European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete optimization

A genetic algorithm with resource buffers for the resource-constrained multi-project scheduling problem

Dries Bredael [a], Mario Vanhoucke [a,b,c,*]

[a] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium
[b] Operations and Technology Management Centre, Vlerick Business School, Reep 1, 9000 Ghent, Belgium
[c] UCL School of Management, University College London, 1 Canada Square, London E14 5AA, United Kingdom

ARTICLE INFO

Keywords:
Project scheduling
Genetic algorithms
Metaheuristics
Multi-project

ABSTRACT

In this study, we compose a new metaheuristic algorithm for solving the resource-constrained multi-project scheduling problem. Our approach is based on a general metaheuristic strategy which incorporates two resource-buffered scheduling tactics. We build on the most effective evolutionary operators and other well-known scheduling methods to create a novel genetic algorithm with resource buffers. We test our algorithm on a large benchmark dataset and compare its performance to ten existing metaheuristic algorithms. Our results show that our algorithm can generate new best-known solutions for about 20% of the test instances, depending on the optimisation criterion and due date. In some cases, our algorithm outperforms all other available methods combined. Finally, we introduce a new schedule metric that can quantitatively measure the dominant structure of a solution, and use it to analyse the differences between the best solutions for different objectives, due dates, and instance parameters.

1. Introduction

The Resource-Constrained Multi-Project Scheduling Problem (RCMPSP) is an optimisation problem where multiple independent projects need to be jointly planned and completed whilst sharing limitedly available renewable resources among them. As an extension of its single-project variant, the multi-project scheduling problem is also NP-hard (Blazewicz et al., 1983). Therefore, finding the optimal schedule for realistically sized RCMPSPs is a difficult feat because exact solution methods quickly become computationally intractable. As a result, most of the academic literature has resorted to designing metaheuristic solution methods to find near-optimal solutions in ample time.

Recently, Bredael and Vanhoucke (2023) have compared the available metaheuristic solution algorithms on a dataset by Van Eynde and Vanhoucke (2020) for multiple optimisation criteria and different types of project due dates. They found that scheduling the projects individually in a certain project order quickly improves the schedule for project tardiness-based objectives. Therefore, the algorithms that employ such a project prioritisation strategy performed well on these criteria. However, enforcing a strict project order during the scheduling process can lead to inefficiencies in resource utilisation, as was demonstrated in this study. This paper builds upon the insights gained from the benchmark comparison and the analysis of ten existing metaheuristic algorithms to develop a new metaheuristic algorithm. The result is a solution procedure that is a composition of previously identified effective elements from literature based on a genetic algorithm. However, the key and novel component of our algorithm is the implementation of two variants of a resource-buffered scheduling strategy within a serial schedule generation scheme. The first variant artificially limits the number of resources that a project is allowed to consume at any point during the scheduling horizon. The second variant enforces a portfolio-level dynamic resource buffer that is iteratively lowered during the scheduling process. The performance of this genetic algorithm with resource buffers is compared to the best-known solutions generated by the ten existing metaheuristics on the available benchmark datasets. Our results show that our algorithm is very competitive with the existing solution procedures and can generate new best solutions for about 20% of test instances on an identical stop criterion, depending on the objective and due date.

Section 2 provides an overview of the relevant literature. In Section 3, we introduce the problem statement and the rationale behind our approach, as well as the resource-buffered scheduling strategies.

* Corresponding author at: Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium.
E-mail addresses: dries.bredael@ugent.be (D. Bredael), mario.vanhoucke@ugent.be, mario.vanhoucke@vlerick.com, m.vanhoucke@ucl.ac.uk (M. Vanhoucke).

https://doi.org/10.1016/j.ejor.2023.11.009
Received 15 December 2022; Accepted 3 November 2023
Available online 8 November 2023

Table 1
Algorithmic comparison of existing and novel centralised metaheuristic solution procedures for the RCMPSP.

Author(s)	Abbrev.	Category[a]	General characteristics			Solution representation					Scheduling strategies			Selection operators				Crossover operators					
			Pre-opt. init. solutions	Project prioritisation scheme	Restart strategy	Activity sequence	Project sequence	(Project-) activity lists	(Set of) priority rule(s)	Random keys	Backward-Forward scheduling	(Time-)Delayed scheduling	(Resource-)Buffered scheduling	Uniform	Tournament	Fitness-weighted	PPL-weighted	One-point	Two-point	Uniform	Adapted order	Project order swap	Hybrid project order
Kumanan et al. (2006)	K06	GA		×			×		×					×				×					
Homberger (2007)	H07	GA			×	×					×			×					×				
Gonçalves et al. (2008)	G08	GA								×		×		×		×				×			
Chen and Shahandashti (2009)	C09	GA/SA-hybrid				×										×			×				
Sonmez and Uysal (2015)	S15	GA/SA-hybrid								×	×			×					×				
Vázquez et al. (2015)	V15	GA				×			×							×					×		
Wauters et al. (2015)	W15-S	LA		×			×	×			×												
Wauters et al. (2015)	W15-I	LA					×	×			×												
Pérez et al. (2016)	P16	GA		(×)						×				×					×				
Van Eynde and Vanhoucke (2020)	V20	GA	×	×		×								×	×				×			×	×
This study	B23	GA	×	×					×	×	×		×	×	×		×	×	×	×			
Initial solutions	–	–		×					×	×													
Evolutionary algorithm	–	GA	×	(×)						×	×		×	×	×		×	×	×	×			

[a] GA: Genetic Algorithm; GA/SA-hybrid: Hybrid of Genetic Algorithm and Simulated Annealing; LA: Learning-based Approach.

We also propose a new schedule metric to analyse the solutions provided by our algorithm. Section 4 describes the composition of the metaheuristic algorithm in more detail. Section 5 presents the results of our experiments, including the optimisation of our algorithm's parameters, the comparison of its performance to existing methods, and the analysis of the contribution of the new resource-buffered scheduling strategies on the algorithmic performance. We also examine the impact of optimisation criteria, due dates, and instance metrics on the relative performance of our algorithm and the structure of the best schedules. Finally, in Section 6, we summarise our main findings and discuss directions for future research.

2. Literature review

An overview of ten existing metaheuristic solution procedures for the RCMPSP, analysed in a benchmark overview by Bredael and Vanhoucke (2023) on the MPLIB1 dataset created by Van Eynde and Vanhoucke (2020), is presented in Table 1. This table summarises five ranges of characteristics. General characteristics refer to strategic components that can be identified in the overall algorithm. Afterwards, the indirect solution representation of each algorithm is summarised. Scheduling strategies refer to heuristic methods to improve the schedule generation procedure for a given indirect solution representation. Finally, the selection and crossover operators of the evolutionary algorithm are presented. It is observed that the majority of the metaheuristic solution procedures in literature are based on genetic algorithms where the main differences can be distinguished in the choice of the indirect solution representation and the evolutionary operators. One of the key differences of the new algorithm, presented in this study, is the integration of novel scheduling approaches with resource buffers as an addition to its schedule generation scheme. Additionally, the metaheuristic algorithm inherits a selection of the most effective components from the solution procedures in literature. In this section, we present a discussion on the algorithmic characteristics that were adapted from these solution procedures as well as the link of the novel elements with the academic literature. First, we discuss the generation of a diverse set of initial solutions based on different well-performing elements from literature. Afterwards we discuss the evolutionary mechanism of the genetic algorithm and its schedule generation schemes.

It has been established that the performance of metaheuristic algorithms depends significantly on the quality of the initial set of solutions (Bredael & Vanhoucke, 2023). The effectiveness of priority rule heuristics has also been confirmed by multiple studies (Browning & Yassine, 2010; Van Eynde & Vanhoucke, 2020). In light of these findings, our approach aims to construct a diverse and competitive set of solutions to initialise the metaheuristic algorithm. *Decoupled* priority rules, which consist of two separate rules, are a key component of our strategy. The first project priority rule determines the project scheduling order, while the other defines the order in which the activities of an individual project are scheduled. Furthermore, inspired by the *sequential* approach observed in Wauters et al. (2015), additional initial solutions are generated by constructing randomised project sequences. Our scheduling scheme generates schedules in a serial manner, sequentially planning the projects according to the determined project priority rule or project sequence. The scheduling order of activities within each project is determined either by the activity priority rule or randomly for the sequential approach. This project prioritisation strategy was shown to be advantageous for mean project-tardiness objectives. To ensure diversity of the initial solutions, the *interleaved* approach (Wauters et al., 2015) is integrated. This approach stands in contrast to the project prioritisation strategy, as it schedules only one activity at a time for each project, proceeding at an equal pace. The (interleaved) order of projects is determined by a randomly generated project sequence. The resulting solutions are particularly effective to minimise the portfolio makespan or a maximum project-tardiness objective. Finally, like most of the available solution procedures, we generate additional solutions by creating randomly determined RK vectors, which are evaluated using the serial schedule generation scheme.

Similar to other population-based methods described in the literature, our evolutionary algorithm is an implementation of the genetic algorithm framework. In our approach, solutions are represented by a random key vector that determines the scheduling order of activities. The initial population is generated using a combination of the methods mentioned earlier. Previous research conducted by Pérez et al. (2016) found that the RK solution representation offers the best performance for RCMPSP test instances. This observation was further supported by a benchmark study which highlighted the RK representation's tendency to converge towards solutions with a form of project prioritisation (Bredael & Vanhoucke, 2023). It is important to note that the solutions generated by these heuristic approaches are converted into a Random Key (RK) representation based on the start times of the activities within the schedule. This conversion is necessary to ensure compatibility with the subsequent evolutionary algorithm's solution representation.

The main differentiating factor, and a core novelty of our overall algorithm, is the scheduling strategy that is embedded within the schedule generation scheme, i.e. the manner in which indirect solution representations are translated into direct schedule representations such that the solution quality can be evaluated. For this study, two variants

of a *resource-buffered* scheduling strategy were developed on the basis of a serial schedule generation scheme. The resource-buffered scheduling technique imposes artificial scheduling restrictions that are not a part of the solution representation. These exogenous resource restrictions limit the access to the shared resources for all or a subset of activities. These scheduling restrictions can be interpreted as a buffer of resources that is either only available for the activities of certain projects or, is only gradually released at certain time-points in the schedule. To the best of our knowledge, this type of scheduling strategy is currently inexistent in literature on the RCMPSP. The G08 algorithm by Gonçalves et al. (2008) includes a variation of a different scheduling strategy with a similar philosophy that instead directly imposes artificial restrictions on the earliest-feasible start times of activities. A key difference is that their scheduling restrictions, in the form of artificial activity delay times and project release dates, are an endogenous part of the solution representation. Because this technique artificially imposes delays (i.e. restrictions) on the start times of activities we term this *time-delayed* scheduling. In contrast to the G08 algorithm, we made the deliberate decision to keep the parameters of the schedule generation scheme external to the algorithm. This choice was based on the observation that a dual solution representation hampers the convergence of the algorithm. Additionally, the exogenous nature of the resource-buffered scheduling parameters allows us to control and analyse the performance of these scheduling strategies.

3. Problem statement and solution strategies

A formal description of the RCMPSP is given in Section 3.1. In Section 3.2, we introduce a new metric to quantify the degree of project prioritisation that is present within a schedule by analysing its indirect activity sequence representation. Section 3.3 discusses the rationale of the resource-buffered scheduling strategy. Finally, Section 3.4 presents two implementations of this resource-buffered scheduling strategy.

3.1. Problem definition

The RCMPSP is described using the symbols in Table 2. A set J of N projects, indexed by j, needs to be planned in a multi-project schedule under joint renewable resource constraints, yet separate precedence restrictions. Each project j consists of I_j non-preemptive activities, indexed by i that have their own finish-to-start precedence relations. Correspondingly to most studies on the RCMPSP, we solely consider the situation where precedence relations exist exclusively within the activities of the same project and not between activities of different projects. A duration d_{ij} and a number of (k) resource requirements, represented by r_{ijk}, is given for every activity a_{ij}. For each project, an arrival time or release date ρ_j is defined that specifies the earliest point in the time horizon that any activity of this project can be planned in the portfolio schedule. Additionally, for every project a due date δ_j is given which acts as a soft constraint on the projects' makespan to penalise project completion late of this due date. Finally, a set of renewable and shared, global resources with limited availability R_k, indexed by k is presented. The combined consumption of these resources by the activities of all the projects can never be higher than their availability at any point during the time horizon of the schedule.

The task presented by the RCMPSP is to define a feasible set of start times, one for each activity, that optimises a given optimisation criterion. Most of literature has limited itself to only studying one or two objective functions at the same time. In this study, we analyse our solution procedure on seven optimisation criteria for the RCMPSP. Objectives that optimise for project tardiness criteria require the specification of a project due date. Table 3 describes the different optimisation criteria and due dates that are used in this study. These are identical to the ones proposed by Bredael and Vanhoucke (2023). A more detailed description of these optimisation criteria and due dates and the performances of these metaheuristic algorithms can be retrieved in the aforementioned study.

3.2. Quantifying the degree of project prioritisation in a schedule

Project prioritisation refers to the grouping of projects' activities in a clustered manner within the portfolio's schedule. It is observed when the start times of activities within one project are much closer to each other compared to the start times of activities from other projects. Currently, there is no metric available to quantify the extent of this clustering behaviour within a schedule. Therefore, we propose a new metric to address this gap. It should be noted that the proposed metric is applicable only to schedules where the number of non-dummy activities is the same for each project, which is the case in the dataset we utilise. This limitation arises because the value of the metric would be influenced by the order of projects within the schedule, which is not desirable in this context.

The *Project Prioritisation Level (PPL)* expresses this degree of project prioritisation as a value between 0 and 1. A value of 0 indicates that no clustering of the activities around the activities of the same project can be observed, while a value of 1 indicates a maximal degree of project prioritisation. A schedule with a maximal degree of project prioritisation is characterised by a maximal clustering of the start times of activities belonging to the same project. This indicates that, for any project, the start time of any activity of any different project is either less than or equal to the start time of the dummy-start activity or, larger than or equal to the start time of the dummy-end activity of the former project. To compute the PPL of a schedule, first, the direct schedule representation is converted into an indirect Activity Sequence (AS) representation by sorting the non-dummy activities from lowest to highest activity start time. In case of a tie, the activity with the smallest finish time is put first in the sequence. In the situation where two activities are tied on both metrics, the activity with the lowest project- and activity-number is prioritised.

The PPL quantifies the extense of this clustering behaviour by analysing the average positions, ap_j, of the activities of each project j in the schedule. The average position of a project, ap_j, is computed by averaging the positional indexes of its constituent activities in the AS associated with the schedule. This computation is formalised in Eq. (1), where v_{ij} denotes the positional index of activity i of project j. As a result, the project with the highest priority will have the lowest average positional index, and vice-versa. In a schedule with a high degree of project prioritisation, the average positions of the projects will be more distant from each other. Therefore, the variance of these average positions, denoted by V, can be used as a proxy to analyse the magnitude of this characteristic and its formula is given in Eq. (2). The obtained variance value, V, is not a relative indicator of the prioritisation level. To generate a relative indicator between 0 and 1, where the latter indicates a schedule with maximal project prioritisation, the variance of the average positions of the current schedule, V_{cur}, is compared to the V-score of a hypothetical schedule with a maximal degree of prioritisation. This extreme schedule is given by an AS where any strict project order is enforced. As a result, its associated V-score, V_{max}, is also maximal. By dividing the variance of the current schedule, V_{cur}, over the variance of the extreme case, V_{max}, the PPL, given in Eq. (3), is obtained.

$$ap_j = \frac{\sum_{i \in Ij} v_{ij}}{n_j}, \forall j \in J \tag{1}$$

$$V = \frac{\sum_{j \in J} (ap_j - \frac{\sum_{j \in J} ap_j}{N})^2}{N} \tag{2}$$

$$PPL = \frac{V_{cur}}{V_{max}} \tag{3}$$

Fig. 1 illustrates the computation of the PPL metric for two schedules with a visually different degree of activity clustering. The schedule on the left panel exhibits a high degree of project prioritisation while the schedule on the right panel does not seem to show this characteristic. The PPL of the left-hand schedule is calculated by computing

Table 2
Overview of notations and formulae.

Notations and formulae			
General parameters		*Network parameters*	
J	The set of projects, indexed by j	P_{ij}	The set of predecessors of a_{ij}
N	The number of projects in the portfolio	CP_j	The critical path length of project j
n	The number of non-dummy activities in the portfolio		
ρ_j	The release date of project j	*Schedule metrics*	
δ_j	The due date of project j	$f_{ij} = s_{ij} + a_{ij}$	The finish time of a_{ij}
I_j	The set of activities of project j, indexed by i	$S_j = s_{0j}$	The start time of project j
n_j	The number of non-dummy activities in project j	$F_j = f_{(n_j+1)j}$	The finish time of project j
a_{ij}	Activity i of project j	$M_j = F_j - S_j$	The makespan of project j
d_{ij}	The duration of a_{ij}	$D_j = F_j - \delta_j$	The delay of project j, $D_j \geq 0$
a_{0j}	The dummy-start activity of project j		
$a_{(n_j+1)j}$	The dummy-end activity of project j	*Decision variables*	
		s_{ij}	The start time of a_{ij}
Resource parameters			
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		
TWK_k	The total resource requirement of resource k by each activity of each project		
TWK_{jk}	The total resource requirement of resource k by each activity of project j		

Table 3
Overview of optimisation criteria and due dates.

Optimisation criteria		
Makespan objectives		
TPM	Total Portfolio Makespan	$max_{j\in J} F_j - min_{j\in J} S_j$
APM	Average Portfolio Makespan	$\frac{1}{J}\sum_{j\in J} F_j - S_j$
Tardiness objectives		
APD	Average Project Delay	$\frac{1}{J}\sum_{j\in J} D_j$
ARG	Average Relative Gap	$\frac{1}{J}\sum_{j\in J} \frac{D_j}{\delta_j - \rho_j}$
SPD	Squared Project Delay	$\frac{1}{J}\sum_{j\in J} (D_j)^2$
MaxPD	Maximum Project Delay	$max_{j\in J} D_j$
MaxRG	Maximum Relative Gap	$max_{j\in J} \frac{D_j}{\delta_j - \rho_j}$
Due dates		
Network-based due dates		
CP1	100% Critical Path length	$\delta_j = (1 \times CP_j) + \overline{\rho}_j, \forall j \in J$
CP2	200% Critical Path length	$\delta_j = (2 \times CP_j) + \overline{\rho}_j, \forall j \in J$
CP3	300% Critical Path length	$\delta_j = (3 \times CP_j) + \overline{\rho}_j, \forall j \in J$
Resource-based due dates		
RLB1	Portfolio-level Resource Lower Bound	$\delta_j = \frac{TWK_{\overline{k}}}{R_{\overline{k}}}, \forall j \in J$
RLB2	Project-adjusted Resource Lower Bound	$\delta_j = \frac{TWK_{j\overline{k}}}{(\frac{TWK_{\overline{k}}}{N})} \times \frac{TWK_{\overline{k}}}{R_{\overline{k}}}, \forall j \in J$

the average position ap_j for each project j based on the indirect AS representation. For the first project, this is achieved by taking the average of its positional indexes v_{i1}, $(\frac{14+16+\cdots+30}{10})$, which is equal to 24.1. The average position for the second and third project is equal to 15.4 and 7.0, respectively. The statistical variance of these three averages is computed by applying Eq. (2), which results in a variance value of 48.74. The maximal variance value attainable, V_{max}, is calculated by constructing any activity sequence where the projects follow a strict order. In the example in Fig. 1, the calculation is performed for an activity sequence where the activities are sorted by decreasing project index (3,2,1). It should be noted that any strict project order leads to the same variance value when an identical number of non-dummy activities for each project is assumed. As a result, the average position is equal to, $(\frac{21+22+\cdots+30}{10})$, 25.5 for the first project and 15.5 and 5.5 for the second and third project, respectively. The variance of these three numbers is equal to 66.67. As a result, by dividing the variance of the left-hand schedule over the theoretical maximal variance, $(\frac{48.74}{66.67})$, its PPL is obtained and is equal to 0.73. A similar calculation can be repeated for the right-hand schedule which leads to a PPL of 0.03.

3.3. Rationale of resource-buffered scheduling strategies

Resource-buffered scheduling is expected to be beneficial for three reasons. First of all, it is observed that, in schedules with a high PPL,

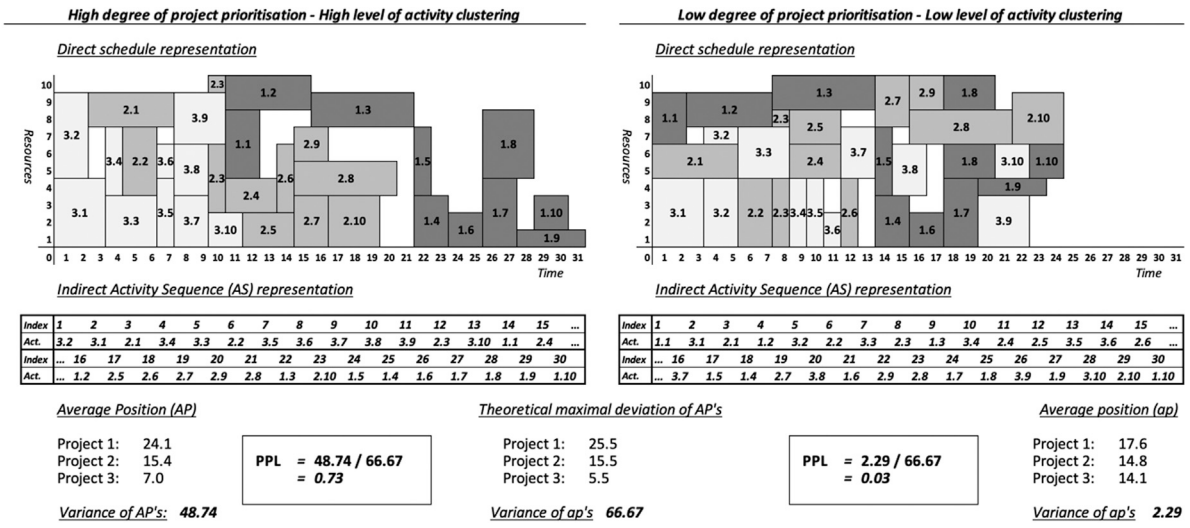

Fig. 1. Computation of the PPL metric for schedules with a high (left) and low (right) degree of project prioritisation.

the most prioritised projects exhibit very unevenly distributed resource consumption patterns across their makespan. In other words, significant peaks in the resource consumption of one or multiple types or resources can be identified. Furthermore, it is observed that a relatively larger proportion of the resources before and after these peaks remains unused because the projects with a lower priority did not have sufficient precedence-feasible activities with a duration short enough to utilise the resources at the restrictive time-window between those peaks. As a result, a project with a lower priority could be forced to delay a critical activity until a time-point after these resource peaks, which causes the entire project to be delayed. Imposing artificial resource restrictions will, therefore, improve the resource utilisation efficiency by distributing the resource usage of the projects with a high priority more evenly over their makespan. Secondly, when the due dates of the projects are not set at their critical path length and, for this reason, have some slack, the risk that a project with a high priority will become tardy by imposing resource buffers is reduced. For this reason, the proposed strategy of imposing resource buffers is expected to be more advantageous in situations where the due dates are more lax and evenly distributed. Finally, in contrast to the time-delayed scheduling strategy employed by Gonçalves et al. (2008), our resource-buffered scheduling approaches are far less restrictive because they only impose one artificial scheduling restriction whereas the former imposes one per activity. A lower number of artificial restrictions increases the likelihood that more optimal settings are predetermined. Additionally, in contrast to time-delayed scheduling, more freedom is given to the scheduling algorithm to assign activities to their earliest feasible start time which is expected to reduce the cost of overestimating the optimal buffer size.

3.4. Resource-buffered scheduling approaches

The two resource-buffered scheduling strategies in our discussion differ in the hierarchic level, either project or portfolio, at which the resource buffer is enforced and the static versus dynamic nature of this resource buffer. We commence by introducing two novel concepts that are present in both of these strategies. First of all, for both approaches, the *Most Critical Resource (MCR)* among the resource types in the portfolio needs to be defined. The MCR is defined as the resource type k with the maximal ratio of Total Work Content (TWK_k) over the entire portfolio divided by its total renewable availability (R_k). Throughout the remainder of this manuscript, the MCR is mathematically represented by $\overline{k}$. Secondly, each resource-buffered scheduling variant defines one artificial *resource restriction* that acts as an additional scheduling constraint on either the project- or the portfolio-level, depending on the specific strategy. A resource restriction can be interpreted as the inverse of a resource buffer because it defines the number of resources that a project or portfolio is allowed to use whereas a buffer defines the number of resources that cannot be used. For both variants, this artificial resource restriction is only generated for the MCR, $\overline{k}$.

The first strategy generates a resource restriction $rr_{j\overline{k}}$ for a single project j in the portfolio and this restriction remains fixed during the scheduling procedure. The second strategy defines a resource restriction $rr_{\overline{k}}$ that restricts all the activities of the entire portfolio and gradually reduces this restriction to zero by the time a target iteration $T_{\overline{k}}$ in the schedule generation scheme is encountered. For this reason, the former approach is termed the *fixed-project resource restriction strategy* while we refer to the latter as the *variable-portfolio resource restriction strategy*.

3.4.1. The fixed-project resource restriction strategy

In this scheduling approach, an artificial resource restriction $rr_{j\overline{k}}$ is defined for the project j with the largest project earliness. This artificial scheduling constraint enforces that the sum of the resource requirement of every activity of this project j is not larger than the resource restriction $rr_{j\overline{k}}$ at any point t in the schedule for the MCR. This project-level resource restriction does not replace the portfolio-level resource availability constraints, although it behaves in a similar manner. As a result, for the project j with the largest project earliness, two different types of resource constraints have to be jointly respected, both on the portfolio-level and on the project-level.

This artificial scheduling restriction can only lead to feasible solutions if it is defined within a feasible range. First of all, a project-level resource restriction may not be smaller than the largest resource requirement of the MCR of any activity in that project. Secondly, while not a feasibility issue, a project-level resource restriction larger than, or equal to the total, shared resource availability of the MCR does not further constrain the problem and should be avoided. This feasible range is mathematically given in Eq. (4). For ease of representation, we denote these feasible minimum and maximum values as $\underline{rr_{j\overline{k}}}$ and $\overline{rr_{j\overline{k}}}$.

$$rr_{j\overline{k}} \in [\underline{rr_{j\overline{k}}}, \overline{rr_{j\overline{k}}}] = [\max_{i \in j}(r_{ij\overline{k}}), R_{\overline{k}}], \forall j \in J \tag{4}$$

A fixed-project resource restriction is determined within this feasible range for the project with the largest slack towards its due date. A feasible solution can then be generated in combination with a given indirect solution representation and the serial or parallel SGS that respects this additional artificial fixed-project resource restriction. As a result, the appropriate a priori determination of this fixed-project

resource restriction is key to the effectiveness of this resource-buffered scheduling approach and will be analysed in Section 5 by further restricting this feasible range.

3.4.2. The variable-portfolio resource restriction strategy

In this scheduling approach, an artificial resource restriction $rr_{\bar{k}}$ is generated that applies to the entire portfolio identically. Similar to the previous strategy, this scheduling constraint enforces that the sum of the resource requirements of all the activities of every project in the portfolio are not larger than the resource restriction $rr_{\bar{k}}$ at any time t in the schedule. In contrast, however, because this scheduling restriction does apply to the entire portfolio and should by definition not be larger than the resource availability $R_{\bar{k}}$, it does effectively replace the shared resource availability constraint $R_{\bar{k}}$.

Just like the project-level resource restriction of the previous approach, this portfolio-level artificial scheduling restriction can only lead to feasible solutions when it is defined within a feasible range. It cannot be smaller than the largest resource requirement of the MCR of any activity in the portfolio and should never be larger than the total, shared resource availability of the MCR. This feasible range is mathematically presented in Eq. (5) and the minimum and maximum values are denoted as $\underline{rr_{\bar{k}}}$ and $\overline{rr_{\bar{k}}}$.

$$rr_{\bar{k}} \in [\underline{rr_{\bar{k}}}, \overline{rr_{\bar{k}}}] = [\max_{j \in J}(\max_{i \in j}(r_{ij\bar{k}})), R_{\bar{k}}] \tag{5}$$

Another difference of this approach is that the portfolio-level resource restriction is dynamic. The exogenously set initial value $rr_{\bar{k}}$ is gradually increased and becomes equal to $R_{\bar{k}}$ when a predetermined target iteration $T_{\bar{k}}$ is reached. In other words, the value of the resource restriction $rr_{\bar{k}}$ is increased with some linear or non-linear decrement each time a new activity has been scheduled. At the target iteration $T_{\bar{k}}$, the variable portfolio-level restriction reaches the same value as the shared resource constraint $R_{\bar{k}}$, at which point the restriction remains constant at this value for the remainder of the scheduling procedure. Similar to the resource restrictions, the target iteration $T_{\bar{k}}$ also has a feasible range defined in Eq. (6). The minimal target iteration $\underline{T_{\bar{k}}}$ is equal to one, which means that the variable resource restriction is increased to $R_{\bar{k}}$ once the first activity has been scheduled. In the other extreme case, the maximal target iteration $\overline{T_{\bar{k}}}$ means the variable resource restriction $rr_{\bar{k}}$ only reaches $R_{\bar{k}}$ once the final non-dummy activity of the portfolio has been scheduled. The maximal target iteration $\overline{T_{\bar{k}}}$ is, therefore, equal to n.

$$T_{\bar{k}} \in [\underline{T_{\bar{k}}}, \overline{T_{\bar{k}}}] = [1, n] \tag{6}$$

Both an initial portfolio-level resource restriction and a target iteration are determined within the aforementioned feasible ranges. In combination with a given indirect solution representation and a schedule generation scheme that respects this additional scheduling restriction, a feasible solution can then be constructed. Key to the performance of this scheduling strategy is, therefore, the a priori determination of the initial portfolio-level resource restriction and the target iteration. This will be analysed in Section 5 by further restricting these feasible ranges.

4. Genetic algorithm with resource buffers

This section provides a detailed overview of the genetic algorithm with resource buffers. In Section 4.1, the solution representation maintained throughout the algorithm is discussed. Section 4.2 elaborates on generating a diverse and competitive set of initial solutions for the metaheuristic algorithm. Section 4.3 presents the operators of the genetic algorithm, embedding our novel scheduling approaches. Finally, Section 4.4 focuses on implementing different schedule generation approaches, which are core elements of our metaheuristic algorithm. Finally, to support the discussion of our algorithm, Fig. 2 provides an overview of its principal components and an overview of its parameters can be found in Table 6.

4.1. Solution representation

The solution representation in a genetic algorithm greatly influences the algorithm's ability to converge towards (near-)optimal solutions. In this study, we adopt the random key (RK) solution representation, which has shown promising results for the RCMPSP. This indirect solution representation assigns a fractional value between 0 and 1 to each of the activities where a smaller value indicates a higher scheduling priority of the activity. As mentioned earlier, the RK solution representation tends to converge to solutions with an approximate ordering of projects in the resulting schedule. This characteristic contributes to an effective solution structure, characterised by a larger PPL-value, that minimises the average project delay (APD). It should be noted that for this convergence behaviour to be observed, the dummy-activities should be included in the RK solution representation, which is the case in our implementation.

A key distinction of our algorithm lies in the construction of these indirect RK-solution representations, which are redetermined each time a schedule is generated. To construct the RK-vector, fractional values between 0 and 1 are assigned to each activity, proportional to its start time in the schedule. A similar approach has been conducted by Debels et al. (2006). For instance, the earliest activities in the schedule are assigned a value of 0, while activities with the largest start time receive a value of 1. Consequently, activities with start times halfway between the earliest and the latest are assigned a value of 0.50, and so forth. This redetermination of the RK-vector based on their direct solution representations is expected to increase the convergence capability of our algorithm. More importantly, however, it establishes a comparable indirect solution representation base, which allows for the regeneration of the schedule using a standard serial schedule generation scheme, even if it was originally generated using one of the resource-buffered scheduling approaches.

4.2. Initial population

It is known that the performance of metaheuristic algorithms depends significantly on the quality of the initial set of solutions (Bredael & Vanhoucke, 2023). Consequently, in our study, we acknowledge the importance of carefully constructing a diverse and competitive set of initial solutions, equivalent to the population size of the evolutionary algorithm. To accomplish this, we employ a combination of five methods, with each method contributing an equal proportion of these initial solutions. Two of these methods include distinct classifications of priority rule (PR) heuristics that have demonstrated effectiveness in various studies (Browning & Yassine, 2010; Van Eynde & Vanhoucke, 2020). These priority rule heuristics, along with other solution generation methods, are elaborated on in the discussion below.

Static priority rules: The first 20% solutions within the initial set of solutions is composed of the top solutions created by priority rule (PR) heuristics that are both decoupled and static. Decoupled priority rules are characterised by an hierarchically related project- and activity-PR duo. The former describes the order of the project while the latter describes the order of the activities within those projects. Static refers to the fact that the priority value does not change while the schedule is being generated. Table 4 lists the 10 project- and 12 activity-PR's that are employed to generate a total of 120 schedules. In case two activities (projects) tie on a certain metric, the activity (project) with the lowest activity (project) number is prioritised. We refer to the available studies on PR's for the RCMPSP for a more detailed description and analysis of these project- and the activity-PR's (Browning & Yassine, 2010; Mittal & Kanda, 2009; Van Eynde & Vanhoucke, 2020).

Dynamic priority rules: An additional 20% of the initial set is composed of the best solutions constructed by the priority rule combinations presented in Table 5. The five project-PR's presented are dynamic which means that their value and, therefore, priority could change

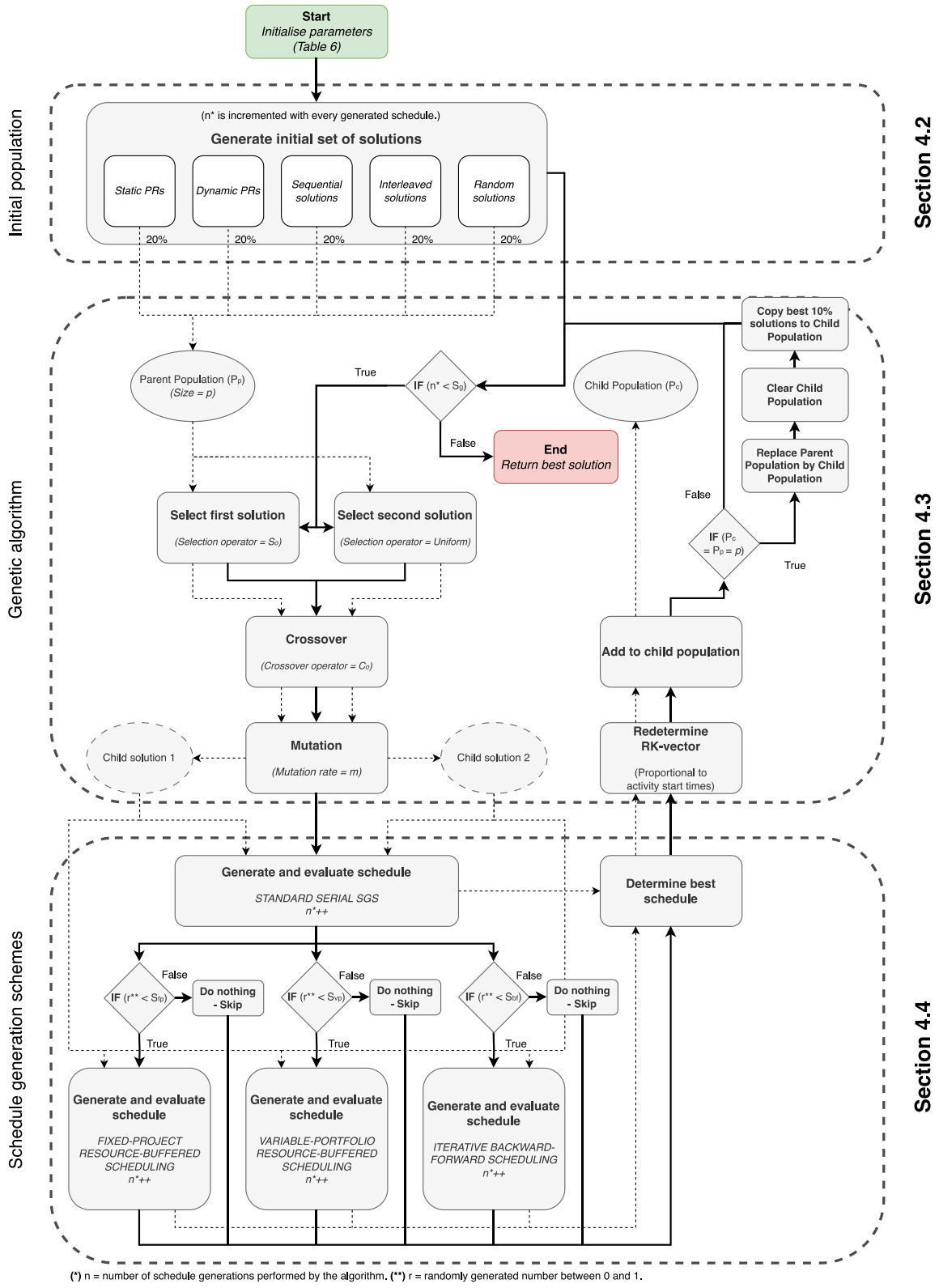

Fig. 2. Overview of genetic algorithm with resource buffers.

while the schedule is being constructed. For example, the 'max. expected project makespan' and 'max. expected project delay' PR's keep track of the highest finish time among the scheduled activities and the remaining critical path length among the unscheduled activities to estimate the expected project makespan or delay, respectively. To this purpose, the latter priority rule additionally subtracts the project's due date from its expected makespan to obtain a lower bound estimate of its expected project delay. The 12 activity-PRs that are used to define the order of the activities within the project are identical to those in Table 4 and, therefore, static in nature. Because a total of 60 solutions is generated in this step, the maximum population size is limited to 300 solutions.

Table 4
Overview of decoupled and static priority rule heuristics for the initial set of solutions.

Project-PR's		Activity-PR's	
Abbrev.	Description	Abbrev.	Description
Min. CP_j	Min. critical path length	Min. ES_{ij}	Min. earliest start time
Max. CP_j	Max. critical path length	Max. ES_{ij}	Max. earliest start time
Min. SP_j	Min. serial/parallel-indicator	Min. LS_{ij}	Min. latest start time
Max. SP_j	Max. serial/parallel-indicator	Max. LS_{ij}	Max. latest start time
Min. TWK_j	Min. project total work content of all resources	Min. EF_{ij}	Min. earliest finish time
Max. TWK_j	Max. project total work content of all resources	Max. EF_{ij}	Max. earliest finish time
Min. ρ_j	Min. project due date	Min. LF_{ij}	Min. latest finish time
Max. ρ_j	Max. project due date	Max. LF_{ij}	Max. latest finish time
Min. $TWK_{j\bar{k}}$	Min. project total work content of the MCR	Min. $Slack_{ij}$	Min. activity slack
Max. $TWK_{j\bar{k}}$	Max. project total work content of the MCR	Max. $Slack_{ij}$	Max. activity slack
		Min. d_{ij}	Min. activity duration
		Max. d_{ij}	Max. activity duration

Table 5
Overview of decoupled and dynamic priority rule heuristics for the initial set of solutions.

Project-PR's		Activity-PR's	
Abbrev.	Description	Abbrev.	Description
Max. rem. CP_j	Max. remaining critical path length	+ 12 static activity-PR's from first stage, see Table 4.	
Max. rem. TWK_j	Max. remaining project total work content of all resources)		
Max. rem. $TWK_{j\bar{k}}$	Max. remaining project total work content of the MCR		
Max. expected M_j	Max. expected project makespan		
Max. expected D_j	Max. expected project delay		

To further diversify the initial solution set, we employ three additional approaches. The third and fourth groups of initial solutions are generated by scheduling the projects based on a uniformly randomly generated sequence of projects using either the *sequential* or *interleaved* approach, respectively. In the sequential approach, activities from the project with the highest priority are scheduled first, followed by the activities of subsequent projects. On the other hand, the interleaved approach schedules one activity at a time for each project, progressing at an equal pace. For a more detailed understanding of these approaches, we refer to the study by Wauters et al. (2015). Finally, the remaining 20% of solutions are generated by constructing RK-vectors that are randomly generated. It should be noted that the serial schedule generation scheme is utilised for all five solution generation methods mentioned above and every schedule generated in this initialisation step is counted towards the stop criterion (S_g) of the subsequent genetic algorithm, as is mentioned in Fig. 2 as well.

4.3. Genetic algorithm

This section presents the genetic algorithm employed in this study. The algorithm consists of two distinct populations: the *parent* population (P_p), which at first represents the initial solutions, and the *child* population (P_c), comprised of newly generated solutions. The evolutionary process can be summarised as follows: Initially, the top 10% best solutions from the parent population are directly copied into the child population. From the parent population, two solutions are selected using distinct selection operators, one for each solution. By applying a crossover operator, two novel RK-vectors are created by recombining the RK-vectors of the selected solutions. Subsequently, a mutation operation is performed on each of these newly generated solutions, with the probability of mutation determined by the parameter m. The resulting child RK-vectors are evaluated by at least one and up to four different schedule generation approaches to assess the performance of their respective schedules for the objective under analysis. In cases where multiple schedules are generated, only the best schedule is retained. This is achieved by redetermining its RK-vector based on its direct schedule representation, with the RK-values set proportionally to the start times of the constituent activities. The resulting solutions are added to the child population until its size matches that of the parent population, which is predefined by a parameter p. Once this occurs, the child population replaces the parent population, and the entire procedure repeats until a predetermined limit of schedule generations, given by the parameter S_g, is attained, upon which the best solution is returned by the algorithm, as visualised in Fig. 2. The remainder of this section discusses the different selection, crossover and mutation operators.

Selection operators: In each iteration of the genetic algorithm, the process of selecting two solutions from the parent population is crucial. The first parent solution is consistently chosen using the *uniform* selection operator, which assigns an equal probability to each solution. The selection of the second parent solution involves either the *uniform* selection operator or one of two variants of the *two-round tournament* selection operator. The first variant of the tournament selection operator, known as *Tournament-fitness*, randomly selects two solutions and retains the one with the best performance on the objective under analysis. This process is repeated in a second round, where again the best solution is preserved. The second variant, called *Tournament-PPL*, operates similarly to the first variant but compares the solutions based on their PPL-score, specifically seeking the one closest to the PPL-score of the incumbent best solution found thus far by the algorithm. The selection of the most effective operator is explored through experimentation in Section 5.

Crossover operators: Once two solutions have been selected, the genetic algorithm employs one of the widely-used crossover operators to generate two new solutions by manipulating their RK-vectors. The choice of using either the *one-point*, *two-point*, or *uniform* crossover operator is examined through experimentation in Section 5.

Mutation operator: First implemented in the P16 algorithm by Pérez et al. (2016), a mutation operator is executed with probability m. This mutation operator generates, for every activity, a new random key value with a probability of 2% for each individual activity.

4.4. Schedule generation schemes

The core of our genetic algorithm with resource buffers is defined by the novel resource-buffered scheduling approaches, which serve as

alternative schedule generation schemes. In this section, we discuss the four different schedule generation schemes present in our algorithm. The first scheme, which is always executed, is the standard serial schedule generation scheme and plans each activity sequentially at its earliest precedence- and resource-feasible start time according to the order defined by the RK-vector. The second scheduling method implements the *fixed-project* resource restriction strategy (Section 3.4.1), with its execution probability determined by the hyperparameter S_{fp}. The third scheduling method is an implementation of the *variable-portfolio* resource restriction strategy (Section 3.4.2), with its execution probability determined by the hyperparameter S_{vp}. The fourth scheduling method is the iterative backward–forward scheduling proposed by Li and Willis (1992), with its execution probability determined by the hyperparameter S_{bf}. Finally, a number of additional hyperparameters further influence the behaviour of the resource-buffered scheduling approaches as detailed below.

Fixed-project resource-buffered scheduling: Two parameters x_1 and x_2 ($x_1 < x_2$) between 0 and 1 further restrict the feasible range of the fixed-project resource restriction as described in Eq. (7). As a result, when x_1 equals 0 and x_2 equals 1, the entire feasible range is considered. Increasing x_1 increases the lower bound, while decreasing x_2 decreases the upper bound of this feasible range.

$$rr_{j\bar{k}} \in [\underline{rr_{j\bar{k}}} + x_1(\overline{rr_{j\bar{k}}} - \underline{rr_{j\bar{k}}}) \ , \ \underline{rr_{j\bar{k}}} + x_2(\overline{rr_{j\bar{k}}} - \underline{rr_{j\bar{k}}})] \tag{7}$$

Variable-portfolio resource-buffered scheduling: Two parameters y_1 and y_2 ($y_1 < y_2$) between 0 and 1 further restrict the feasible range of the variable-portfolio resource restriction. Additionally, two parameters z_1 and z_2 ($z_1 < z_2$) between 0 and 1 further restrict the feasible range of the target iteration. The former is described in Eq. (8) while the formula for the latter can be found in Eq. (9).

$$rr_{\bar{k}} \in [\underline{rr_{\bar{k}}} + y_1(\overline{rr_{\bar{k}}} - \underline{rr_{\bar{k}}}) \ , \ \underline{rr_{\bar{k}}} + y_2(\overline{rr_{\bar{k}}} - \underline{rr_{\bar{k}}})] \tag{8}$$

$$T_{\bar{k}} \in [\underline{T_{\bar{k}}} + z_1(\overline{T_{\bar{k}}} - \underline{T_{\bar{k}}}) \ , \underline{T_{\bar{k}}} + z_2(\overline{T_{\bar{k}}} - \underline{T_{\bar{k}}})] \tag{9}$$

The variable-portfolio resource restriction is increased and becomes equal to $R_{\bar{k}}$ once the target iteration is attained according to a certain *reduction mode* R_m. We distinguish three variants of these reduction modes which are subjected to experimentation in Section 5. The first variant, called the *linear* reduction mode, gradually increases the resource restriction at a fixed rate given by $\frac{R_{\bar{k}} - rr_{\bar{k}}}{T_{\bar{k}}}$. At the target iteration $T_{\bar{k}}$, the resource restriction is maintained constant at $R_{\bar{k}}$. The second variant, known as the *concave* reduction mode, initially exhibits a slower increase in the resource restriction. This increase rate is defined by $\frac{R_{\bar{k}} - rr_{\bar{k}}}{T_{\bar{k}}} * \frac{2i}{T_{\bar{k}}}$ where i is the current number of activities scheduled by the algorithm. Once the resource restriction attains the value of $R_{\bar{k}}$, it is remained constant at this level. Finally, the *constant* reduction mode maintains the resource restriction at its original value until the target iteration is attained. Subsequently, the resource restriction is immediately increased to the value of $R_{\bar{k}}$.

The algorithm follows a systematic process comprising several steps that are summarised in Fig. 2. Initially, the standard serial schedule generation scheme is applied to the given RK-vector, and its performance is evaluated, resulting in a baseline schedule. Subsequently, the selection of alternative schedule generation methods depends on the probabilities associated with each method. If the fixed-project resource-buffered scheduling variant is chosen, a project-level resource restriction is randomly determined within a feasible range defined by parameters x_1 and x_2, specifically targeting the project with the largest earliness in the baseline schedule. In cases where no project is early, an artificial restriction is imposed on the project with the smallest delay. On the other hand, if the variable-portfolio resource-buffered scheduling variant is selected, both a portfolio-level resource restriction and a target iteration are randomly determined within feasible ranges specified by parameters y_1, y_2, z_1, and z_2. For either of these approaches, the resource-buffered schedule is then generated with an adapted serial schedule generation scheme that adheres to this additional artificial scheduling restriction and its fitness is evaluated. In the case of the iterative backward–forward schedule generation scheme, the algorithm proceeds by conducting backward and forward scheduling passes until no further improvement in the objective can be achieved. At this point, the algorithm returns the best schedule obtained. If any of the schedules generated by the three additional scheduling methods outperform the baseline schedule, it replaces the baseline schedule. The RK-vector associated with the improved schedule is recalculated based on the activity start times, and this solution is stored in the child population.

5. Computational experiments

In this section, our primary focus is on optimising the algorithmic parameters of our genetic algorithm with resource buffers for the Average Project Delay (APD) optimisation criterion and the portfolio-level Resource Lower Bound (RLB1) due date. The APD criterion has gained widespread adoption in the literature for the RCMPSP, making it a suitable objective for our algorithm's optimisation. However, rather than relying on the commonly used project-level critical path length (CP1) metric for assigning due dates to projects, we chose to optimise our algorithm specifically for the RLB1 due date. The RLB1 due date assigns an identical, and generally laxer, due date to each project, allowing for potential project earliness. We hypothesise that our algorithm will exhibit superior effectiveness in optimising objectives for this type of due date. For ease of comprehension, an overview of all the parameters of the metaheuristic algorithm that are subjected to optimisation is presented in Table 6.

Section 5.1 describes the optimisation process for the parameters of the evolutionary algorithm. Furthermore, Section 5.2 summarises the parameter optimisation specifically related to the resource-buffered scheduling approaches embedded within the genetic algorithm. The optimisation of the frequency of the different schedule generation methods is the subject of Section 5.3. To alleviate the computational burden and prevent overfitting of our algorithmic parameters, the optimisation process discussed in these sections is conducted using the first set of the MPLIB1 dataset. This dataset comprises 833 RCMPSP instances, each containing 6 projects and 60 non-dummy activities per project. Section 5.4 presents the convergence curves of our algorithm and highlights several observations pertaining to the integration of the resource-buffered scheduling approaches. An in-depth overview of our algorithm's performance is provided in Section 5.5. Additionally, we compare the results of our algorithm with the combined best-known solutions generated by ten competing metaheuristic algorithms across all 4550 test instances of the MPLIB1 dataset, considering multiple optimisation criteria and due dates. Finally, in Section 5.6, we discuss the impact of various problem instance metrics on the relative performance of our algorithm. We also explore the structure of the schedule, as measured by the Project Portfolio Level (PPL) metric, described in Section 3.2.

5.1. Optimisation of evolutionary parameters

In our optimisation approach, the parameters of the evolutionary algorithm are optimised before any of the other parameters. This requires fixing the non-evolutionary parameters at a reasonable value. To achieve this, the resource-buffered scheduling parameters are fixed by setting x_1 and y_1 at 0 and x_2 and y_2 at 1, indicating that the full range of feasible resource restrictions is allowed. Similarly, the z_1 and z_2 parameters that determine the range of the target iteration are set at 0 and 1, respectively. Furthermore, the parameters that determine the frequency of the different schedule generation methods S_{fp}, S_{vp} and S_{bf} are set at 0.50 meaning that each of these schedule generation methods is executed with a probability of 50% for every newly generated RK-vector.

Table 6
Overview of algorithmic parameters.

Parameter	Values	Description	Section
Evolutionary			
p	100; 150; 200; 250; 300	Population size	5.1
m	0.10; 0.15; 0.20; 0.25; 0.30	Mutation rate	5.1
S_o	Tournament-fitness; Tournament-PPL; Uniform	Selection operator	5.1
C_o	One-point; Two-point; Uniform	Crossover operator	5.1
S_g	10,000 - 100,000	Stop criterion: number of schedule generations	5.4
Fixed-project RBS			
x_1	0; 0.2; 0.4; 0.6; 0.8 ($< x_2$)	Relative lower limit on project-level resource restriction	5.2.1
x_2	0.2; 0.4; 0.6; 0.8; 1 ($> x_1$)	Relative upper limit on project-level resource restriction	5.2.1
Variable-portfolio RBS			
y_1	0; 0.2; 0.4; 0.6; 0.8 ($< y_2$)	Relative lower limit on portfolio-level resource restriction	5.2.2
y_2	0.2; 0.4; 0.6; 0.8; 1 ($> y_1$)	Relative upper limit on portfolio-level resource restriction	5.2.2
z_1	0; 0.2; 0.4; 0.6; 0.8 ($< z_2$)	Relative lower limit on target iteration	5.2.2
z_2	0.2; 0.4; 0.6; 0.8; 1 ($> z_1$)	Relative upper limit on target iteration	5.2.2
R_m	Linear; Concave; Constant	Resource restriction reduction method	5.2.2
Frequency of schedule generation methods			
S_{fp}	0; 0.25; 0.50; 0.75; 1	Frequency of 'fixed-project' schedule generation method	5.3
S_{vp}	0; 0.25; 0.50; 0.75; 1	Frequency of 'variable-portfolio' schedule generation method	5.3
S_{bf}	0; 0.25; 0.50; 0.75; 1	Frequency of 'backward–forward' schedule generation method	5.3

Table 7
Orthogonal experiment on the evolutionary algorithm.

Population size (p)	Mutation rate (m)	Selection operator (S_o)	Crossover operator (C_o)	APD - RLB1
250	0.30	Tournament-fitness	Two-point	34.54
300	0.10	Tournament-fitness	One-point	34.79
250	0.15	Tournament-fitness	One-point	34.82
150	0.25	Tournament-fitness	One-point	35.01
200	0.20	Tournament-fitness	One-point	35.10
100	0,15	Tournament-fitness	Two-point	35.35
150	0,10	Tournament-fitness	Uniform	35.77
100	0.10	Uniform	One-point	36.26
200	0.10	Tournament-PPL	Two-point	36.41
150	0.15	Tournament-PPL	One-point	36.56
100	0.30	Tournament-fitness	Uniform	36.57
300	0.20	Uniform	One-point	37.42
250	0.25	Uniform	One-point	37.54
100	0.25	Uniform	Two-point	37.57
200	0.30	Uniform	One-point	37.74
250	0.20	Tournament-PPL	Two-point	37.75
150	0.30	Tournament-PPL	Two-point	37.96
300	0.25	Tournament-fitness	Uniform	38.07
300	0.30	Tournament-PPL	Two-point	38.49
150	0.20	Uniform	Uniform	43.35
100	0.20	Tournament-PPL	Uniform	43.44
200	0.15	Uniform	Uniform	43.51
250	0.10	Uniform	Uniform	43.75
300	0,15	Tournament-PPL	Uniform	44.08
200	0.25	Tournament-PPL	Uniform	44.10

Subsequently, the four remaining evolutionary parameters are optimised. To conduct this optimisation, we employ an orthogonal experiment described in Table 7. This experiment comprises 25 runs, where we perform a full-factorial test of the population size and mutation rate in combination with different selection and crossover operators. We ensure that each combination of evolutionary operators is tested at least once, and every operator is individually tested in conjunction with each population size value and mutation rate value. The results of the experiment are sorted from best (top) to worst (bottom) and quickly reveal the significant superiority of the tournament-fitness selection operator compared to the other two selection operators. It is expected that the tournament-fitness operator outperforms the uniform selection operator since it tends to select solutions with better performance, thereby enhancing the algorithm's convergence behaviour. However, interestingly, the experiment also highlights that the PPL-metric fails to serve as a reliable proxy for solution quality, as it does not align with the performance of the tournament-fitness selection operator. Furthermore, while the uniform crossover operator is significantly outperformed, the results are less conclusive in differentiating between the one-point and two-point crossover operators. To address this, we conducted a full-factorial experiment for both crossover operators, considering all values of the population size and mutation rate in combination with the tournament-fitness selection operator. The results revealed a slight advantage for the two-point crossover operator over the one-point crossover operator. The best performance was observed with a population size of 250 and a mutation rate of 0.20, resulting

Table 8
Optimisation of the limits on the feasible range for the fixed-project resource restriction.

		Best: $x_1 = 0.00$, $x_2 = 0.20$				
x_2	1.00	36.86	36.91	36.97	37.16	37.26
	0.80	36.79	36.83	36.93	37.06	–
	0.60	36.81	36.81	36.93	–	–
	0.40	36.84	36.88	–	–	–
	0.20	**36.78**	–	–	–	–
APD - RLB1		**0.00**	0.20	0.40	0.60	0.80
				x_1		

in an APD of 34.49. Consequently, these values are established as the optimal settings for the evolutionary algorithm.

5.2. Optimisation of resource-buffered scheduling approaches

Section 5.2.1 discusses the parameter optimisation for the fixed-project resource-buffered scheduling technique, while Section 5.2.2 analyses the optimal parameter values for the variable-portfolio approach.

5.2.1. Optimisation of fixed-project approach

As mentioned earlier, the performance of the fixed-project resource-buffered scheduling method depends on two parameters, x_1 and x_2, that define the range of resource restrictions that could be imposed on the project with the largest earliness. Table 8 compares the performance of this resource buffered scheduling technique for different ranges of resource restrictions. In this experiment, the evolutionary parameters of the algorithm are set at their optimal value as defined in the preceding section. Additionally, the S_{fp} parameter was set at 1, while the S_{vp} and S_{bf} parameters were set at 0, indicating that only the standard serial schedule generation scheme and the fixed-project schedule generation method are executed for every new solution. The results indicate that large resource restrictions are preferred over small ones, as is indicated by the large performance differences between the bottom-left and upper-right performance score. Moreover, excluding the largest resource restrictions from the interval (larger x_1) leads to more significant performance differences than excluding the smallest resource restrictions (smaller x_2). The optimal setting was detected with x_1 equal to 0 and x_2 equal to 0.20.

5.2.2. Optimisation of variable-portfolio approach

Similarly to the previous approach, the variable-portfolio resource-buffered scheduling method also relies on two parameters y_1 and y_2 that restrict the feasible range of the dynamic, portfolio-level resource buffer. Table 9 lists the average performance values for different y_1 and y_2 parameter settings over all three reduction modes R_m. The S_{vp} parameter was set at 1, while the S_{fp} and S_{bf} parameters were set at 0, such that only the variable-portfolio approach is executed. The best performance was encountered when y_1 is set to 0.60 and y_2 is set to 1, indicating that laxer portfolio-level resource restrictions achieved better performance. Furthermore, to identify the best target iteration of the resource restriction, Table 10 presents the average performance values, tested over all three reduction modes R_m, for different z_1 and z_2 values, which were set to 0 and 1, respectively, in the previous experiment. This experiment is conducted using the best, previously identified, y_1 and y_2 values. The best performance was encountered when the z_1- and z_2-parameters were set at 0 and 0.60, respectively. This suggests that it is preferable to increase the resource restriction to its original level well before the final activities are scheduled, enabling optimal resource utilisation. Additionally, both experiments revealed that the impact of the reduction mode R_m was negligible. However, the linear reduction mode yielded the best performance with an APD of 37.01.

5.3. Optimisation of frequency of schedule generation methods

After determining the optimal parameter settings for both the evolutionary algorithm and the resource-buffered scheduling methods, we proceed to analyse the ideal frequency at which each of these methods should be executed for every newly generated RK-vector. Increasing the number of schedule generation methods is likely to enhance the quality of schedules produced from a given RK-vector. However, it is important to consider that each schedule generation contributes to the algorithm's stop criterion. Consequently, a higher frequency of the schedule generation methods limits the number of evaluations of different RK-vectors, thereby reducing the algorithm's convergence capability. In this section, we conduct a comprehensive full-factorial experiment to examine the optimal combination of the S_{fp}, S_{vp} and S_{bf} parameters.

Table 11 presents the average performance across all five values of the S_{bf} parameter, representing the frequency of the iterative backward–forward schedule generation scheme, for all values of the S_{fp} and S_{vp} parameters, which correspond to the fixed-project and variable-portfolio schedule generation approaches, respectively. First of all, although not explicitly stated in the overview provided by Table 11, the performance of the algorithm is heavily influenced by the S_{bf} parameter. The lowest average performance is observed when it is set at 0, indicating that the iterative backward–forward schedule generation scheme is omitted from the algorithm. This results in an average APD score of 36.77 across all values of S_{fp} and S_{vp}. The best average performance for the S_{bf} parameter is obtained when it is set equal to 0.50 with an average APD value of 34.61. Higher values of this parameter lead to lower average performance scores, likely due to the increased number of schedule generations required by the iterative scheduling approach. Furthermore, while the data exhibits slight oscillation for the different S_{bf} settings, the average performance scores in Table 11 reveal a clear relation between the performance of the algorithm and the frequency of the fixed-project and variable-portfolio schedule generation methods (S_{fp} and S_{vp}, respectively). A higher frequency of the fixed-project schedule generation approach contributes to performance improvements, albeit with diminishing returns. On the other hand, the variable-portfolio schedule generation method reaches its optimal frequency when S_{vp} is set equal to 0.25, indicating its contribution to the algorithm's performance while suggesting a comparatively lower effectiveness compared to the fixed-project approach. The best combination of the three parameters is identified with S_{fp} S_{vp} and S_{bf} at 1.00, 0.25 and 0.50, respectively, resulting in an APD of 34.20.

5.4. Convergence curves

In order to evaluate the effectiveness of the novel resource-buffered scheduling approaches, we examine the convergence curves presented in Fig. 3. The figure illustrates two versions of the metaheuristic algorithm: one incorporating the resource-buffered scheduling approaches and the other without them. In the former version, all algorithm parameters were set to their optimal values, while in the latter version, the S_{fp} and S_{vp} parameters were instead set at 0, thereby excluding both the fixed-project and variable-portfolio resource-buffered scheduling

Table 9
Optimisation of the limits on the feasible range for the variable-portfolio resource restriction.

		Average performance over all R_m				
	Best: $y_1 = 0.60,\ y_2 = 1.00$					
y_2	**1.00**	37.24	37.20	37.26	**37.08**	37.16
	0.80	37.27	37.24	37.23	37.20	–
	0.60	37.32	37.28	37.21	–	–
	0.40	37.28	37.27	–	–	–
	0.20	37.36	–	–	–	–
	APD - RLB1	0.00	0.20	0.40	**0.60**	0.80
				y_1		

Table 10
Optimisation of the limits on the feasible range for the target iteration.

		Average performance over all R_m				
	Best: $z_1 = 0,\ z_2 = 0.60$					
z_2	1.00	37.08	37.24	37.29	37.36	37.39
	0.80	37.10	37.30	37.30	37.45	–
	0.60	**37.06**	37.18	37.29	–	–
	0.40	37.06	37.23	–	–	–
	0.20	37.12	–	–	–	–
	APD - RLB1	**0.00**	0.20	0.40	0.60	0.80
				z_1		

Table 11
Optimisation of frequency of the schedule generation methods.

		Average performance over all five values of S_{bf}				
	Best: $S_{fp} = 1.00,\ S_{vp} = 0.25,\ (S_{bf} = 0.50)$					
S_{vp}	1.00	35.29	35.23	35.22	35.37	35.36
	0.75	35.12	35.01	35.09	35.22	35.20
	0.50	35.06	34.98	34.95	34.94	34.89
	0.25	34.99	34.87	34.80	34.83	**34.80**
	0.00	35.20	34.99	34.98	34.89	34.98
	APD - RLB1	0.00	0.25	0.50	0.75	**1.00**
				S_{fp}		

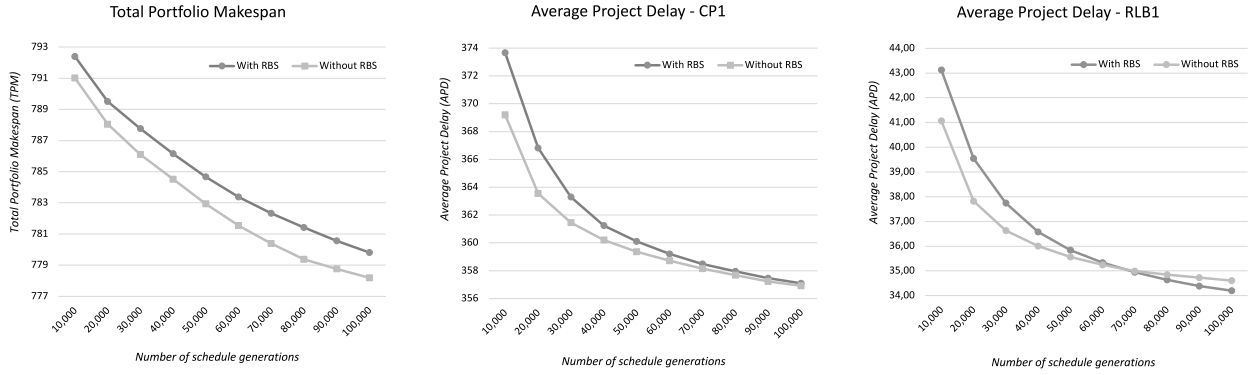

Fig. 3. Convergence curves of the genetic algorithm with and without resource-buffered scheduling (RBS).

approaches. Fig. 3 reports the outcomes of this analysis for three different combination of objective and due date: Total Portfolio Makespan (TPM) and Average Project Delay (APD) on both the CP1 and RLB1 due dates.

It is evident that, regardless of the optimisation criterion, the genetic algorithm incorporating resource buffers exhibits slower convergence compared to the algorithm without these resource-buffered scheduling approaches. This can be attributed to the increased number of schedule generations and evaluations for each newly generated RK-vector, which decelerates the convergence process in the algorithm with resource-buffered scheduling. However, the final performance of these algorithms varies depending on the specific objectives. Regarding the Total Portfolio Makespan (TPM) objective, the algorithm with resource buffers is consistently outperformed and fails to compensate for its slower convergence. In this case, the algorithm without resource buffers achieves superior results. In contrast, when considering the Average Project Delay (APD) objectives, different observations arise. For the CP1 due dates, the algorithm with resource buffers initially lags behind the algorithm without these scheduling approaches. However, as the iterations progress, it catches up and eventually achieves a performance score that closely resembles the algorithm without resource buffers. Conversely, for the RLB1 due dates, the algorithm with resource buffers surpasses the other version of the algorithm at around 70,000 schedule generations and evaluations. Consequently, on

Table 12
Benchmark performance of genetic algorithm with resource buffers at APD - RLB1 optimal settings, 100,000 schedule generations.

	Due date	Opt. criterion	Best existing metaheuristic		Best solutions among 10 existing metaheuristics			B23: RBS - genetic algorithm				New average of best-known solutions			
			Procedure	Average	Average	#Best	#BKS	Average	#Best	%Best	#BKS	All sets	Set 1	Set 2	Set 3
MPLIB1 - All sets		TPM	W15-I	**1056.03**	1053.02	3780	4186	1062.26	364	8.00%	770	1052.42	777.19	944.24	1224.35
		APM	V20	**646.38**	645.45	4100	4214	660.68	336	7.38%	450	645.29	521.67	591.90	725.63
	CP1	APD	V20	**471.35**	470.48	4102	4213	485.70	337	7.41%	448	470.41	348.79	417.06	549.99
		ARG	V20	**2.834**	2.831	4154	4268	2.915	282	6.20%	396	2.831	2.240	2.536	3.240
		SPD	V20	**434,904**	433,935	3446	3543	456,179	1007	22.13%	1103	433,247	229,969	333,750	572,952
		MaxPD	V20	**849.47**	843.98	4158	4320	868.56	230	5.05%	392	843.70	581.45	738.66	1008.79
		MaxRG	V20	**4.095**	4.092	2469	2679	4.132	1871	41.12%	2081	4.067	3.113	3.657	4.685
	RLB1	APD	V20	21.60	18.01	2812	3145	**19.35**	1405	30.88%	1636	17.31	33.47	17.71	11.07
		ARG	V20	0.062	0.057	2042	3008	**0.057**	1542	33.89%	2508	0.055	0.105	0.058	0.035
		SPD	V20	4050	3541	3446	3573	**3828**	977	21.47%	1104	3373	7352	3339	1925
		MaxPD	V20	88.13	84.03	3162	3784	**87.42**	766	16.84%	1388	82.72	114.88	82.97	70.68
		MaxRG	V20	0.151	0.150	2107	2728	**0.149**	1822	40.04%	2443	0.146	0.249	0.155	0.102
MPLIB1 - Set 1	CP2	APD	V20	**211.82**	210.85	539	658	216.85	175	21.01%	294	–	209.93	–	–
		ARG	V20	**0.699**	0.697	570	692	0.717	141	16.93%	263	–	0.695	–	–
		SPD	P16	127,226	125,550	526	632	**126,158**	201	24.13%	307	–	121,393	–	–
		MaxPD	V20	**384.56**	384.15	559	696	393.44	137	16.45%	274	–	382.97	–	–
		MaxRG	V20	**1.092**	1.090	526	691	1.114	142	17.05%	307	–	1.087	–	–
	CP3	APD	V20	**112.43**	111.95	463	719	116.49	114	13.69%	370	–	111.36	–	–
		ARG	V20	**0.259**	0.258	498	762	0.269	71	8.52%	335	–	0.258	–	–
		SPD	P16	58,753	56,557	469	716	**57,321**	117	14.05%	364	–	54,690	–	–
		MaxPD	V20	**216.92**	216.73	461	735	224.66	98	11.76%	372	–	215.74	–	–
		MaxRG	V20	**0.459**	0.458	468	762	0.476	71	8.52%	365	–	0.457	–	–
	RLB2	APD	P16	20.56	18.71	136	511	**17.92**	322	38.66%	697	–	17.35	–	–
		ARG	P16	0.089	0.087	96	505	**0.085**	328	39.38%	737	–	0.085	–	–
		SPD	P16	3,645	3,288	146	545	**3132**	288	34.57%	687	–	3030	–	–
		MaxPD	V20	58.14	54.12	137	577	**51.54**	256	30.73%	696	–	50.01	–	–
		MaxRG	V20	0.183	0.181	95	511	**0.175**	322	38.66%	738	–	0.173	–	–

#Best: The number of solutions better than that of any other algorithm. #BKS: The number of solutions equal to the best-known solution.

the RLB1 due dates, the algorithm with resource buffers demonstrates superior performance over the algorithm without resource buffers.

5.5. Performance of genetic algorithm with resource buffers

The performance of the genetic algorithm with resource buffers is reported on the MPLIB1 dataset (Van Eynde & Vanhoucke, 2020) and compared against the performance of ten existing algorithms that were benchmark in an earlier study (Bredael & Vanhoucke, 2023). However, in this preceding benchmark analysis, the performance of the metaheuristic algorithms was only reported on all three sets of MPLIB1 (4550 instances) for the most commonly used CP1 due date while the performance of the remaining four due dates was only reported on the first set of the MPLIB1 dataset (833 instances). Because our algorithm is likely more effective for multi-project scheduling problems on the RLB1 due date, as shown in Section 5.4, we extended the benchmark analysis to include the entire dataset for this due date. In our analysis, we will focus on the CP1 and RLB1 due dates because these due dates form project-level network- and portfolio-level resource lower bounds on the multi-project tardiness, respectively.

Table 12 shows the benchmark performance of our algorithm using the previously identified optimal parameter values. The stop criteria of our algorithm is equal to 100,000 schedule generations which is identical to that of the previously benchmarked algorithms. The results are reported for all three sets of the MPLIB1 dataset on the CP1 and RLB1 due dates. Table 12 also reports the results on the remaining three due dates for the first set of the MPLIB1 dataset, however, to condense our analysis, these performance measures will not be elaborated on. In this table, the performance averages are an unweighted average of the value of the optimisation criterion for all instances. From left to right, the first average reports the average performance of the best existing metaheuristic algorithm for that combination of objective and due date. The following average lists the average performance of the best solution returned by any of the ten existing metaheuristics for every instance. In other words, it is equivalent to the average of the best-known solutions reported in the earlier benchmark study. Afterwards, the number of times that at least one of the ten metaheuristics in our comparison generated a solution that is better than the solution of our metaheuristic algorithm is reported (#Best), while the following value reports the number of times their solution was better or equally good (#BKS). The following four values then report the actual performance of our genetic algorithm with resource buffers. Similarly, the average performance, number of best solutions (#Best) and number of best or equally good solutions (#BKS) are reported. Finally, we report the new average of best-known solutions generated by any of the 11 metaheuristics such that future studies can compare the performance of their algorithm as well. The results show that our novel algorithm is competitive with the ten existing metaheuristics, however, its relative performance differs depending on the objective and due date, which will be discussed more extensively in Section 5.6. In general, when we consider the performance on every objective and due date combination of equal importance, our algorithm generates new best-known solutions for 20.04% of the test instances.

5.6. Analysis of impact of problem instance metrics

In this section, we focus on the effects of the objectives, due dates and various problem characteristics on the relative performance of our algorithm and the Project Prioritisation Level (PPL) of the best schedule returned by our algorithm. First, we look at the impact of the objectives, due dates and the size of the portfolio. Afterwards, we examine the effect of the distribution of SP-indicators among the projects within the portfolio. Finally, we provide concluding observations on the effects of resource metrics.

Impact of objectives, due dates and portfolio size: Table 13 provides insights into the performance of our algorithm and the structure of the best solution, as measured by the Project Prioritisation Level (PPL) metric. The MPLIB1 dataset comprises three sets, each containing a specific number of projects per instance: 6, 12, or 24 projects. For each set, the percentage of instances in which our algorithm outperformed other algorithms is reported. Additionally, the table presents the average (Avg.), standard deviation (SD), and minimum and maximum values (Range) of the PPL for each combination of portfolio size, optimisation criterion, and due date, generated by our algorithm. It is worth noting that the Total Portfolio Makespan (TPM) and Average Portfolio Makespan (APM) objectives, which do not require project due date specifications, are reported separately, for this reason. The findings highlight that the performance of our algorithm depends on the number of projects in the portfolio, with a more pronounced effect

Table 13
Impact of objectives, due dates and portfolio size on relative algorithmic performance and PPL of the schedules.

Opt. criterion	Set 1 - (6 projects, 360 activities)				Set 2 - (12 projects, 720 activities)				Set 3 - (24 projects, 1440 activities)			
		PPL				PPL				PPL		
	%-Best	Avg.	SD	Range	%-Best	Avg.	SD	Range	%-Best	Avg.	SD	Range
TPM	34.45%	0.063	0.092	[0.000–0.532]	3.62%	0.059	0.090	[0.000–0.492]	1.06%	0.048	0.088	[0.000–0.551]
APM	12.97%	0.706	0.167	[0.008–0.926]	4.65%	0.731	0.138	[0.010–0.915]	7.10%	0.767	0.117	[0.008–0.919]
# Instances			833				1463				2254	

CP1 due date:

Opt. criterion	Set 1 - (6 projects, 360 activities)				Set 2 - (12 projects, 720 activities)				Set 3 - (24 projects, 1440 activities)			
		PPL				PPL				PPL		
	%-Best	Avg.	SD	Range	%-Best	Avg.	SD	Range	%-Best	Avg.	SD	Range
APD	13.57%	0.706	0.168	[0.004–0.925]	4.72%	0.731	0.137	[0.010–0.918]	6.88%	0.767	0.116	[0.009–0.918]
ARG	9.00%	0.758	0.163	[0.007–0.937]	4.24%	0.774	0.139	[0.008–0.932]	6.43%	0.799	0.114	[0.008–0.917]
SPD	30.37%	0.683	0.179	[0.003–0.933]	20.44%	0.711	0.146	[0.010–0.921]	20.19%	0.752	0.119	[0.018–0.914]
MaxPD	14.41%	0.206	0.148	[0.001–0.724]	3.55%	0.261	0.145	[0.000–0.710]	2.57%	0.334	0.152	[0.000–0.766]
MaxRG	34.09%	0.584	0.222	[0.018–0.920]	37.73%	0.648	0.182	[0.022–0.918]	45.92%	0.712	0.139	[0.031–0.913]
# Instances			833				1463				2254	

RLB1 due date:

Opt. criterion	Set 1 - (6 projects, 360 activities)				Set 2 - (12 projects, 720 activities)				Set 3 - (24 projects, 1440 activities)			
		PPL				PPL				PPL		
	%-Best	Avg.	SD	Range	%-Best	Avg.	SD	Range	%-Best	Avg.	SD	Range
APD	57.02%	0.360	0.182	[0.005–0.855]	29.80%	0.341	0.171	[0.001–0.799]	21.92%	0.357	0.191	[0.001–0.819]
ARG	62.42%	0.363	0.181	[0.006–0.811]	35.41%	0.345	0.172	[0.003–0.851]	22.36%	0.357	0.193	[0.002–0.830]
SPD	51.62%	0.280	0.196	[0.001–0.829]	18.52%	0.279	0.184	[0.000–0.789]	12.24%	0.290	0.196	[0.000–0.777]
MaxPD	46.22%	0.063	0.093	[0.000–0.577]	14.70%	0.059	0.092	[0.000–0.519]	7.36%	0.046	0.086	[0.000–0.551]
MaxRG	62.91%	0.063	0.093	[0.000–0.577]	40.67%	0.059	0.092	[0.000–0.519]	31.19%	0.046	0.086	[0.000–0.551]
# Instances			833				1463				2254	

observed for the RLB1 due date compared to the CP1 due date. This could be attributed to our fixed-project resource-buffered scheduling approach, where only one project is subjected to an artificial resource restriction. Consequently, as the number of projects in the portfolio increases, the impact of our resource-buffered scheduling methods diminishes. Furthermore, the choice of optimisation criterion and due date significantly influences the structure of the best solution, as indicated by the PPL, and impacts the algorithm's performance. The best schedules for the TPM objectives lean towards those with low project prioritisation behaviour (low PPL), suggesting that progressing projects at a similar speed is more effective. Conversely, for objectives related to the CP1 due date (excluding the MaxPD criterion), the opposite trend is observed. For the RLB1 due date, the APD, ARG, and SPD objectives generate best schedules with an intermediate prioritisation strategy, where our algorithm achieves significant performance. Prioritising certain projects over others is no longer the unique optimal strategy because it leads to early completion of prioritised projects, resulting in an opportunity cost for resource utilisation in progressing other late projects.

Impact of network indicators: The composition of the portfolio, including the combination of serial and parallel projects, has an impact on both the relative performance of our algorithm and the structure of the best solutions generated. The MPLIB1 dataset instances are divided into seven groups based on the SP-indicators of their projects, focusing on the composition of parallel and serial projects. A first group of test instances in the MPLIB1 dataset is composed of only projects with a more parallel network structure (SP $<$ 0.35), a second group consists of only projects with an SP between 0.35 and 0.65 and a third group is composed of more serial projects (SP $\geq$ 0.65). A fourth group is composed of an equal number of projects with low, medium and high SP's within the aforementioned ranges resulting in a large deviation of the SP-indicators. While three additional groups with different compositions exist, they are excluded for concise analysis. Table 14 provides an overview of the algorithm's relative performance in terms of the number of best solutions generated and the average PPL (Avg.) of the best schedules for these four groups as well as their standard deviations (SD).

Several observations can be made from the analysis. Firstly, the composition of network structures within the portfolio significantly affects the relative performance of the genetic algorithm with resource buffers, with the direction of the effect dependent on the objective and due date. For the TPM objective, our algorithm performs best when the projects are predominantly serial or when there is a considerable number of highly serial projects. Conversely, for the APD objective with the CP1 due date, our algorithm performs relatively better for more parallel projects, while its performance worsens when the network structures within the portfolio are highly diverse. However, for the RLB1 due date, better performance is observed when the portfolio consists of mainly serial projects, while the algorithm's performance remains inferior when the projects exhibit high diversity in network structures. The PPL of the best schedules also varies based on the portfolio composition, with the highest PPLs observed for instances with highly diverse network structures. This indicates that a project prioritisation strategy is more effective when projects differ significantly in their network structures. Similar observations can be made on the remaining objectives and are left to the interested reader to review in Table 14.

Impact of resource indicators: The relative performance of our algorithm improves when the workloads among the projects are more differentiated. This is shown in Table 15 where the instances are divided over three groups determined by the deviation of the total work contents of the most critical resource ($TWK_{\bar{k}}$) among the projects in the portfolio. The 1518 portfolios with the lowest deviation of $TWK_{\bar{k}}$ are assigned to the left-most group in the table while the 1518 instances with the highest deviation are assigned to the right-most group. The remaining 1520 instances are assigned to the middle group. In addition to the effect on the relative performance, it is observed that a more heterogeneous composition of the portfolio in terms of the projects' relative workloads increases the PPL of the best schedule.

Table 14
Impact of average and deviation of SP-indicators among the projects on relative algorithmic performance and PPL of the schedules.

Due date	Objective	Low dev. of SPs									High dev. of SPs		
		Low avg. SP			Medium avg. SP			High avg. SP					
		SPs: [0–0.35[			SPs: *[0.35–0.65[*			*SPs: [0.65–1]*			*SPs: [0–1]*		
			PPL			PPL			PPL			PPL	
		#Best	Avg.	SD	#Best	Avg.	SD	#Best	Avg.	SD	#Best	Avg.	SD
	TPM	16	0.045	0.057	51	0.042	0.076	85	0.035	0.074	59	0.071	0.101
	APM	139	0.757	0.125	47	0.721	0.144	41	0.703	0.170	24	0.763	0.070
CP1	APD	131	0.757	0.125	45	0.722	0.144	30	0.702	0.170	36	0.762	0.102
CP1	ARG	119	0.786	0.121	40	0.762	0.148	28	0.713	0.170	18	0.815	0.095
CP1	SPD	246	0.742	0.131	176	0.693	0.154	165	0.662	0.178	99	0.756	0.101
CP1	MaxPD	36	0.320	0.113	29	0.219	0.141	38	0.159	0.134	24	0.355	0.132
CP1	MaxRG	350	0.675	0.138	318	0.574	0.153	271	0.486	0.162	249	0.773	0.105
RLB1	APD	164	0.283	0.160	285	0.271	0.173	333	0.287	0.174	139	0.415	0.168
RLB1	ARG	100	0.283	0.160	306	0.273	0.176	360	0.289	0.175	189	0.417	0.167
RLB1	SPD	95	0.250	0.168	196	0.204	0.171	237	0.203	0.181	109	0.341	0.185
RLB1	MaxPD	54	0.044	0.057	109	0.042	0.076	146	0.033	0.073	122	0.073	0.105
RLB1	MaxRG	377	0.044	0.057	180	0.042	0.076	177	0.033	0.073	294	0.073	0.105
# instances		650			650			650			650		

Table 15
Impact of the deviation of the projects' relative consumption of the most critical resource, $TWK_{\bar{k}}$.

Due date	Objective	Low dev. $TWK_{\bar{k}}$			Medium dev. $TWK_{\bar{k}}$			High dev. $TWK_{\bar{k}}$		
			PPL			PPL			PPL	
		#Best	Avg.	SD	#Best	Avg.	SD	#Best	Avg.	SD
	TPM	119	0.027	0.059	100	0.042	0.076	147	0.094	0.110
	APM	54	0.739	0.132	82	0.733	0.140	200	0.760	0.124
CP1	APD	45	0.739	0.132	80	0.733	0.140	212	0.760	0.123
CP1	ARG	65	0.727	0.132	81	0.779	0.140	137	0.800	0.118
CP1	SPD	267	0.723	0.143	286	0.717	0.146	455	0.739	0.126
CP1	MaxPD	70	0.261	0.152	69	0.286	0.152	92	0.316	0.136
CP1	MaxRG	615	0.624	0.172	579	0.665	0.169	678	0.715	0.146
RLB1	APD	507	0.337	0.173	416	0.348	0.182	487	0.372	0.191
RLB1	ARG	527	0.340	0.178	466	0.350	0.181	553	0.373	0.191
RLB1	SPD	357	0.254	0.177	300	0.277	0.188	324	0.324	0.203
RLB1	MaxPD	258	0.026	0.058	221	0.040	0.074	290	0.095	0.112
RLB1	MaxRG	450	0.026	0.058	494	0.040	0.074	881	0.095	0.112
# instances		1518			1520			1518		

6. Conclusion

In this study, we propose a new genetic algorithm with resource buffers for the RCMPSP. Our proposed algorithm demonstrates competitiveness with and, for 20.04% of the test instances, superior performance compared to ten other metaheuristic approaches. By incorporating resource-buffered scheduling strategies, our algorithm excels in scenarios where project due dates are more lenient and evenly distributed, allowing for potential project earliness. Moreover, our results indicate that the effectiveness of our algorithm depends on the specific optimisation criterion and on the size of the portfolio, with a superior performance on instances with a smaller number of projects. Furthermore, our findings highlight the importance of project prioritisation in heterogeneous project portfolios, as evidenced by the analysis of the Project Prioritisation Level (PPL) metric developed within this study. This finding aligns with previous observations by Bredael and Vanhoucke (2023). To further enhance the algorithm's performance, future research could focus on incorporating resource-buffering strategies for multiple projects.

The benchmark data for the 11 metaheuristics discussed in this study is available online at www.projectmanagement.ugent.be/research/project_scheduling/RCMPSP. This data is intended to encourage the development of more effective exact or heuristic solution methods for the RCMPSP.

Acknowledgements

We acknowledge the support provided by the Bijzonder Onderzoeksfonds (BOF), Belgium with contract number 24J046-17 and the Research Foundation – Flanders (FWO), Belgium for the project, under contract number 3G012616. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputing Center), funded by Ghent University, Belgium, FWO and the Flemish Government department EWI.

References

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, *5*, 11–24.

Bredael, D., & Vanhoucke, M. (2023). Multi-project scheduling: A benchmark analysis of metaheuristic algorithms on various optimisation criteria and due dates. *European Journal of Operational Research*, *308*, 54–75.

Browning, T. R., & Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, *126*(2), 212–228.

Chen, P., & Shahandashti, S. (2009). Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Automation in Construction*, *18*, 434–443.

Debels, D., De Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, *169*, 638–653.

Gonçalves, J., Mendes, J., & Resende, M. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, *189*, 1171–1190.

Homberger, J. (2007). A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, *14*, 565–589.

Kumanan, S., Jose, G. J., & Raja, K. (2006). Multi-project scheduling using an heuristic and a genetic algorithm. *International Journal of Advanced Manufacturing Technology*, *31*, 360–366.

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

Mittal, M., & Kanda, A. (2009). Two-phase heuristics for scheduling of multiple projects. *International Journal of Operational Research*, *4*, 159–177.

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*, 1879–1896.

Sonmez, R., & Uysal, F. (2015). Backward-forward hybrid genetic algorithm for resource-constrained multiproject scheduling problem. *Journal of Computing in Civil Engineering*, *29*, 159–177.

Van Eynde, R., & Vanhoucke, M. (2020). Resource-constrained multi-project scheduling: Benchmark datasets and decoupled scheduling. *Journal of Scheduling*, *23*, 301–325.

Vázquez, E., Calvo, M., & Ordóñez, P. (2015). Learning process on priority rules to solve the RCMPSP. *Journal of Intelligent Manufacturing*, *26*, 123–138.

Wauters, T., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2015). A learning-based optimisation approach to multi-project scheduling. *Journal of Scheduling*, *18*(1), 61–74.