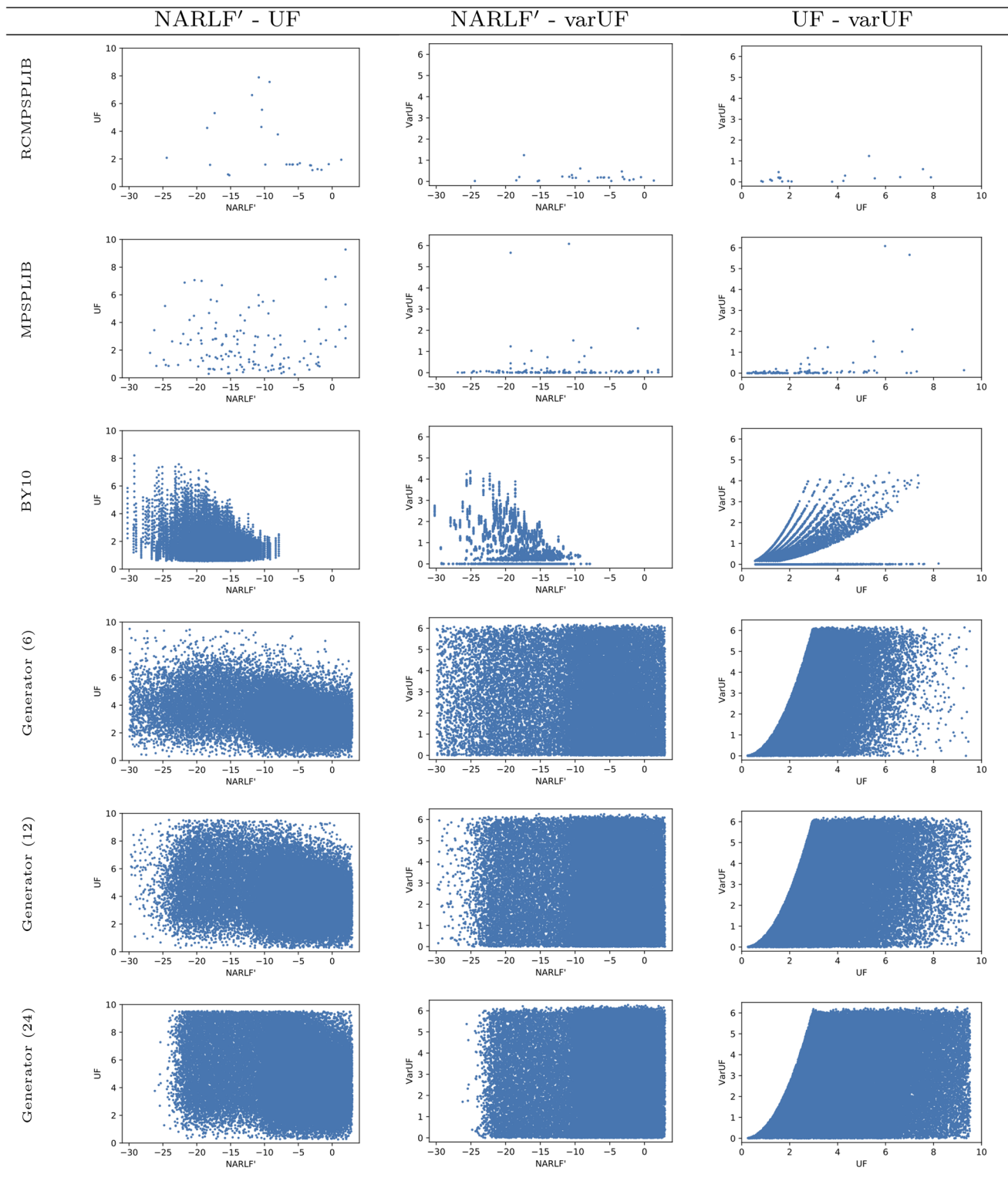


Fig. 2 The two-dimensional parameter space plots

project instances can be generated. However, higher Gini values result in more activities that have a very low resource demand which are thus less likely to trigger resource con-

flicts. As such, the resulting instances may not be realistic anymore because the scheduling problem becomes easier. In order to set a threshold for acceptable Gini values, we eval-

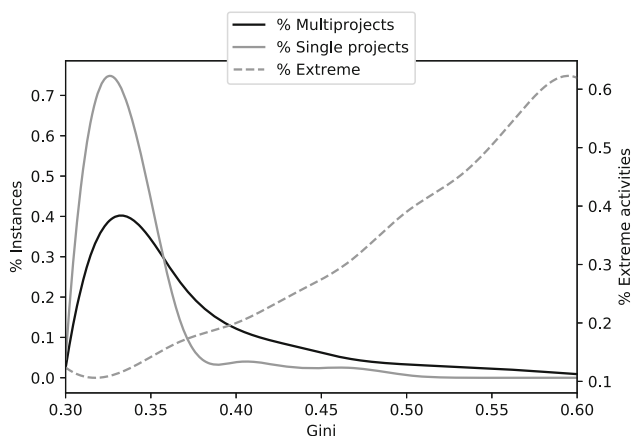


Fig. 3 Gini distribution

uated its relation with the percentage of *extreme activities*: the percentage of activities that require less than 5% of the available resources for each resource type. This relation is expressed by the dashed line in Fig. 3. We chose to keep the average percentage of extreme activities below 20%. This leads to a threshold for Gini of 0.42, allowing us to include 74.86% of all possible instances.

4.3 Dataset generation

The previous subsection showed that the feasible NARLF' and UF ranges depend on the SP vector of the portfolio, which prevents the construction of a complete orthogonal test design in which each parameter combination occurs. We restrict the NARLF' and UF values to the ranges where the procedure was able to generate instances for different SP vectors. Furthermore, the procedure only accepts instances if their Gini coefficient is less than or equal to 0.42. For each parameter combination, the procedure tried to generate one instance, terminating after 10 unsuccessful trials.

We generated three datasets (6 × 60, 12 × 60 and 24 × 60) with 6, 12 and 24 projects per instance, respectively. Each

project consists of 60 activities, resulting in 360, 720 and 1440 activities per instance for the respective datasets. All instances have 4 global resource types. Table 7 shows the ranges of parameter values for each of the datasets. Note that the NARLF' range decreases for the larger datasets, while the UF range increases. The range for σ_{UF}^2 remains constant over the three sets. The seven SP-combinations are listed in the second section of the table. Even within these restricted parameter ranges, the procedure was not able to generate every parameter combination. It could generate the least instances when all projects were in the high-SP category.

From the set of successfully generated instances, we selected a subset such that each SP vector was equally represented and that the instances were divided as equally as possible over the ranges of the three other parameters. This resulted in a selection of 833, 1463 and 2254 instances for the three respective datasets.

The incumbent and newly created datasets are available on www.projectmanagement.ugent.be. All datasets were converted to the same format, such that researchers can easily access and use the available data. This format is also explained on the website.

5 Computational results

In this section, we will first evaluate the performance of the priority rules on the existing and new datasets. Then, we will test a genetic algorithm on the datasets to evaluate whether the proposed datasets are challenging enough for more advanced algorithms.

5.1 Evaluation priority rules

We will analyse the performance of the decoupled SGS' for the objective functions APD and PDEL. We evaluated every combination of activity and project PR, which means that 256 different PR combinations are tested on both the parallel

Table 7 Parameter settings for the datasets

Parameter	6 × 60		12 × 60		24 × 60	
	Range	Increment	Range	Increment	Range	Increment
NARLF'	[-14, 0]	1	[-12, 0]	1	[-8, 0]	1
UF	[0.25, 4.25]	0.5	[0.25, 5.25]	0.5	[0.25, 7.25]	0.5
σ_{UF}^2	[0, 5]	1	[0, 5]	1	[0, 5]	1
SP Type 1	6L, 6M, 6H,		12L, 12M, 12H,		24L, 24H, 24H,	
SP Type 2	4L-M-H, L-4M-H, L-M-4H,		8L-2M-2H, 2L-8M-2H, 2L-2M-8H,		16L-4M-4H, 4L-16M-4H, 4L-4M-16H,	
SP Type 3	2L-2M-2H		4L-4M-4H		8L-8M-8H	
Design size	5,670		6,006		5,670	
Selection size	833		1,463		2,254	

Table 8 Best priority rules for APD

Existing datasets	New datasets
MINTWR-MINLST (S)	MINCP-MINSLKd (P)
MINTWR-MINLST (P)	MINCP-MINLST (S)
MINTWR-MINSLKd (P)	MINCP-MINLST (P)
MAXWK-MINLST (P)	MINCP-MINLFT (P)
MAXWK-MINSLKd (P)	MINSP-MINSLKd (P)
MINTWR-MINSLKs (S)	MINSP-MINLST (P)
MINTWR-MINSLKs (P)	MINSP-MINLST (S)
MINTWR-MINWCS (S)	MINSP-MINLFT (P)
MINTWR-MINLFT (S)	MINCP-MINEST (P)
MINTWR-MINSLKd (S)	MINCP-MINLFT (S)

and serial decoupled SGS. We will compare the performance over the different sets and analyse the impact of the different parameters on the performance.

5.1.1 APD

Table 8 reports the 10 best priority rules for the existing and new datasets. The letter between parentheses indicates whether the results were obtained with the serial (S) or parallel (P) variant of the SGS.

A first observation is that the decoupled scheduling schemes outperform the existing variants as the 10 best rules are decoupled. For the existing datasets, the decoupled rule MINTWR-MINLST gives the best results. This observation matches the conclusions of Mittal and Kanda (2009), where MINTWR combined with MINLST, MINSLKd and MINLFT scored best on the average delay. For our own generated data, the rule MINCP-MINSLKd performed best. Although the coupled rules MAXTWK and SASP do not occur in the top 10, it is worthwhile to note that they consistently exhibit a better performance than the single-project rules on our datasets. Second, we observe that most of the best performing activity selection rules are critical path based (e.g. minimum LFT, minimum slack,...) and that this is consistent over all datasets. However, for the existing datasets, work content-related project rules perform best, while for our datasets the best project rules are related to the (critical path) length of the projects. The difference in PR performance between our dataset and the others can be explained by the variation in the SP values within a portfolio. To analyse this difference, we define the SP spread of a portfolio as the difference in SP value between the most serial and the most parallel of its projects. Table 9 shows the minimum, maximum and average SP spread per dataset. Our datasets have on average a higher SP spread and a larger range of possible SP spread values. This means that most instances in the incumbent sets consist of projects with very similar

Table 9 The SP spread statistics per dataset

Dataset	Min	Avg	Max
BY10	0	0.04	0.16
MPSPLIB	0	0.04	0.13
RCMPSPLIB	0	0.04	0.23
6 × 60	0	0.45	0.82
12 × 60	0.1	0.51	0.82
24 × 60	0.2	0.54	0.82

SP values and critical path durations. It follows that due to these small differences, project priority rules that use SP-related information will not perform well and others, related to e.g. work content, are better suited. However, when the difference between the length of the project varies more, the length-related project priority rules exhibit the best performance. This result shows that our dataset contains instances for which insights from previous research are not necessarily generalisable.

Next, we evaluate which step in the algorithm has the strongest effect on the performance: the project or activity selection decision. Table 10 reports the average APD obtained over all instances in dataset 24 × 60, grouped per project or activity priority rule. For the sake of brevity, we only report the results for set 24 × 60, but the same conclusions can be drawn from the other two sets. The table shows that for APD the choice of project PR has a large impact on the performance of the decoupled SGS, while the activity PR has a smaller impact. MINCP and MINSP are the best project PR's, while MINSLKd and MINLFT are the best activity PR's.¹ This is in line with the observation of Asta et al. (2016) that high-quality solutions tend to have an approximate ordering of projects. We conjecture that the project PR will have a large impact for objective functions that average a certain value over all projects in the portfolio.

Furthermore, we can deduce from the table the choice of scheduling scheme has a relatively small impact on the performance for most of the rules. However, for certain rules the choice of SGS has a larger impact that is consistent over the three datasets. We highlighted the project (activity) rules in boldface for which one SGS outperforms the other with minimum 5% (2%) for the three datasets.

Next, we will evaluate the impact of the different problem parameters on the performance of the decoupled scheduling schemes. Figure 4 shows the average APD over each of the parameter values for the best rule MINCP-MINSLKd (P).

¹ The instances in each of our datasets have the same number of activities, so all projects are tied for the MAXACT and MINACT rules. Because we set MINCP as tie breaker, MINACT and MAXACT are equivalent to MINCP in this case. Therefore, we omit these rules here from the discussion.

Table 10 Performance grouped per project and activity PR (24×60)

Project PR	Parallel	Serial	Activity PR	Parallel	Serial
MAXWK	3.61	3.59	MINEST	5.05	4.99
MINWK	7.52	7.75	MAXEST	5.11	5.05
MAXTWK	4.84	4.93	SPT	5.09	5.06
MINTWK	4.42	4.49	LPT	5.09	4.97
MINCP	3.38	3.43	MINSLKd	4.94	4.94
MAXCP	5.97	6.06	MAXSLKd	5.04	5.06
MAXTWR	8.05	8.30	MINLST	5.03	4.92
MINTWR	4.03	4.49	MAXLST	5.13	5.07
MAXSP	5.84	5.92	MINWK	5.11	5.07
MINSP	3.39	3.45	MAXWK	5.07	4.97
MAXACT	3.38	3.43	MAXSUC	5.07	5.02
MINACT	3.38	3.43	FCFS	5.06	4.98
MINRCP	6.26	4.62	MINWCS	5.09	4.95
MAXRCP	5.66	6.03	MINSLKs	5.07	4.97
MINEDL	5.92	6.17	MAXSLKs	5.03	5.06
MAXEDL	5.27	3.94	MINLFT	4.93	4.96

Again we only show the effects for the dataset 24×60 , but the results are similar for the other two sets. The effects of NARLF' and UF are in line with the conclusions drawn by Browning and Yassine (2010b), with a lower NARLF' or a higher UF leading to a higher APD. For the network structure, we observe that the average APD is the lowest when all projects fall in the same category of SP values. When there is variation in SP values within the portfolio, it seems that the number of low seriality projects in the portfolio has the strongest effect on the APD. The last factor σ_{UF}^2 exhibits a positive correlation with APD. However, this contradicts to Browning and Yassine (2010b), where a negative correlation was observed.

Figure 5 shows the two-way interaction effects. The strongest interaction effect is observed between SP and UF. The strength of the relationship between APD and UF clearly depends on the network structure of the portfolio. A second, smaller interaction effect can be observed between UF and NARLF'. A highly negative NARLF' combined with a high UF leads to a bigger increase in APD. We conclude from the plots that there is no clear discernible interaction between NARLF' and σ_{UF}^2 or between SP and σ_{UF}^2 . Note that we do not show the NARLF'-SP and UF- σ_{UF}^2 interaction plots because these parameters depend on each other and it was not possible to generate every combination of the two parameters. As such, these plots do not bring any useful insight.

In addition to the plots, we created an OLS regression model to quantify the effect of the parameters on the performance of the scheduling schemes. A model consisting of all main effects resulted in an adjusted R^2 of 88.8%. Including the SP-UF interaction effects increased the adjusted R^2 to

94.9%. Adding other interaction effects did not increase the R^2 of the model significantly and lead to multi-collinearity problems. We also tried including the NARLF'-SP and UF- σ_{UF}^2 interactions, neither of them increased the explanatory power of the model. The regression calculated a negative coefficient for σ_{UF}^2 (i.e. -0.1215), confirming the observations of Browning and Yassine (2010b). This is probably caused by the oversimplification of a main effect plot, which only takes into account the explanatory power of one parameter. The coefficients for the other parameters are in line with the observations in the main effect plots.

5.1.2 PDEL

Now we will discuss the performance of the priority rules for the PDEL objective, which measures the makespan of the portfolio. Table 11 reports the 10 best PRs for the incumbent and the new datasets, respectively. For the existing sets, the single-project rules MINWCS, MINSLKd and MINLST form the top 3, followed by decoupled rules. This confirms the conclusions of Browning and Yassine (2010b) and Lova and Tormos (2001) that single-project rules are better when the objective is PDEL.

On the contrary, on our datasets, decoupled PR's do consistently outperform single-project rules, confirming the observations of Mittal and Kanda (2009). For the new datasets, it is worthwhile to include multi-project information in the heuristics, even though the objective function is equivalent to single-project makespan minimisation. Furthermore, there are only two well-performing project rules for our dataset: MAXTWR and MINWK. To conclude, we observe that the parallel SGS is consistently better than the serial SGS for the PDEL objective.

The impact of the project and activity selection rules is shown in Table 12, where the results are grouped per project and activity rule, respectively. We again restrict our analysis to the dataset 24×60 . Similar to the results for APD, the choice of project priority rule explains more variation than the choice of activity priority rule. However, the differences between project rules are much smaller than with APD. Because PDEL only looks at the maximum finish time over all projects, the finish time of all other projects becomes irrelevant, which explains why the impact of project priorities is stronger for APD than for PDEL. Furthermore, the parallel SGS on average outperforms the serial for all rules, which was not the case for APD.

The main effect plots for MAXTWR-MINLFT(P) over the different parameters are shown in Fig. 6. For NARLF', UF and σ_{UF}^2 , the effects have the same direction as those observed for APD. However, the impact of NARLF' on PDEL is smaller than on APD. According to the figure, the network structure has a different effect on PDEL than on APD, and the size of the effect is smaller. For 24H, the PDEL is still

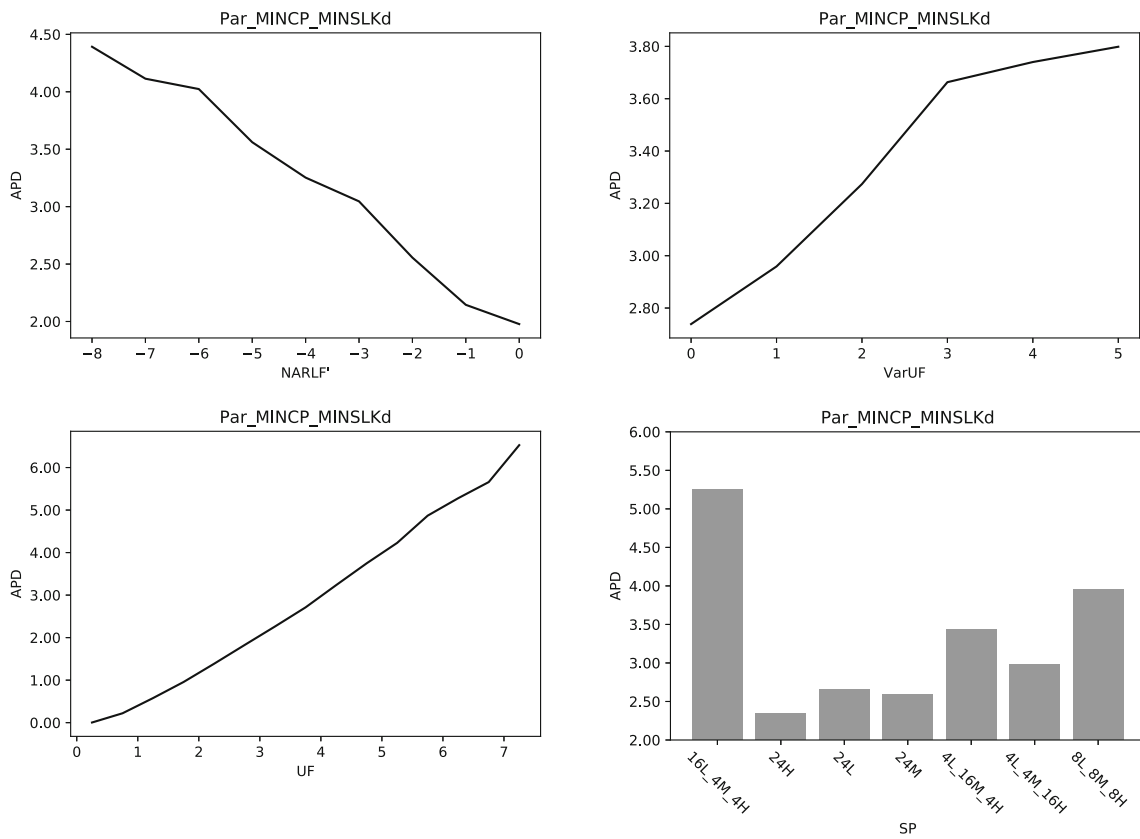


Fig. 4 Main effect plots APD

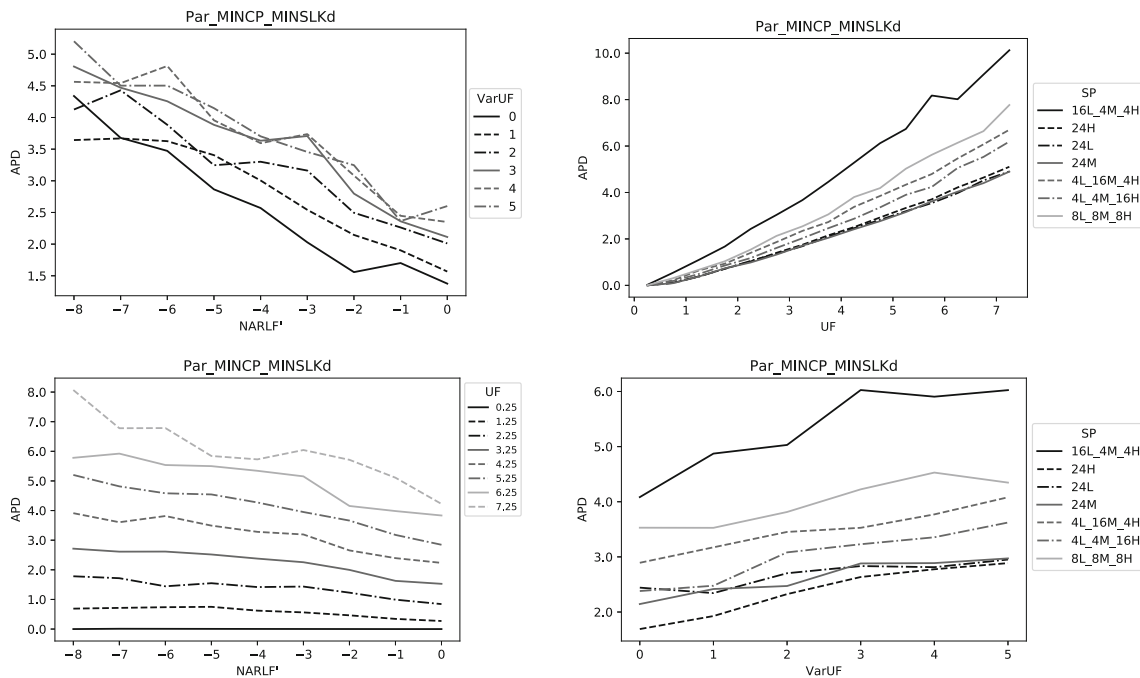


Fig. 5 Interaction effect plots APD

the lowest, but the average PDEL is higher for 24M and 24L. When there is variation in the SP vector, the number of

projects in categories M and H seems to be the driving factors behind the increase in of PDEL, as the largest increases

Table 11 Best priority rules for PDEL

Existing datasets	New datasets
MINSLKd (P)	MAXTWR-MINLFT (P)
MINLST (P)	MAXTWR-MINSLKd (P)
MINWCS (P)	MAXTWR-MINLST (P)
MAXTWR-MINSLKd (P)	MAXTWR-MINEST (P)
MAXTWR-MINLST (P)	MAXTWR-FCFS (P)
MAXEDL-MINSLKd (P)	MINWK-MINLFT (P)
MINWK-MINLST (P)	MAXTWR-MAXWK (P)
MINRCP-MINSLKd (P)	MINWK-MINSLKd (P)
MINWK-MINSLKd (P)	MINWK-MINLST (P)
MINRCP-MINLST (P)	MAXTWR-MINWCS (P)

Table 12 Average PDEL grouped per project and activity PR (24 × 60)

Project PR	Parallel	Serial	Activity PR	Parallel	Serial
MAXWK	4.10	4.52	MINEST	3.89	4.21
MINWK	3.55	3.66	MAXEST	3.91	4.24
MAXTWK	4.00	4.28	SPT	3.90	4.25
MINTWK	3.95	4.28	LPT	3.90	4.19
MINCP	4.12	4.48	MINSLKd	3.89	4.21
MAXCP	3.78	4.04	MAXSLKd	3.91	4.27
MAXTWR	3.51	3.62	MINLST	3.89	4.17
MINTWR	3.99	4.28	MAXLST	3.91	4.25
MAXSP	3.82	4.08	MINWK	3.91	4.25
MINSPP	4.12	4.48	MAXWK	3.90	4.19
MAXACT	4.12	4.48	MAXSUC	3.90	4.22
MINACT	4.12	4.48	FCFS	3.90	4.21
MINRCP	3.71	4.35	MINWCS	3.90	4.20
MAXRCP	3.81	4.11	MINSLKs	3.90	4.20
MINEDL	3.78	3.94	MAXSLKs	3.91	4.27
MAXEDL	3.95	4.46	MINLFT	3.89	4.21

are observed for the instances with 8 or 16 projects in these categories.

The interaction plots are shown in Fig. 7. The interaction effects SP-UF and NARLF'-UF that were observed for APD are absent for PDEL. Only for the SP vector equal to 24L, the increase due to UF seems to be somewhat smaller than for the other network types. Again there is no clear interaction effect between NARLF' and σ_{UF}^2 or SP and σ_{UF}^2 . These observations were confirmed by the OLS regression model that was constructed. The model consisting of only the main effects obtained an adjusted R^2 of 98.7%. None of the interaction effects increased this R^2 further.

To conclude, we summarise the insights gained. First of all, decoupled scheduling outperformed the existing scheduling schemes on all datasets for APD, but only on the new datasets for PDEL. This last result shows that even though the

objective is equivalent to single-project makespan minimisation, including multi-project information in the heuristic can improve the performance. Second, our proposed datasets contain more variety regarding the network structure of the underlying project and as a result different priority rules performed best than on the existing sets. Third, the project selection decision has the largest impact on the performance of the scheduling scheme and this effect is stronger for APD than for PDEL. Last, the impact of the different summary measures is in line with conclusions from previous research Browning and Yassine (2010b).

5.2 Metaheuristic

Although the computational experiments with priority rules lead to valuable insights, multi-project research has progressed to more advanced algorithmic procedures. In order to evaluate whether our datasets are also challenging for more complex algorithms, we implemented a genetic algorithm. In this subsection, we start by describing the genetic algorithm. Afterwards, we will discuss the results of a computational experiment and draw conclusions regarding the proposed datasets.

5.2.1 Description of genetic algorithm

For the genetic algorithm (GA), a schedule is represented by an activity list, where $p(a_{ij})$ denotes the position of activity a_{ij} in a given activity list. The list is precedence feasible, i.e. $p(a_{kj}) < p(a_{ij}), \forall a_{kj} \in P_{ij}$ (Hartmann 1998). Now, the range of feasible positions for a_{ij} is denoted by $[a_{ij}, \bar{a}_{ij}]$, where a_{ij} and \bar{a}_{ij} are, respectively, the leftmost and rightmost position in the list where the activity can be placed without violating the precedence constraints.

The genetic algorithm has a population size of 200. In the initial population, 150 random precedence feasible lists are created and the remaining 50 lists are obtained by scheduling the instance with all previously discussed priority rules and converting the 50 best schedules to their corresponding activity lists. In every iteration, the 16 best individuals are copied to the next population. The rest of the population is obtained by repeatedly selecting two parents, creating offspring using crossover operators and possibly applying a mutator to the offspring. The first parent is selected by a two-round tournament selection. The second parent is selected either by a one-round tournament selection or by a uniform selection; both selection operators have a 50% probability of being used. The offspring are created by randomly applying one out of four crossover operators on the parents. Each of the children is mutated with a probability of 20%. When mutation is triggered, the GA selects one out of seven mutator operators. The parameter values for the genetic algorithm were set based on experiments on a training set.

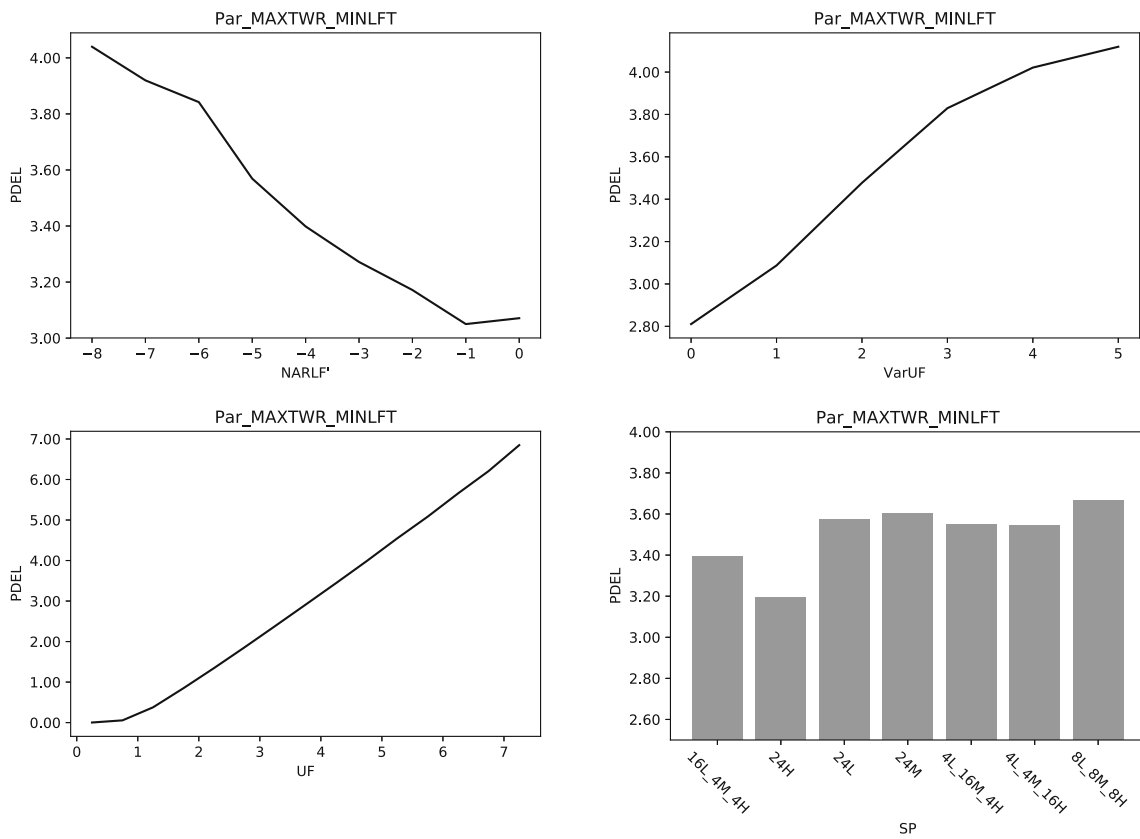


Fig. 6 Main effect plots PDEL

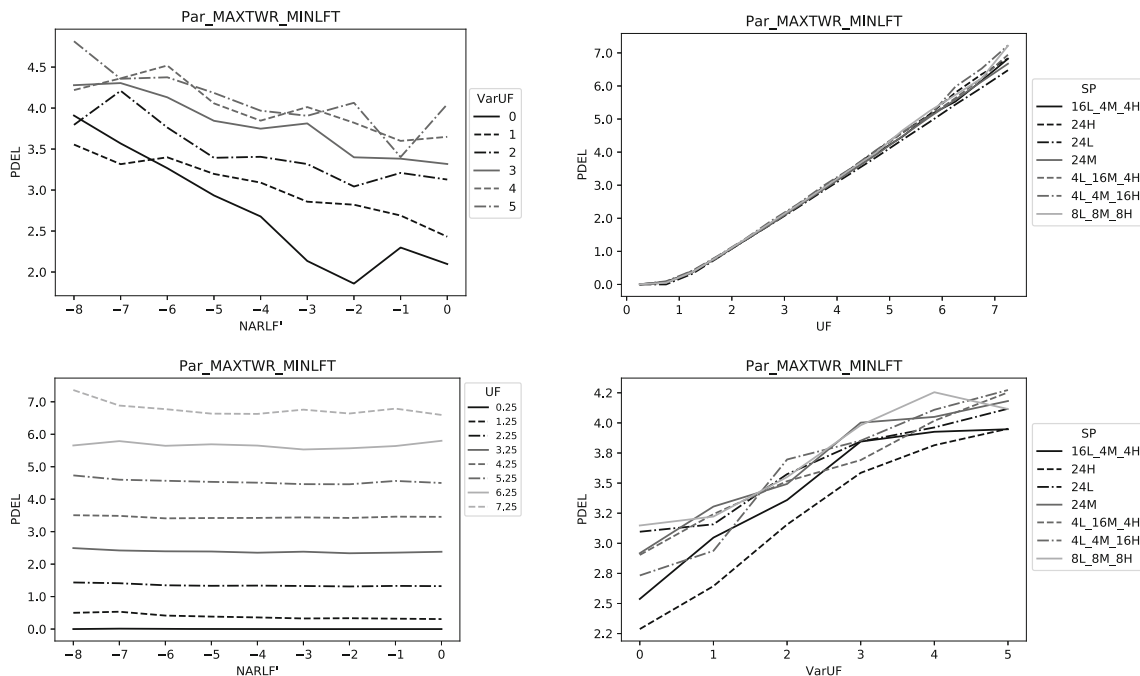


Fig. 7 Interaction effect plots PDEL

Now we will explain the crossover and mutator operators that were implemented for the genetic algorithm. To clarify the discussion, we will first explain two concepts. The order

o_{ij} of an activity indicates its position in the list relative to the other activities of the same project. $o_{ij} = z$ means that z activities of project j precede a_{ij} in the list. As such, for

any activity a_{ij} of project j , $o_{ij} \in [0, |I_j| - 1]$. Second, the centre c_j of a project gives an indication of the relative position of projects in the activity list. It is defined as $c_j = \frac{1}{|I_j|} \sum_{a_{ij} \in I_j} p(a_{ij})$ (Asta et al. 2016). If the activities of project j are situated more in the front of the list than those of project l , it follows that $c_j < c_l$. As our results in the previous subsection and recent research (Asta et al. 2016; Vázquez et al. 2015; Zheng et al. 2013) confirmed that separating project and activity selection can improve the performance, we chose to implement operators that change activity lists at the project level, in addition to those that work on the activity level.

Crossover operators: we chose to implement a traditional two-point crossover and three new project level crossover operators.

- *Two-Point:* Select two positions, keep the activities to the left of the first position and to the right of the second position. Swap the activities between the two positions with the other parent. For the details, we refer to Hartmann (1998).
- *Project Order Swap:* Copy one of the parents into the child list. Run through the child list and at each position replace a_{ij} by the activity of project j that has order o_{ij} in the other parent. Repeat this process, starting with a copy from the other parent.
- *Hybrid Project Order 1:* Calculate the project orders for both parents, based on their centre. Construct a precedence graph of the projects where project j precedes project k if j precedes k in the order of both parents. Create a new random project order that respects the project precedence graph. For each child, run through the project order, and per project add all its activities to the child following the activity order of one of the parents. The first child and second child inherit the activity order of the first and second parent, respectively.
- *Hybrid Project Order 2:* This operator is similar to the previous operator, but for each child a different project order is created.

Mutator operators: we based ourselves on the neighbourhood operators from Asta et al. (2016) as they were shown to perform well for the RCMPSP.

- *Swap:* Select a random activity a_{ij} and a second random activity a_{kl} with $p(a_{kl}) \in [\underline{a}_{ij}, \overline{a}_{ij}]$. If $p(a_{ij}) \in [\underline{a}_{kl}, \overline{a}_{kl}]$, swap the activities. Otherwise, do nothing.
- *Insert:* Select a random activity a_{ij} and insert it in a random position in $[\underline{a}_{ij}, \overline{a}_{ij}]$.
- *Scramble:* Choose two random positions in the list and remove all activities between these positions. Randomly reinsert the removed activities in the empty spots, respecting precedence constraints.

- *Swap Projects:* Randomly select two projects k and l and remove all their activities from the list. Insert all activities of l in the first $|I_l|$ empty spots, preserving the original activity order. Then, insert all activities of k in the remaining spots, again preserving the order.
- *Swap Adjacent Projects:* Sort the projects according to their centre value and select two projects that are adjacent in this ordering. Apply it Swap Projects to these projects.
- *Compress Project:* Select a random project j and remove all its activities from the list. Left shift all remaining activities as much as possible, such that they form one contiguous block. Select a random position p in $[0, s - |I_j|]$ and right shift the activities on positions $[p, s - |I_j|]$ as much as possible, leaving a contiguous block of empty spots at the interval $[p, p + |I_j| - 1]$. Insert all activities of project j in this empty block, respecting the original activity ordering.
- *Shuffle Project:* Select a random project j and remove all its activities from the list. Reinsert the activities in the empty positions in a random order, respecting precedence constraints.

Each time an operator is applied, it is randomly chosen from the list above. Based on how often an operator improves the solution, its selection probability can be adapted by the algorithm. This will make the algorithm favour well-performing operators. However, in order to stimulate diversification, we set a lower bound on the selection probability of the operators.

5.2.2 Discussion of performance

For the computational experiment, we let the genetic algorithm run for 30 s on each instance. The tests were executed on the STEVIN HPC-UGent Supercomputer (Processor architecture: 2x18-core Intel Xeon Gold 6140, clock speed 2.3 GHz). The left hand side of Fig. 8 shows the improvement divided by the total improvement found after 30 s. The right hand side shows the same data, but with the first 2 s of running time truncated, allowing a more nuanced comparison of the improvement profiles. A first observation is that for the sets BY10, RCMPSP LIB and 6_60, the GA improves quickly, but stabilises relatively fast. This indicates that the algorithm can quickly find good solutions which can almost not be improved any further, even after a long time searching. The other three datasets prove to be more difficult, as the genetic algorithm keeps finding considerable improvements in later time slots. The figure suggests that the improvement potential of the algorithm on the dataset 24_60 is not completely exploited after 30 s.

To support the previous observations, we set up a second experiment where the initial population consisted of 200

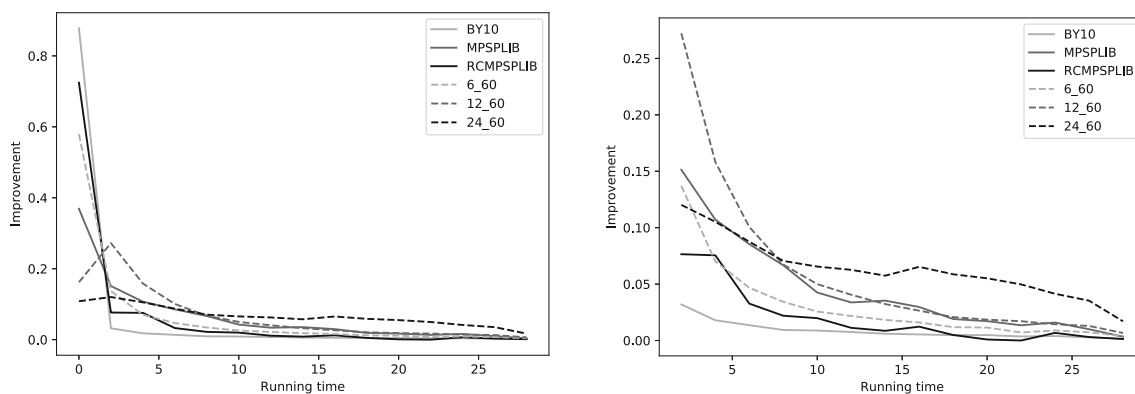


Fig. 8 Improvement profile genetic algorithm

Table 13 Time to beat best PR

Dataset	Time	% Outperformed
BY10	1.94	93
MPSPLIB	7.71	62
RCMPSPLIB	1.76	85
6_60	1.80	98
12_60	4.73	93
24_60	13.79	82

random activity lists and measured how much time the GA needed to obtain a solution with the same objective function value as the best priority rule for an instance, with a maximum running time of 30 s. Table 13 shows the average time the algorithm needed to surpass the best priority rule and the percentage of instances for which it outperformed the priority rule after 30 s.

The table shows that the randomly initialised version of the GA needs much more time to match the best priority rule on the datasets MPSPLIB, 12_60 and especially on 24_60 in comparison with the others. This confirms the observation from the previous experiment that these three datasets are the most challenging for more advanced algorithms. The second column shows that MPSPLIB contains (relatively speaking) the most instances for which the GA could not outperform the PR, followed by 24_60 and RCMPSPLIB. Furthermore, the table shows that decoupled priority rules provide invaluable knowledge for more advanced heuristics, especially when the instances become more challenging.

6 Conclusion

This paper provides an overview of the existing multi-project measures and benchmark datasets and proposes a new dataset. Furthermore, decoupled versions of the single-project schedule generation schemes are presented and

evaluated on all available datasets for the basic variant of the RCMPSP.

Our literature review revealed a positive trend towards reproducibility and comparability in multi-project literature, which improves the consistency of conclusions made by researchers. The existing summary measures were reviewed, and three adaptations were proposed: (1) NARLF was modified to NARLF', (2) SP was used to describe the network structure, and (3) σ_{UF}^2 was calculated in a more general manner.

In Sect. 3, we applied the multi-project approach to priority rule-based scheduling, resulting in the decoupled schedule generation schemes. Our computational experiments showed that decoupled scheduling schemes are valuable because of two reasons. First, they can easily be used by practitioners to improve the quality of their schedules. Second, these schemes can be implemented as building blocks in more advanced algorithms. Our second computational experiment showed that this is very important when the instances become more difficult to solve.

In Sect. 4, we reimplemented the generation procedure of Browning and Yassine (2010b). Using the reimplemented procedure, we proposed new datasets that complement the existing data in two ways: a wider range of parameter combinations and portfolios with a wider variety of SP values. Adding this variation to our datasets leads to new insights regarding the best performing priority rules. Furthermore, two of the three sets proved to be challenging for the genetic algorithm that we implemented. During the generation process, we observed that the feasible ranges for the NARLF' and UF values depend on the network structure and the portfolio size.

The computational experiments in Sect. 5 provided insights into the performance of the different priority rules. For the APD objective, the best activity PR's are MINLST, MINSLKd and MINLFT. For the datasets with projects that have similar SP values, the project PR's MINTWR, MINTWK and MAXWK performed best, while MINCP

and MINSP performed better on the datasets where the SP varies more over projects. Furthermore, we observed that the project PR has the strongest impact on the performance of the scheduling schemes. This confirms the insights from Asta et al. (2016) that good solutions approximate a total order on the projects.

For the PDEL objective, we found that the decoupled schemes were slightly worse on the existing sets, but were better on our own datasets. This confirms again that our datasets include cases that are absent in the existing sets. Additionally, these results show that even for the single-project objective PDEL, decoupled scheduling outperforms single-project priority rules. The best activity PR’s are the same as for APD, but the best project PR’s are now MAX-TWR and MINWK. The choice of project PR is still important, but has a smaller impact than for APD. Furthermore, for PDEL the parallel scheduling scheme is consistently better than the serial one.

We identify the following opportunities for future research. A first opportunity is to explore additional summary measures for the RCMPSP that either determine PR performance or that have fewer dependencies, which would facilitate the construction of full factorial test designs. A second possible research avenue is to investigate whether the decoupled scheduling approach also performs well on other objective functions or on extensions of the basic RCMPSP. Third, in the spirit of the MISTA 2013 challenge, more advanced scheduling algorithms could be designed that incorporate decoupled scheduling as a building block.

Acknowledgements We acknowledge the support provided by the Special Research Fund (BOF Grant No. DOC014-18 Van Eynde) and the National Bank of Belgium for providing the first author with a pre-doctoral fellowship. The computational resources (Stevin Super-computer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government department EWI.

Appendix: Summary measures

Measure	Formula	Notes
<i>Network measures</i>		
C	$\frac{A' - A'_{\min}}{A'_{\max} - A'_{\min}}$	
NC	$\frac{A_j^n}{ I_j + 2}$	The average number of non-redundant arcs per node, including dummies ^a
NPL	$\frac{ I_j }{L_j}$	The average number of activities in parallel
OS	$\frac{A_j^r}{ I_j * (I_j - 1) / 2}$	The ratio of the number of arcs (excluding dummies) and the theoretical number of arcs ^b
SP	$\frac{L_j - 1}{ I_j - 1}$	The ratio of the longest chain in the network and the total number of activities
<i>Resource measures</i>		
ARLF	$\frac{1}{CP_j} \sum_{t=r_j}^{r_j+CP_j} \sum_{i \in I_j} \sum_{k \in H_{ij}} Z_{ijt} X_{ijt} \left(\frac{r_{ijk}}{ H_{ij} } \right)$	
AUF	$\frac{1}{S} \sum_{s=1}^S \frac{W_{sk}}{l_s R_k}$	S is the number of intervals, W_{sk} the total resource demand for resource k in interval s and l_s the length of the interval
Load	$\frac{R_k}{ J \cdot d_k}$	d_k is the duration of an activity with resource demand for type k
MAUF	$\frac{1}{M_{ES}} \sum_{s=\min_j r_j}^{M_{ES}} \frac{W_{sk}}{l_s R_k}$	Similar to AUF, with $l_s = 1$
σ_{UF}^2	$\frac{\sum_{k \in K} (UF - UF_k)^2}{ K }$	
MUF	$\max_{k \in K} \frac{\sum_{j \in J} \sum_{i \in I_j} r_{ijk} d_{ij}}{R_k M_{ES}}$	The ratio of the earliest start schedule resource requirements and the available resources
NARLF	$\frac{1}{ J * CP_{\max}} \sum_{j \in J} \sum_{t=r_j}^{r_j+CP_j} \sum_{i \in I_j} \sum_{k \in H_{ij}} Z_{ijt} X_{ijt} \left(\frac{r_{ijk}}{ H_{ij} } \right)$	
NARLF'	$\frac{1}{ J * CP_{\max}} \sum_{j \in J} \sum_{t=r_j}^{r_j+CP_j} \sum_{i \in I_j} \sum_{k \in H_{ij}} Z'_t X_{ijt} \left(\frac{r_{ijk}}{ H_{ij} } \right)$	
RC	$\frac{1}{ K } \sum_{k \in K} \frac{\bar{r}_k}{R_k}$	Where \bar{r}_k is the average resource requirement, when it is required by an activity
RF	$\frac{\sum_{j \in J} \sum_{i \in I_j} H_{ij} }{ K \sum_{j \in J} I_j }$	The average percentage of resource types that an activity requires
RS	$\frac{R_k - r_k^{\min}}{r_k^{\max} - r_k^{\min}}$	The resource strength. r_k^{\min} is the maximum demand for resource k over all activities, r_k^{\max} the peak demand for type k in the unconstrained earliest start schedule

^a A_j^n is the number of non-redundant arcs in the network of project j , including dummy arcs

^b A_j^r is the total number of redundant and non-redundant arcs in the network of project j , excluding dummy arcs

References

- Adhau, S., Mittal, M., & Mittal, A. (2012). A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. *Engineering Applications of Artificial Intelligence*, 25(8), 1738–1751.
- Adhau, S., Mittal, M., & Mittal, A. (2013). A multi-agent system for decentralized multi-project scheduling with resource transfers. *International Journal of Production Economics*, 146(2), 646–661.
- Asta, S., Karapetyan, D., Kheiri, A., Özcan, E., & Parkes, A. J. (2016). Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences*, 373, 476–498.
- Beşikci, U., Bilge, Ü., & Ulusoy, G. (2013). Resource dedication problem in a multi-project environment. *Flexible Services and Manufacturing Journal*, 25(1–2), 206–229.
- Beşikci, U., Bilge, Ü., & Ulusoy, G. (2015). Multi-mode resource constrained multi-project scheduling and resource portfolio problem. *European Journal of Operational Research*, 240(1), 22–31.
- Browning, T. R., & Yassine, A. A. (2010a). A random generator of resource-constrained multi-project network problems. *Journal of Scheduling*, 13(2), 143–161.
- Browning, T. R., & Yassine, A. A. (2010b). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, 126(2), 212–228.
- Chakraborty, R. K., Sarker, R. A., & Essam, D. L. (2017). Resource constrained multi-project scheduling: A priority rule based evolutionary local search approach. In G. Leu, H. K. Singh, & S. Elsayed (Eds.), *Intelligent and evolutionary systems* (pp. 75–86). Canberra, Australia: Springer.
- Chen, P.-H., & Shahandashti, S. M. (2009). Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Automation in Construction*, 18(4), 434–443.
- Chiu, H. N., & Tsai, D. M. (2002). An efficient search procedure for the resource-constrained multi-project scheduling problem with discounted cash flows. *Construction Management and Economics*, 20(1), 55–66.
- Confessore, G., Giordani, S., & Rismondo, S. (2007). A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research*, 150(1), 115–135.
- Davis, E. W. (1975). Project network summary measures constrained-resource scheduling. *AIIE Transactions*, 7(2), 132–142.
- Deckro, R. F., Winkofsky, E., Hebert, J. E., & Gagnon, R. (1991). A decomposition approach to multi-project scheduling. *European Journal of Operational Research*, 51(1), 110–118.
- Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). Rangen: A random network generator for activity-on-the-node networks. *Journal of scheduling*, 6(1), 17–38.
- Dumond, J., & Mabert, V. A. (1988). Evaluating project scheduling and due date assignment procedures: An experimental analysis. *Management Science*, 34(1), 101–118.
- Elmaghraby, S. E. (1977). *Activity networks: Project planning and control by network models*. New York: Wiley.
- Geiger, M. J. (2017). A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, 256(3), 729–741.
- Gonçalves, J. F., Mendes, J. J., & Resende, M. G. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189(3), 1171–1190.
- Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7), 733–750.
- Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1), 1–14.
- Homberger, J. (2007). A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, 14(6), 565–589.
- Homberger, J. (2012). A (μ, λ) -coordination mechanism for agent-based multi-project scheduling. *OR Spectrum*, 34(1), 107–132.
- Kolisch, R., & Sprecher, A. (1997). Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European Journal of Operational Research*, 96(1), 205–216.
- Krüger, D., & Scholl, A. (2009). A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197(2), 492–508.
- Krüger, D., & Scholl, A. (2010). Managing and modelling general resource transfers in (multi-) project scheduling. *OR Spectrum*, 32(2), 369–394.
- Kumanan, S., Jose, G. J., & Raja, K. (2006). Multi-project scheduling using an heuristic and a genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 31(3–4), 360–366.
- Kurtulus, I. (1985). Multiproject scheduling: Analysis of scheduling strategies under unequal delay penalties. *Journal of Operations Management*, 5(3), 291–307.
- Kurtulus, I., & Davis, E. (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, 28(2), 161–172.
- Lawrence, S. R., & Morton, T. E. (1993). Resource-constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. *European Journal of Operational Research*, 64(2), 168–187.
- Lee, Y.-H., Kumara, S. R., & Chatterjee, K. (2003). Multiagent based dynamic resource scheduling for distributed multiple projects using a market mechanism. *Journal of Intelligent Manufacturing*, 14(5), 471–484.
- Liberatore, M. J., & Pollack-Johnson, B. (2003). Factors influencing the usage and selection of project management software. *IEEE transactions on Engineering Management*, 50(2), 164–174.
- Lova, A., Maroto, C., & Tormos, P. (2000). A multicriteria heuristic method to improve resource allocation in multiproject scheduling. *European Journal of Operational Research*, 127(2), 408–424.
- Lova, A., & Tormos, P. (2001). Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling. *Annals of Operations Research*, 102(1–4), 263–286.
- Maroto, C., Tormos, P., & Lova, A. (1999). The evolution of software quality in project scheduling. In J. Weglarz (Ed.), *Project scheduling* (pp. 239–259). Boston, MA: Springer.
- Mittal, M., & Kanda, A. (2009). Two-phase heuristics for scheduling of multiple projects. *International Journal of Operational Research*, 4(2), 159–177.
- Payne, J. H. (1995). Management of multiple simultaneous projects: A state-of-the-art review. *International Journal of Project Management*, 13(3), 163–168.
- Pérez, E., Posada, M., & Lorenzana, A. (2016). Taking advantage of solving the resource constrained multi-project scheduling problems using multi-modal genetic algorithms. *Soft Computing*, 20(5), 1879–1896.
- Singh, A. (2014). Resource constrained multi-project scheduling with priority rules and analytic hierarchy process. *Procedia Engineering*, 69, 725–734.
- Song, W., Kang, D., Zhang, J., & Xi, H. (2017). A multi-unit combinatorial auction based approach for decentralized multi-project scheduling. *Autonomous Agents and Multi-agent Systems*, 31(6), 1548–1577.

- Toffolo, T. A., Santos, H. G., Carvalho, M. A., & Soares, J. A. (2016). An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *Journal of Scheduling*, 19(3), 295–307.
- Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., & Tavares, L. V. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187(2), 511–524.
- Vázquez, E. P., Calvo, M. P., & Ordóñez, P. M. (2015). Learning process on priority rules to solve the RCMPSP. *Journal of Intelligent Manufacturing*, 26(1), 123–138.
- Vercellis, C. (1994). Constrained multi-project planning problems: A Lagrangean decomposition approach. *European Journal of Operational Research*, 78(2), 267–275.
- Wang, X., Chen, Q., Mao, N., Chen, X., & Li, Z. (2015). Proactive approach for stochastic RCMPSP based on multi-priority rule combinations. *International Journal of Production Research*, 53(4), 1098–1110.
- Wang, Y., He, Z., Kerkhove, L.-P., & Vanhoucke, M. (2017). On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. *Computers and Industrial Engineering*, 114, 223–234.
- Wauters, T., Kinable, J., Smet, P., Vancroonenburg, W., Berghe, G. V., & Verstichel, J. (2016). The multi-mode resource-constrained multi-project scheduling problem. *Journal of Scheduling*, 19(3), 271–283.
- Wauters, T., Verbeeck, K., De Causmaecker, P., & Berghe, G. V. (2015). A learning-based optimization approach to multi-project scheduling. *Journal of Scheduling*, 18(1), 61–74.
- Yang, K.-K., & Sum, C.-C. (1993). A comparison of resource allocation and activity scheduling rules in a dynamic multi-project environment. *Journal of Operations Management*, 11(2), 207–218.
- Yang, K.-K., & Sum, C.-C. (1997). An evaluation of due date, resource allocation, project release, and activity scheduling rules in a multi-project environment. *European Journal of Operational Research*, 103(1), 139–154.
- Zheng, Z., Shumin, L., Ze, G., & Yueni, Z. (2013). Resource-constraint multi-project scheduling with priorities and uncertain activity durations. *International Journal of Computational Intelligence Systems*, 6(3), 530–547.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.