Annals of Operations Research
https://doi.org/10.1007/s10479-022-05109-0

ORIGINAL RESEARCH

MIP-based solution approaches for multi-site resource-constrained project scheduling

Tamara Bigler[1] · **Mario Gnägi**[1] · **Norbert Trautmann**[1]

Accepted: 22 November 2022

Abstract
The execution of a project is often distributed among multiple sites. The planning of such a project includes selecting a specific site for the execution of each of the project's activities and allocating the available resource units to the execution of these activities over time. While some resource units are available at a certain site only, others can be moved across sites. Given the spatial distance between sites, transportation times arise if a resource unit must be transported from one site to another or if the output of an activity must be transported to another site. This planning problem has been introduced in recent literature as the multi-site resource-constrained project scheduling problem. We present a continuous-time model and devise a matheuristic for this planning problem. The continuous-time model uses, among others, binary variables to impose a sequence between activities assigned to the same resource units. In the matheuristic, the binary restrictions on these variables are initially relaxed and iteratively restored for the subset of activities scheduled in the current iteration. We compare the performance of the continuous-time model and the matheuristic to the performance of a discrete-time model and several metaheuristics from the literature using two sets of test instances from the literature. Both the continuous-time model and the matheuristic derive on average superior solutions in shorter average running times than the reference approaches.

Keywords Resource-constrained project scheduling · Multiple sites · Mathematical programming · Matheuristic

1 Introduction

Recently, Laurent et al. (2017) introduced the multi-site resource-constrained project scheduling problem, or short multi-site RCPSP, which allows to consider the execution of a project's activities at several sites and the sharing of resources among these sites. This planning problem is motivated by a real-world application from health care management, where medical examinations are conducted in different hospitals and the required medical staff is shared between these hospitals. According to Laurent et al. (2017), this pooling of medical examinations and staff increases the hospitals' total productivity even though additional transportation

✉ Norbert Trautmann
norbert.trautmann@unibe.ch

[1] Department of Business Administration, University of Bern, Bern, Switzerland

Published online: 01 December 2022

times occur due to the spatial distance between the hospitals. Another sample application are R&D projects whose research activities may be carried out in several laboratories that share some of their resources. In this application, mobile resource units, e.g., research staff or equipment, or the output of some research activities, e.g., laboratory samples or prototypes, must be transported from one laboratory to another, resulting in transportation times that must be considered. The multi-site RCPSP is an extension of the widely studied single-site RCPSP, which represents an NP-hard optimization problem. Consequently, the multi-site RCPSP is also NP-hard.

The multi-site RCPSP—similar to the single-site RCPSP—consists of allocating individual units of some renewable resources over time to the execution of the activities of a project such that the project duration is minimized while taking into account a prescribed set of completion-start precedence relationships between pairs of activities. In addition, in the single-site RCPSP, it is assumed that all activities are executed at the same site and, consequently, that all resources are located at this site. In the multi-site RCPSP, in contrast, a specific site for the execution of each activity must be selected. Moreover, the spatial distance between the sites gives rise to two different types of transportation times that must be considered in the project schedule. First, during the transport of a resource unit between the sites, the resource unit cannot be allocated to the execution of an activity. Second, if two activities that are interrelated by a precedence relationship are executed at different sites, then a minimum time lag between the completion of the first activity and the start of the second activity must be taken into account, which corresponds to the time required to transport the first activity's output between the respective sites. Hereafter, we refer to the first type of transport as resource transport and to the second type of transport as output transport.

To the best of our knowledge, the only solution approaches to the multi-site RCPSP that are documented in the literature are the formulation as a binary linear program and the four different metaheuristics proposed by Laurent et al. (2017). The linear programming formulation belongs to the class of discrete-time models; i.e., the planning horizon is divided into a set of equally long periods, and it is assumed that an activity can be started at the beginning of such a period only. In an experimental analysis performed by Laurent et al. (2017) with CPLEX using a set of 192 instances, where the number of activities was varied between 5 and 30, it turned out that within a prescribed time limit of 3600 s of running time, none of the instances comprising 30 activities could be solved to optimality. The four metaheuristics are based on a representation of feasible solutions by an activity list and a site list and apply the search strategies local search, simulated annealing and iterated local search. In another experimental analysis performed by Laurent et al. (2017) with instances comprising 30–120 activities, it turned out that the iterated local search and the simulated annealing metaheuristics performed best.

The contribution of this paper is twofold: first, we provide a continuous-time model for the multi-site RCPSP that is applicable to challenging instances comprising up to 30 activities such that it can be used to evaluate the performance of heuristic methods; and second, we devise a novel matheuristic for the multi-site RCPSP that follows an iterative, relax-optimize-and-fix strategy and applies a relaxation of the novel continuous-time model in each iteration. More specifically, the novel model, which we elaborate starting from the single-site RCPSP formulation presented in Rihm and Trautmann (2017), is formulated as a mixed-binary linear programming (MBLP) model and employs continuous start-time variables and binary site-selection, resource-assignment and sequencing variables. In an experimental performance analysis, we apply the novel continuous-time model and the discrete-time model proposed by Laurent et al. (2017) to a set of 960 instances comprising 30 activities and 2 or 3 sites that were generated by Laurent et al. (2017). Feasible solutions are devised for all instances with

the novel continuous-time model, and a large number of these instances can even be solved to optimality. Moreover, for a considerable number of instances, using the novel continuous-time model yields a feasible solution that has a better objective function value than the best solution devised by the discrete-time model presented in Laurent et al. (2017) in a shorter average running time. Based on this novel continuous-time model, we propose a matheuristic for larger instances. In this matheuristic, subsets of the activities are iteratively scheduled using a relaxation of the model. All activities of the instance are considered in this relaxation. During the solution process, however, several activities may overlap with each other, i.e., use the same resource unit at the same time. In each iteration, the activities in the respective subset are sequenced among themselves and among the already-scheduled activities, until after the last iteration, no overlaps remain. The matheuristic obtains good feasible solutions for instances comprising 30 activities with a lower average gap to the critical-path-based lower bound than the continuous-time model. For the instances comprising 60 activities, it outperforms the metaheuristics of Laurent et al. (2017) on all considered performance metrics.

The remainder of this paper is organized as follows. In Sect. 2, we illustrate the multi-site RCPSP by an example and provide an overview of related project scheduling problems. In Sects. 3 and 4, we present the novel continuous-time model and the matheuristic. In Sect. 5, we report our computational results. In Sect. 6, we provide some conclusions and provide an outlook for future research.

2 Planning problem

In this section, we introduce some basic notation (Sect. 2.1), illustrate the planning problem by means of an example project (Sect. 2.2), and provide an overview of related project scheduling problems with transportation times discussed in the literature (Sect. 2.3).

2.1 Basic notation

We consider a project that consists of n real activities. These real activities form the set V. In addition, we introduce two fictitious activities, 0 and $n+1$, representing the start and end of the project, respectively; both of these activities are assumed to have a duration of zero and to require no resources. Each real activity $i \in V$ has a prescribed duration p_i and must not be pre-empted after it has been started. Furthermore, given is a set E containing the pairs of activities between which there is a completion-start precedence relationship. For the execution of the activities, a set R of different renewable resource types and a set L of alternative sites are given. For each resource type $k \in R$, we denote the available number of units as R_k and the required number of units for the execution of activity $i \in V$ as $r_{ik} \leq R_k$. Moreover, each available unit $u \in \{1, \ldots, R_k\}$ of resource type $k \in R$ is either mobile (i.e., $M_{ku} = 1$) or nonmobile (i.e., $M_{ku} = 0$). All mobile resource units are assumed to be initially located at the site at which their first assigned real activity is executed, i.e., the mobile resource units do not have to be moved before the start of the first real activity assigned to them. Moreover, the mobile resource units are assumed to remain at the site at which the last real activity assigned to them is executed, i.e., the mobile resource units do not have to be moved after the completion of the last real activity assigned to them. For each nonmobile resource unit, there is a site loc_{ku} at which the resource unit is permanently located. Finally, $\delta_{ll'}$ denotes the transportation time between sites l and $l' \in L \times L$.

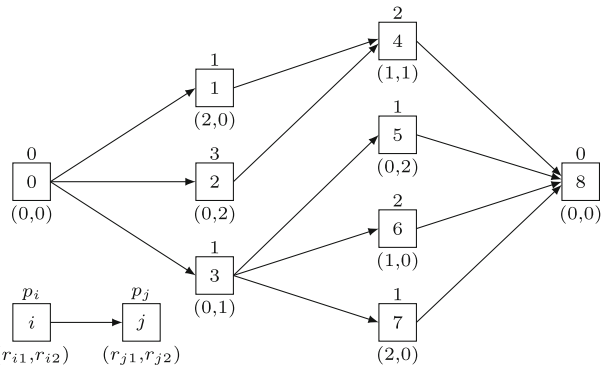

Fig. 1 Activity-on-node network for the example project

2.2 Illustration of the planning problem

We illustrate the planning problem by means of an example project that comprises seven real activities, i.e., $V = \{1, \ldots, 7\}$. For the execution of these activities, there are two sites, A and B, i.e., $L = \{\text{A}, \text{B}\}$, between which we assume a transportation time of one time unit in both directions, i.e., $\delta_{AB} = \delta_{BA} = 1$. Moreover, two different resource types are required to execute the activities, i.e., $R = \{1, 2\}$, with two resource units of both resource types available, i.e., $R_1 = 2$ and $R_2 = 2$. Both resource units of resource type $k = 1$ are nonmobile, i.e., $M_{11} = M_{12} = 0$, and permanently located at site A, i.e., $loc_{11} = loc_{12} = \text{A}$. The first unit of resource type $k = 2$ is also nonmobile, i.e., $M_{21} = 0$, and permanently located at site B, i.e., $loc_{21} = \text{B}$, while the second unit is mobile, i.e., $M_{22} = 1$. Figure 1 shows an activity-on-node network, which depicts the activities as nodes and the precedence relationships as directed edges between the nodes, for the example project. The durations and resource requirements of the activities are indicated above and below the nodes of the network, respectively.

An optimal solution for the example project with a minimal makespan of eight time units is shown in Fig. 2. Each line represents a resource unit u of a resource type k, and the rectangles represent the activities. Each real activity is assigned to at least one resource unit and exactly one site. Activities 1, 4, 6 and 7 are executed at site A, while activities 2, 3 and 5 are executed at site B. Resource transport, e.g., between activities 5 and 4, is indicated by a rectangle with a cross. These two activities are both assigned to the resource unit $u = 2$ of resource type $k = 2$ and take place at different sites. Thus, the commonly used resource unit must be moved from site B to site A before activity 4 can start. Output transports, e.g., between activities 3 and 7, are indicated with arrows. These two activities are precedence related and take place at different sites. Thus, the output of activity 3 must be moved from site B to site A before activity 7 can start.

2.3 Related project scheduling problems with transportation times

In addition to the work of Laurent et al. (2017), which introduces and addresses the multi-site RCPSP for the first time, there are other papers in the literature that discuss related project scheduling problems. In the following, we outline these related project planning problems and point out the differences to the problem considered in this paper.

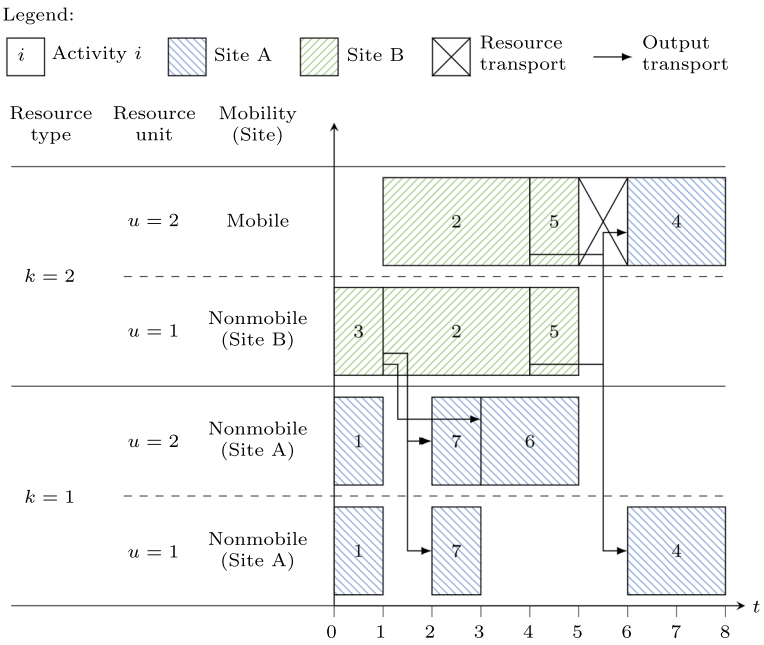

Fig. 2 An optimal solution for the example project

A well-known scheduling problem that has similar characteristics as the multi-site RCPSP is the multi-mode RCPSP (cf., e.g., Gnägi et al., 2019). In the multi-mode RCPSP, the project activities can be executed in alternative modes that differ in terms of the activities' durations and resource requirements. Selecting an execution mode for an activity can be interpreted as assigning the activity to a specific site. The selection of an execution mode for an activity, however, affects only the duration of this specific activity individually. Therefore, output transport cannot be addressed since such transport always includes two activities. Resource transport cannot be addressed, either, since the movement of mobile resource units between different sites cannot be modeled based on mode selection. Several extensions of the multi-mode RCPSP that explicitly consider transportation times (also called time lags, setup times or transfer times) are discussed in the literature. In the multi-mode RCPSP with mode-dependent time lags, which has been discussed by Sabzehparvar and Seyed-Hosseini (2008), the length of the time lags depends on the selected modes of two activities. Therefore, when the mode selection is interpreted as a site selection, output transport, as in the multi-site RCPSP, may be addressed. However, the sharing of mobile resource units among different sites cannot be addressed. The multi-mode RCPSP with schedule-dependent setup times, considered by Mika et al. (2008), explicitly involves the assignment of activities to locations and the movement of semifinished products between the locations of precendence-related activities. The required time for this movement is interpreted as a setup time for the subsequent activity. However, also in this planning problem, the locations of the resource units are assumed to be fixed, and thus, no sharing of mobile resource units among different locations is addressed. Another extension, namely, the multi-mode RCPSP with sequence-dependent transfer times, is considered by Kadri and Boctor (2014). Transfer times occur when resource units are moved between several locations, and the duration of the transfers depends on the locations

at which the involved activities are executed. Unlike in the multi-site RCPSP, however, in this case, the locations for the execution of the activities are assumed to be given.

A less related stream of literature, initiated by Krüger and Scholl (2009, 2010), deals with the RCPSP with transfer times in the context of single- and multi-project scheduling. In these problem settings, transfer times occur when resource units are transferred from one activity to another, while the duration of the transfer depends on the involved activities as well as the resource type. These transfer times may occur within the same project but also among multiple projects, but the belonging of the activities to the projects is considered as given. Therefore, the multi-project context cannot be exploited to model the selection of sites for the execution of the activities. The identical planning problem in the context of single-project scheduling is considered by Poppenborg and Knust (2016), Kadri and Boctor (2018) and Liu et al. (2021). Liu et al. (2021) focus on the special case where only resources with one available unit are considered.

In sum, some of the abovementioned papers address isolated parts of the problem discussed in this paper. Most of them do not explicitly involve the selection of sites for the execution of the activities and the transport of semifinished products between the sites of precedence-related activities, and none of them accounts for the sharing of mobile resource units between different sites.

3 Continuous-time MBLP model

In this section, we present the continuous-time MBLP model for the multi-site RCPSP; a preliminary version of this model is presented in Gnägi and Trautmann (2019). In Sect. 3.1, we illustrate the different types of variables by means of the example project introduced in Sect. 2. In Sect. 3.2, we present the formulation of the objective and the constraints.

3.1 Types of variables

The model employs the four types of variables listed in Table 1. The continuous start-time variables S_i $(i \in V \cup \{0, n+1\})$ indicate when to start the activities. The project is assumed to start at time point zero, i.e., $S_0 := 0$, and consequently, the starting time of activity $n+1$ represents the project duration. The binary site-selection variables s_{il} $(i \in V;\ l \in L)$ indicate at which site the activities are executed. The binary resource-assignment variables r^u_{ik} $(i \in V;\ k \in R;\ u \in \{1, \ldots, R_k\} : r_{ik} > 0)$ indicate to which resource units the activities are assigned. Finally, the binary sequencing variables y_{ij} $(i, j \in V \times V : i \neq j,\ (i, j) \notin TE)$ indicate the sequence in which two activities i and j must be executed. Note that the sequencing variables are only defined for all pairs of activities (i, j) with $i \neq j$ which can be executed simultaneously with respect to the precedence relationships, i.e., $(i, j) \notin TE$. If two of these activities, i.e., activities i and j, use at least one common resource unit of the same resource type, then a sequencing between the two activities must be ensured, i.e., either y_{ij} or y_{ji} will take the value one. Figure 3 illustrates the four types of variables by means of the example project introduced in Sect. 2. For the binary variables, only those variables with a positive value in the given optimal solution are displayed.

Table 1 Variable descriptions

Variable		Description
*	S_i	Starting time of activity i
*	s_{il}	$\begin{cases} = 1, & \text{if activity } i \text{ is executed at site } l \\ = 0, & \text{otherwise} \end{cases}$
*	r_{ik}^{u}	$\begin{cases} = 1, & \text{if activity } i \text{ is assigned to unit } u \text{ of resource type } k \\ = 0, & \text{otherwise} \end{cases}$
*	y_{ij}	$\begin{cases} = 1, & \text{if activity } i \text{ must be completed before the start of activity } j \\ = 0, & \text{otherwise} \end{cases}$

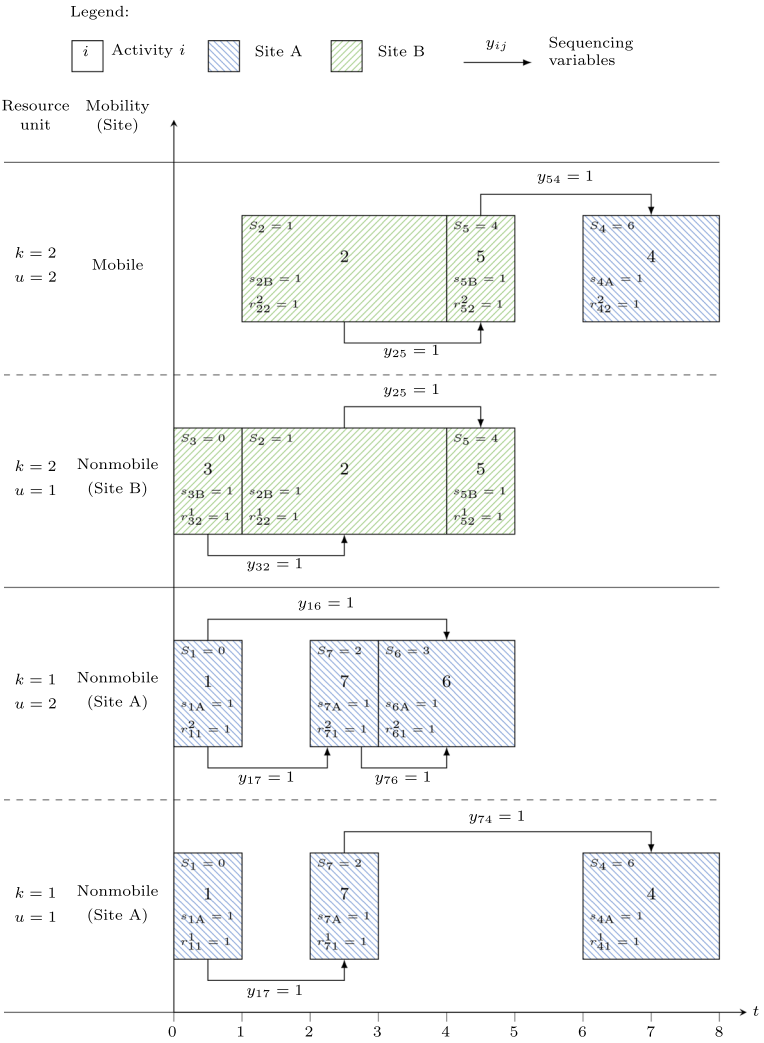

Fig. 3 Illustration of the types of variables for the example project

3.2 Formulation of objective and constraints

The objective is to minimize the duration of the project:

$$\text{Min. } S_{n+1}$$

Constraints (1) account for the prescribed precedence relationships among the real activities. Moreover, a transportation time of $\delta_{ll'}$ is triggered if two precedence-related real activities i and j are performed at two different sites l and l'. This transportation time is required to move the output of the first activity between the respective sites.

$$S_i + p_i + (s_{il} + s_{jl'} - 1)\delta_{ll'} \leq S_j \quad (i, j \in V \times V : \ (i, j) \in E; \ l, l' \in L \times L) \tag{1}$$

Constraints (2) enforce that if two real activities i and j are both assigned to the same unit u of a resource type k, then either activity i must be completed before the start of activity j or vice versa, i.e., either $y_{ij} = 1$ or $y_{ji} = 1$.

$$r_{ik}^u + r_{jk}^u \leq y_{ij} + y_{ji} + 1 \quad (i, j \in V \times V; \ k \in R; \ u \in \{1, \ldots, R_k\} : \\ i < j, \ (i, j) \notin TE, \ r_{ik} > 0, \ r_{jk} > 0) \tag{2}$$

Constraints (3) link the start-time variables and the sequencing variables, where $\delta^{max} := \max_{l,l' \in L \times L}\{\delta_{ll'}\}$ corresponds to the longest transportation time between all pairs of sites. A transportation time of $\delta_{ll'}$ is triggered if two activities i and j are both assigned to at least one common resource unit and are performed at two different sites l and l'. This transportation time is required to move the commonly used resource units between the respective sites.

$$S_i + p_i + (s_{il} + s_{jl'} - 1)\delta_{ll'} \leq S_j + \left(\sum_{h \in V} p_h + n\delta^{max}\right)(1 - y_{ij}) \\ (i, j \in V \times V : \ i \neq j, \ (i, j) \notin TE; \ l, l' \in L \times L) \tag{3}$$

Constraints (4) ensure that all real activities are completed before the project ends. Constraints (4) are required in addition to constraints (1) because the latter only account for the precedence relationships between all real activities, which do not include activity $n + 1$.

$$S_i + p_i \leq S_{n+1} \quad (i \in V \cup \{0\}) \tag{4}$$

Constraints (5) guarantee that the number of resource units allocated to each activity is equal to the number of required resource units of the respective activity.

$$\sum_{u=1}^{R_k} r_{ik}^u = r_{ik} \quad (i \in V; \ k \in R : \ r_{ik} > 0) \tag{5}$$

Constraints (6) enforce that each activity i that is assigned to a nonmobile unit u of any resource type k must be performed at the site at which the respective unit is permanently located.

$$r_{ik}^u \leq s_{i,loc_{ku}} \quad (i \in V; \ k \in R; \ u \in \{1, \ldots, R_k\} : \ M_{ku} = 0, \ r_{ik} > 0) \tag{6}$$

Constraints (7) ensure that each activity is performed at exactly one site.

$$\sum_{l \in L} s_{il} = 1 \quad (i \in V) \tag{7}$$

Finally, (8) and (9) are redundant constraints, where the set F^2 comprises all pairs of activities i and j that cannot be performed in parallel since together they require more resource units than are available, i.e., $r_{ik} + r_{jk} > R_k$ for some resource type k. These redundant constraints turned out to substantially speed up the solution process of the mathematical programming solver used during our computational experiment.

$$y_{ij} + y_{ji} = 1 \quad ((i, j) \in F^2) \tag{8}$$

$$y_{ij} + y_{ji} \leq 1 \quad (i, j \in V \times V : i < j,\ (i, j) \notin TE) \tag{9}$$

The novel model, which we refer to as the model (CT) hereafter, reads as follows:

$$\text{(CT)} \begin{cases} \text{Min.} & S_{n+1} \\ \text{s.t.} & (1) - (9) \\ & S_i \in \mathbb{R}_{\geq 0} \quad (i \in V \cup \{0, n+1\}) \\ & s_{il} \in \{0, 1\} \quad (i \in V;\ l \in L) \\ & y_{ij} \in \{0, 1\} \quad (i, j \in V \times V : i \neq j,\ (i, j) \notin TE) \\ & r^u_{ik} \in \{0, 1\} \quad (i \in V;\ k \in R;\ u \in \{1, \ldots, R_k\} : r_{ik} > 0) \end{cases}$$

Alternatively, the start-time variables can also be defined as nonnegative integer variables, i.e., $S_i \in \mathbb{Z}_{\geq 0}$ $(i \in V \cup \{0, n+1\})$, as long as the durations of the activities and the transportation times are all integers. In Sect. 5, we report some results of our computational experiment that indicate that this accelerates the solution process of the mathematical programming solver used.

The continuous-time model presented in this section turns out to be applicable to challenging instances comprising up to 30 activities (see Sect. 5). For instances comprising considerably more than 30 activities, a heuristic approach such as the matheuristic presented in the following section, which is able to derive feasible solutions within a reasonable running time, may be more appropriate.

4 Relax-optimize-and-fix matheuristic

In this section, we present the relax-optimize-and-fix matheuristic for the multi-site RCPSP. In Sect. 4.1, we give an overview of the novel matheuristic. In Sect. 4.2, we describe the different steps of our matheuristic in detail. In Sect. 4.3, we illustrate the matheuristic with the example project from Sect. 2.

4.1 Overview

Matheuristics are recent approaches for combinatorial optimization problems that combine advantages of mathematical models and heuristic techniques (cf. Maniezzo et al., 2021) or metaheuristic techniques (cf. Boschetti et al., 2009). As to the four essential characteristics of accuracy, speed, simplicity and flexibility of good heuristics (cf. Cordeau et al., 2002), matheuristics especially excel regarding their simplicity and flexibility. Matheuristics are simple to implement since nowadays relatively easy-to-use modeling languages and interfaces to various programming languages are available. Moreover, matheuristics are flexible because additional constraints or alternative objective functions can easily be incorporated using mathematical modeling.

As mentioned above, our matheuristic is specifically designed for devising good feasible solutions within reasonable running time to instances of the multi-site RCPSP whose size

does not permit an immediate application of the mathematical model presented in Sect. 3. To this end, we combine the mathematical model with a heuristic technique. Among the different types of matheuristics presented in Della Croce et al. (2013), our approach is related to the group of the continuous relaxation based matheuristics, which according to Della Croce et al. (2013) cleverly exploit information extracted from the LP relaxation of a mathematical model. The basic idea of our matheuristic is to iteratively schedule subsets of the activities by solving an appropriate relaxation of the mathematical model, which is obtained by relaxing the binary restrictions for subsets of the sequencing variables.

More specifically, our matheuristic runs as follows. In each iteration, a subset of the activities is scheduled while taking into account all activities of the instance. We use an appropriate variant of the model (CT) presented in Sect. 3 to schedule these activities. The matheuristic follows a relax-optimize-and-fix strategy; i.e., initially, the binary restrictions on the sequencing variables in the variant of the model (CT) are relaxed and then iteratively restored for a subset of the activities scheduled in the next iteration. After each iteration, some activities of the subset are considered as scheduled, and hence, some variables are fixed for these activities. The other activities remain in the subset and can be rescheduled along with newly selected activities in the next iteration. An important advantage of using the model (CT) in the matheuristic is that in the model (CT), the sequence of the activities and the start times of the activities are captured using different variable types. Hence, we can fix the sequence between the scheduled activities and still keep the flexibility to move these activities along the timeline.

In our matheuristic, the trade-off between solution quality and running time can be controlled by two parameters. The first parameter indicates how many activities are scheduled in each iteration, i.e., it influences the number of variables for which the binary restrictions are restored in each iteration. In general, the solution quality improves with an increasing number of activities scheduled in each iteration, but the running time to solve the variant of the model (CT) in each iteration also increases. Therefore, this parameter trades off the solution quality against the running time in a single iteration. The second parameter indicates how many activities are considered as scheduled after each iteration and consequently how many activities are allowed to be rescheduled and how many new activities are selected. In general, the solution quality improves with a decreasing number of activities that are considered as scheduled after each iteration, but the overall running time also increases because an increasing number of iterations has to be conducted. Hence, this parameter allows us to control the number of iterations that are required to derive a feasible solution to an instance, and it trades off the solution quality against the running time over the entire process of the matheuristic.

4.2 Different steps of the matheuristic

In the matheuristic, set V is divided into three sets that contain the activities V^N that have not yet been scheduled, the activities V^C that are scheduled in the current iteration and the activities V^S that have been scheduled and for which several variables are fixed. Figure 4 shows an overview of the different steps of the matheuristic. In the following, we describe these steps in more detail.

- In step ①, the set V^N is initialized to contain all real activities, and the sets V^C and V^S are initialized as empty sets.
- In step ②, $c > 0$ activities are selected from V^N. To select these activities, a priority rule is applied which may be of the same type such as those used in, e.g., the serial

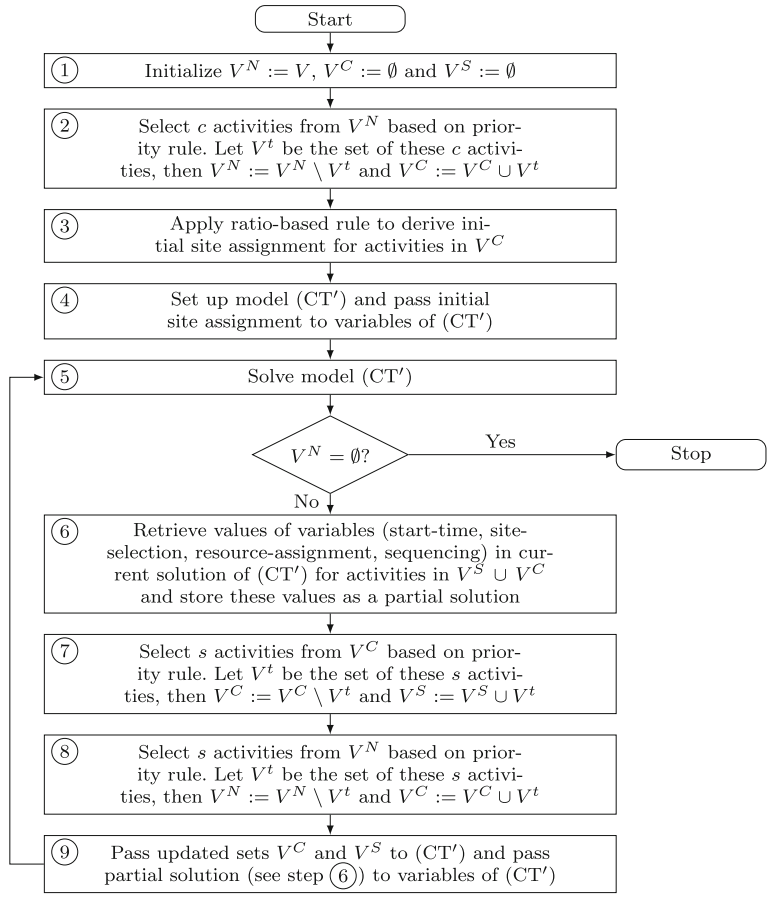

Fig. 4 Overview of the different steps of the matheuristic

schedule generation scheme. Please note that the selection of an appropriate priority rule considerably influences the performance of the matheuristic. In Sect. 5.4, we discuss how to define a suitable value for c.

– In step ③, we derive an initial site assignment for the activities in V^C. The idea here is to select for each activity i a resource type k, of which many units are required to execute activity i and of which only a few mobile units are available; therefore, it might be promising to schedule activity i at a site where many nonmobile resource units of resource type k are available. Thus, we propose a rule that is based on a ratio between two key figures that can be computed for each activity i and each resource type k. For each activity, the resource type k^* with the largest ratio $\frac{r_{ik^*}}{1+\sum_{u=1}^{R_{k^*}} M_{k^*u}}$ is selected. Next, we determine at which site the highest number of nonmobile resource units of the selected resource type k^* are located. This site is selected as the initial site assignment of activity i. If resource type k^* has no nonmobile resource units, the site at which the highest number of nonmobile resource units are located over all resource types is selected as the initial site assignment of activity i. In the case of any ties, the respective resource or site index is used as tie-breaker. Initial testing has indicated that providing an initial site assignment by

using this simple rule improves the solution quality of the matheuristic without increasing the running time. It is further important to note that the initial site assignment will be passed to the corresponding variables of a variant of the model (CT) introduced in step ④ as *initial* values, but the values of these variables may still change during the solution process.

– In step ④, a variant of the model (CT) referred to as model (CT′) is set up. The model (CT′) differs from the model (CT) in regard to several constraints and the domains of some of the variables. The binary restrictions on the sequencing variables y_{ij} are initially relaxed for all activities and restored only if both activities i and j are in $V^C \cup V^S$. In the model (CT), the redundant constraints (8) and (9) are added for all activities. However, it turns out to substantially speed up the solution process in the model (CT′) to add only the constraints in which the corresponding sequencing variables are defined as binary in each iteration instead of also adding the constraints in which the corresponding binary restrictions of the sequencing variables are relaxed. Hence, the redundant constraints (10) are set up for the activities (i, j) in F^2 only if these activities are in $V^C \cup V^S$:

$$y_{ij} + y_{ji} = 1 \quad ((i, j) \in F^2 : i, j \in V^C \cup V^S) \tag{10}$$

For the same reason, the second type of redundant constraints (11) are added only for the activities in $V^C \cup V^S$:

$$y_{ij} + y_{ji} \leq 1 \quad (i, j \in V^C \cup V^S : i < j, (i, j) \notin TE) \tag{11}$$

Starting from the second iteration, some of the variables associated with the activities in V^S are fixed. Here, s^*_{il}, r^{*u}_{ik} and y^*_{ij} denote the values that the site-selection, resource-assignment and sequencing variables took in the solution to the model (CT′) in the previous iteration, respectively. Constraints (12) fix the site-selection variables of the activities in V^S to the values that the corresponding variables took in the last iteration.

$$s_{il} = s^*_{il} \quad (i \in V^S; \ l \in L) \tag{12}$$

Constraints (13) fix the resource-assignment variables of the activities in V^S to the values that these variables took in the last iteration.

$$r^u_{ik} = r^{*u}_{ik} \quad (i \in V^S; \ k \in R; \ u = 1, \ldots, R_k : \ r_{ik} > 0) \tag{13}$$

Finally, constraints (14) fix the sequencing variables between activities in V^S to the values that the corresponding variables took in the last iteration.

$$y_{ij} = y^*_{ij} \quad (i, j \in V^S) \tag{14}$$

The complete model (CT′) reads as follows:

$$(\text{CT}') \begin{cases} \text{Min.} & S_{n+1} \\ \text{s.t.} & (1) - (7) \\ & (10) - (14) \\ & S_i \in \mathbb{Z}_{\geq 0} \quad (i \in V \cup \{0, n+1\}) \\ & s_{il} \in \{0, 1\} \quad (i \in V; \ l \in L) \\ & r^u_{ik} \in \{0, 1\} \quad (i \in V; \ k \in R; \ u = 1, \ldots, R_k : \ r_{ik} > 0) \\ & y_{ij} \in \{0, 1\} \quad (i, j \in V^C \cup V^S : \ i \neq j, \ (i, j) \notin TE) \\ & y_{ij} \in [0, 1] \quad (i, j \in V : \ i \neq j, \ (i, j) \notin TE, \ \{i, j\} \not\subseteq V^C \cup V^S) \end{cases}$$

For instances that include activities with noninteger durations or noninteger transportation times, we set $S_i \in \mathbb{R}_{\geq 0}$ $(i \in V \cup \{0, n+1\})$. Moreover, we pass the initial site assignment

derived in step ③ as initial values for the corresponding variables of (CT′). Furthermore, the parameter c (see step ②) influences how many sequencing variables are defined as binary in the model (CT′). We discuss in the Appendix how an appropriate value for parameter c can be derived.

- In step ⑤, the model (CT′) is solved using a generic MBLP solver with a prescribed time limit. If no feasible solution is found within the time limit, the limit is extended until a feasible solution is found. If V^N is empty, we stop; otherwise, there are still activities that need to be scheduled, and we continue with step ⑥.
- In step ⑥, a partial solution is stored based on the devised solution to the model (CT′). The partial solution comprises the values of the start-time, the site-selection and the resource-assignment variables for the activities in $V^S \cup V^C$ as well as the values of the sequencing variables between the activities in $V^S \cup V^C$ in the devised solution.
- In step ⑦, s activities with $0 < s \leq c$ are selected from V^C based on the priority rule from step ②. These activities are then removed from V^C and included in V^S. For these activities, some variables are fixed in the next iteration (see constraints (12)–(14)).
- In step ⑧, s activities are selected from V^N based on the priority rule from step ②. These activities are removed from V^N and included in V^C to be scheduled in the next iteration. If there are fewer than s activities in V^N, all activities in V^N are selected.
- In step ⑨, because V^C was updated in steps ⑦ and ⑧, some binary restrictions that were relaxed are now restored (see the domains of the sequencing variables in (CT′)). Moreover, as V^S was updated in step ⑦, some variables that were not fixed are now fixed to their corresponding values (see constraints (12)–(14)). Finally, some redundant constraints are added (see constraints (10)–(11)). Next, the partial solution from step ⑥ is provided as initial values for the variables of the model (CT′) in the same way as the initial site assignment in step ④.

The matheuristic stops when—after step ⑤ is completed—V^N is empty and, thus, a feasible solution has been derived.

4.3 Illustration of the matheuristic

We illustrate the matheuristic with the example project from Sect. 2 with $c = 4$ and $s = 3$. First, the sets $V^S = \emptyset$, $V^C = \emptyset$, and $V^N = \{1, 2, 3, 4, 5, 6, 7\}$ are initialized.

Second, $c = 4$ activities are selected from V^N. Here, we apply the latest start time priority rule with the latest finish time priority rule as a tie breaker (LST + LFT); i.e., the activity with the smallest latest start time (and, in case of a tie, with the smallest latest finish time) is selected first. The latest start time and the latest finish time of each real activity are determined by (forward and) backward pass calculation (cf. Demeulemeester & Herroelen, 2002). The latest start time and the latest finish time are standard priority rules that have been shown to perform well in the recent scheduling literature (cf., e.g., Almeida et al., 2016) and perform well in our matheuristic. Table 2 states the latest start time and the latest finish time of each real activity, which results in $V^C = \{2, 1, 3, 4\}$ and $V^N = \{5, 6, 7\}$.

Third, the initial site assignment is derived by the ratio-based rule. The rule derives site A as the initial site for activities 1 and 4 and site B as the initial site for activities 2 and 3. For activity 4, for example, the resource type $k^* = 1$ is selected, since it has the largest ratio $\frac{r_{41}}{1+\sum_{u=1}^{R_1} M_{1u}} = \frac{1}{1+0} = 1$. Resource type $k^* = 1$ has two nonmobile resource units located at site A but none at site B. Thus, site A is selected as the initial site for activity 4.

Table 2 Latest start times (LST) and latest finish times (LFT) of the activities in the example project

Activity	1	2	3	4	5	6	7
LST	2	0	2	3	4	3	4
LFT	3	3	3	5	5	5	5

Fourth, the model (CT′) is set up, and the initial site assignment is passed to the variables of the model (CT′).

Fifth, the (CT′) is solved with the Gurobi solver and a time limit of 60 s. Figure 5 (top) shows the optimal solution obtained in the first iteration. The activities in V^C are highlighted in bold and the activities in V^N are highlighted as slightly transparent. Two pairs of activities $\{2, 5\}$ and $\{6, 7\}$ are executed in parallel, which is feasible because the binary restrictions on the sequencing variables between these pairs of activities were relaxed in the model (CT′) in the first iteration.

Sixth, the partial solution based on the current solution to the model (CT′) is stored. For example, for the site-selection variable s_{4A}, the value of 1 is stored.

Seventh, the $s = 3$ activities with the highest priority among the c activities in V^C are removed from V^C and included in V^S. In the first iteration, these are the activities $\{2, 1, 3\}$.

Eighth, $s = 3$ activities with the highest priority among the activities in V^N are removed from V^N and included in V^C. These are the activities $\{6, 5, 7\}$.

In Iteration 2, the activities $V^C = \{4, 6, 5, 7\}$ must be scheduled. Figure 5 (bottom) shows the optimal solution obtained in the second iteration. The activities V^S are not highlighted. No activities are executed in parallel because all activities are either in V^C or V^S and, thus, all sequencing variables are defined as binary. Activity 2 is scheduled one time unit later than in Iteration 1 even though it is in V^S. This is possible because only the sequencing variables between all activities in V^S are fixed, but not their start time variables. After Iteration 2, the matheuristic stops, and a feasible solution has been found that also turns out to be an optimal solution for our example project.

5 Computational results

In this section, we provide the results of our computational experiments in which we compare the model presented in Sect. 3 and the matheuristic presented in Sect. 4 with the state-of-the-art approaches from the literature presented by Laurent et al. (2017). In Sect. 5.1, we describe the experimental design, and in Sect. 5.2, we describe the metrics used to evaluate the performance of the examined approaches. In Sect. 5.3, we compare the performance of three exact approaches based on the presented continuous-time model and the discrete-time model proposed by Laurent et al. (2017) using a standard mathematical programming solver. In Sect. 5.4, we analyze the trade-off between solution quality and running time that can be controlled by setting the parameter values of the presented matheuristic, and we compare its performance to the performance of the four metaheuristics of Laurent et al. (2017).

5.1 Experimental design

We implemented all models and our matheuristic in Python 3.7, and as mathematical programming solver we used the Gurobi Optimizer 9.1 with the default solver settings. All computations were performed on an HP workstation with one Intel Xeon CPU with 2.20

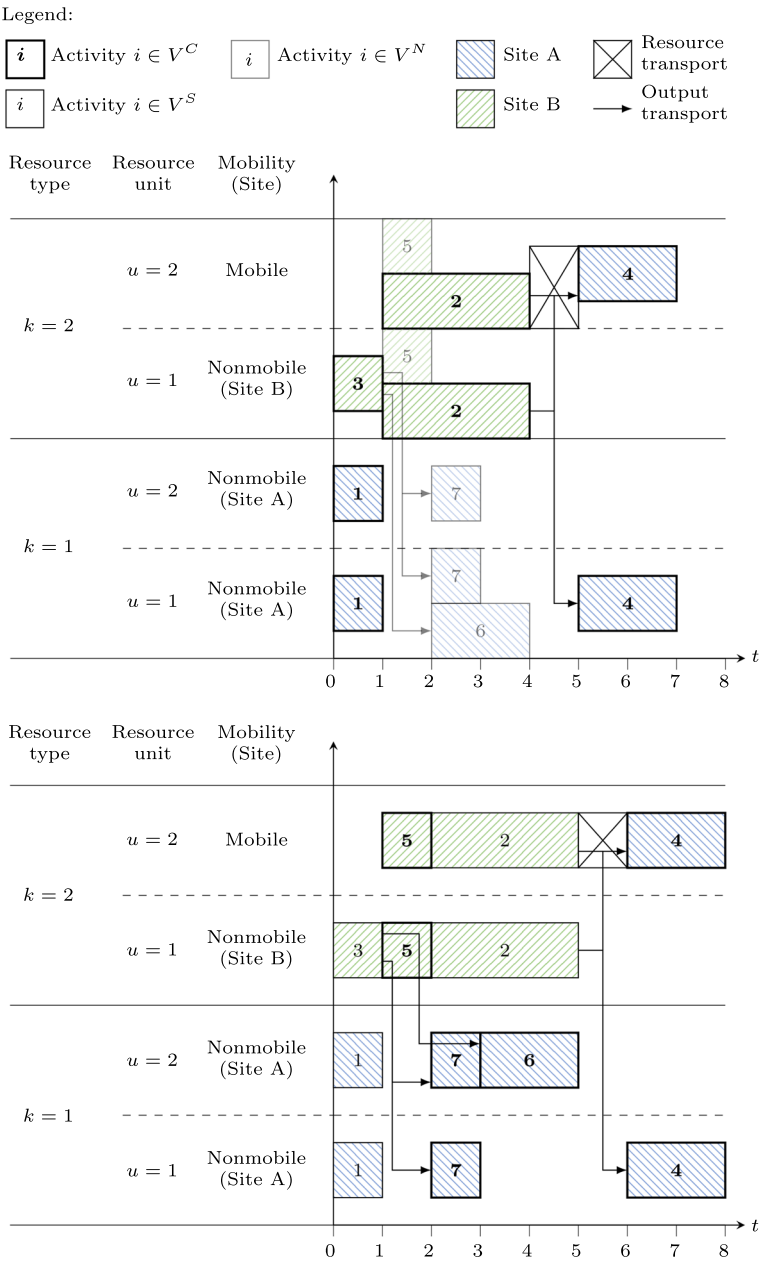

Fig. 5 Schedule for the example project derived by the matheuristic after Iteration 1 (top) and after Iteration 2 (bottom)

GHz clock speed and 128 GB RAM. For the three exact approaches, we set a time limit of 300 s. For the matheuristic, we used the same parameter setting as described in Sect. 4.3 except for the values of the parameters c and s (see Sect. 5.4 and the Appendix). The initial values passed to the model (CT′) in the matheuristic are provided to the Gurobi Optimizer by

the start attribute of the variables (cf. https://www.gurobi.com/documentation/9.1/refman/start.html). To ensure a fair comparison of the performance, we also ran the implementation of the four metaheuristics of Laurent et al. (2017) on the same workstation; this implementation was kindly provided by the authors. Moreover, Laurent et al. (2017) adjusted the test instances of the sets j30 and j60 from the PSPLIB (cf. Kolisch & Sprecher, 1996) to the multi-site context and named these newly generated sets MSj30 and MSj60, respectively. We used the same sets of instances for our experiments.

5.2 Description of performance metrics

To evaluate the performance of the tested exact and heuristic approaches, we use the following metrics:

- #Feas: Number of instances for which a feasible schedule is found within the prescribed time limit.
- #Opt: Number of instances for which a feasible schedule is found and proved to be optimal within the prescribed time limit.
- Gap^{CP} (%): Average relative deviation between the objective function value of the best solution returned by the respective approach (OFV) and the critical-path-based lower bound (CP), calculated as $(OFV - CP)/CP$.
- CPU (s): Average running time to derive the best solution returned by the respective approach.
- $\#\text{MH}^{+}$: Number of instances for which the matheuristic finds a solution with a lower OFV than the respective metaheuristic.
- $\#\text{MH}^{-}$: Number of instances for which the matheuristic finds a solution with a higher OFV than the respective metaheuristic.

In all tables, bold values indicate the best results among all tested approaches.

5.3 Computational results: Exact approaches

In Table 3, we report the results of the exact approaches based on the model (CT), the model (CT) with integer start-time variables, subsequently referred to as model (CT^{int}), and the discrete-time model of Laurent et al. (2017), subsequently referred to as model (DT), for all MSj30 instances. In addition, in Table 4, we present the corresponding results for the MSj30 instances for which each approach obtains at least a feasible solution.

With the model (CT), a feasible solution for all MSj30 instances is found. In contrast, with the model (DT), no feasible solutions are found for numerous instances. In terms of all other performance metrics, the model (CT) outperforms the model (DT); i.e., with the model (CT), a higher number of optimal solutions are derived, a lower average gap to the critical-path-based lower bound is achieved, and a lower average running time is required. Moreover, our results indicate that the problem becomes more challenging as the number of sites increases. We find that defining the start-time variables of the model (CT) as integer variables further improves its performance.

5.4 Computational results: Heuristic approaches

In this subsection, we analyze an important feature of our matheuristic, namely the possibility to control the trade-off between solution quality and computation time. Moreover, we com-

Table 3 Results for all MSj30 instances with the models (CT), (CT^{int}) and (DT)

Activities	Sites	Model	#Feas	#Opt	Gap^{CP} (%)	CPU (s)
30	2	(CT)	**480**	327	25.86	112.00
		(CT^{int})	**480**	**331**	**25.17**	**110.83**
		(DT)	455	272	34.93	159.44
30	3	(CT)	**480**	284	33.92	**138.40**
		(CT^{int})	**480**	**289**	**33.28**	139.25
		(DT)	444	224	51.80	190.93

Table 4 Results for the MSj30 instances for which with each of the models (CT), (CT^{int}) and (DT) at least a feasible solution was obtained

Activities	Sites	Model	#Feas	#Opt	Gap^{CP} (%)	CPU (s)
30	2	(CT)	455	323	22.04	102.92
		(CT^{int})	455	**325**	**21.35**	**102.80**
		(DT)	455	272	34.93	151.71
30	3	(CT)	444	279	28.51	**127.50**
		(CT^{int})	444	**284**	**27.91**	128.25
		(DT)	444	224	51.80	182.08

pare the performance of our matheuristic to the performance of the state-of-the-art heuristic approaches from the literature.

First, we examine the trade-off between solution quality and running time that can be controlled by the two parameters c and s of our matheuristic. Recall that parameter c indicates how many activities are scheduled in each iteration, and parameter s indicates how many activities are considered as scheduled after each iteration. To examine this trade-off, we ran our matheuristic for various values of parameter s, while fixing parameter c to a value of $c = 12$. For the sake of readability, we explain how we derived this value for parameter c in the Appendix.

In Table 5, we report the results of the matheuristic with these parameter settings for all MSj30 and MSj60 instances. The results indicate that decreasing parameter s improves the solution quality considerably at the expense of additional running time. Moreover, the results show that our matheuristic finds a feasible solution for all MSj30 and MSj60 instances within a reasonable running time. Additionally, compared to the exact approach based on the model (CT) with the imposed time limit of 300 s, the matheuristic on average derives solutions with a lower objective function value, while its average running time is considerably lower. We further evaluated the percentage of the running time that the matheuristic spends in the mathematical programming component (i.e., the total time used to set up and solve model (CT′)) versus the percentage of the running time it spends in the heuristic component (e.g., to apply the priority rule to select the first c or the next s activities). Overall, more than 99% of the total running time is spent in the mathematical programming component.

Second, we compare the performance of our matheuristic to the performance of the following four metaheuristics of Laurent et al. (2017): local search (LS), simulated annealing (SA), iterated local search with a better walk acceptance criterion (ILS BW) and iterated local

Table 5 Results for all MSj30 and MSj60 instances for the matheuristic (MH) for various values of s

Activities	Sites	Parameters of MH		#Feas	Gap^{CP} (%)	CPU (s)
30	2	$c = 12$	$s = 2$	480	**25.02**	61.86
			$s = 4$	480	25.48	36.75
			$s = 6$	480	26.01	28.81
			$s = 8$	480	27.13	24.39
			$s = 10$	480	27.98	21.14
			$s = 12$	480	29.48	**18.81**
30	3	$c = 12$	$s = 2$	480	**31.59**	102.57
			$s = 4$	480	32.35	61.32
			$s = 6$	480	32.86	47.97
			$s = 8$	480	34.21	38.83
			$s = 10$	480	34.90	34.07
			$s = 12$	480	37.20	**29.76**
60	2	$c = 12$	$s = 2$	480	**23.17**	277.73
			$s = 4$	480	23.91	164.50
			$s = 6$	480	24.99	123.18
			$s = 8$	480	25.59	100.40
			$s = 10$	480	26.62	88.50
			$s = 12$	480	28.05	**83.45**
60	3	$c = 12$	$s = 2$	480	**31.09**	375.90
			$s = 4$	480	32.52	221.63
			$s = 6$	480	33.45	166.20
			$s = 8$	480	35.42	137.08
			$s = 10$	480	38.14	122.94
			$s = 12$	480	39.52	**117.38**

search with a simulated annealing acceptance criterion (ILS SA). For a fair comparison, we use for each combination of the number of activities (30 and 60) and the number of sites (2 and 3) in the instances the parameter setting of the matheuristic for which the average running time is the closest to the average running times of the metaheuristics.

In Table 6, we report the results of the matheuristic with these parameter settings and the four metaheuristics for all MSj30 and MSj60 instances. For the MSj30 instances, the average running time of the matheuristic is slightly higher than the average running time of some of the metaheuristics, while the matheuristic has on average a better solution quality than do the four metaheuristics. For the MSj60 instances, the matheuristic has a shorter average running time than all four metaheuristics and derives solutions with a lower average gap to the critical-path-based lower bound. Moreover, we can see from Table 6 that for a large number of instances, the matheuristic derives a solution with a lower objective function value than the metaheuristics.

Table 6 Comparison of the matheuristic to the state-of-the-art heuristics for all MSj30 and MSj60 instances

Activities	Sites	Approach	#Feas	Gap^{CP} (%)	$\#\text{MH}^{+}$	$\#\text{MH}^{-}$	CPU (s)
30	2	MH ($c = 12, s = 2$)	480	**25.02**	0	0	61.86
		LS	480	29.72	258	61	**55.50**
		SA	480	26.50	178	93	55.51
		ILS LS	480	25.86	153	103	70.35
		ILS SA	480	26.04	152	110	70.80
30	3	MH ($c = 12, s = 4$)	480	**32.35**	0	0	61.32
		LS	480	37.65	276	89	**60.17**
		SA	480	34.11	219	119	60.44
		ILS LS	480	33.42	180	142	76.53
		ILS SA	480	33.37	192	152	76.42
60	2	MH ($c = 12, s = 6$)	480	**24.99**	0	0	**123.18**
		LS	480	27.95	231	133	128.97
		SA	480	26.19	211	168	130.26
		ILS LS	480	26.41	198	163	168.71
		ILS SA	480	26.57	197	161	168.68
60	3	MH ($c = 12, s = 8$)	480	**35.42**	0	0	**137.08**
		LS	480	38.29	289	123	142.76
		SA	480	35.51	235	172	143.66
		ILS LS	480	36.03	242	172	185.79
		ILS SA	480	36.43	249	149	185.98

6 Conclusion

In this paper, we studied a variant of the resource-constrained project scheduling problem in which activities can be scheduled at different sites and some of the resource units can be transported between sites. We proposed a continuous-time model and a matheuristic for this planning problem. The continuous-time model is based on continuous variables that represent the start times of the activities and binary variables that represent the selection of a site for the activities, the assignment of the activities to the resource units and the sequence between the activities if the activities are assigned to at least one common resource unit. The matheuristic is based on a variant of the continuous-time model in which the binary restrictions on the sequencing variables are initially relaxed for all activities and then iteratively restored for a subset of the activities that is scheduled in the current iteration. In each iteration of the matheuristic, all activities are taken into account when the respective subset of activities is scheduled. The model and the matheuristic outperform state-of-the-art exact and heuristic approaches from the literature in terms of various performance measures on a set of standard instances.

In future research, the developed approaches could be used to further analyze the benefit of pooling resources among different sites. Furthermore, for instances with substantially more than 60 activities, the time used to solve the variant of the continuous-time model in each iteration of the matheuristic may strongly increase. Thus, we suggest dividing these instances into several smaller instances and successively deriving a schedule for these smaller instances. Finally, similar matheuristics based on the relax-optimize-and-fix strategy could be designed

Table 7 Results for the MSj60 instances comprising 3 sites for the matheuristic for various values of c

Activities	Sites	c	#Feas	Gap^{CP} (%)	CPU (s)
60	3	8	480	33.90	230.04
		12	480	31.09	375.90
		16	480	31.21	470.80
		20	480	33.72	532.09

for related resource-constrained project scheduling problems. For the multi-mode RCPSP, for example, the sequencing between activities could be similarly disregarded in a first step and then iteratively determined.

Funding Open access funding provided by University of Bern

Appendix

We derived the value of parameter c used for the computational analysis in Sect. 5.4 by running our matheuristic for various values of parameter c, while fixing parameter s to a value of $s = 2$. We fixed the value of parameter s to a small value, since small values generally lead to a better solution quality than large values (at the expense of a longer running time). To derive a value for parameter c, we selected the largest instances comprising 60 activities and 3 sites.

In Table 7, we report the results of this examination. When parameter c is increased, the solution quality first improves and then deteriorates again while the average running time increases consistently. From the tested values of parameter c, we selected $c = 12$ because this value resulted in the best solution quality.

In general, the solution quality improves for increasing values of parameter c at the expense of additional running time. The deterioration of the solution quality that we observe for large values of parameter c, however, is due to the running time limit of 60 s imposed in each iteration of the matheuristic. We imposed this running time limit to prevent excessively long total running times. However, for large values of parameter c, this running time limit may force the solution process of model (CT′) to be terminated prematurely which may lead to a poor solution quality.

References

Almeida, B. F., Correia, I., & Saldanha-da-Gama, F. (2016). Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications, 57*, 91–103.

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: optimization, simulation and control. In C. Blum, L. Di Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *International Workshop on Hybrid Metaheuristics* (pp. 171–177). Springer.

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society, 53*(5), 512–522.

Della Croce, F., Grosso, A. C., & Salassa, F. (2013). Matheuristics: Embedding MILP solvers into heuristic algorithms for combinatorial optimization problems. In P. Siarry (Ed.), *Heuristics: Theory and Applications* (pp. 53–67). Nova Science Publishers.

Demeulemeester, E. L., & Herroelen, W. (2002). *Project Scheduling: A Research Handbook*. Kluwer Academic Publishers.

Gnägi, M.,& Trautmann, N. (2019). A continuous-time mixed-binary linear programming formulation for the multi-site resource-constrained project scheduling problem. In Wang M, Li J, Tsung F, Yeung A (eds.), Proceedings of the 2019 IEEE International Conference on Industrial Engineering and Engineering Management, pp 611–614

Gnägi, M., Rihm, T., Zimmermann, A., & Trautmann, N. (2019). Two continuous-time assignment-based models for the multi-mode resource-constrained project scheduling problem. *Computers & Industrial Engineering, 129*, 346–353.

Kadri, R.,& Boctor, F. (2014). Multi-mode resource-constrained project scheduling with sequence dependent transfer times. In Fliedner T, Kolisch R, Naber A (eds.), 14th International Conference on Project Management and Scheduling, pp 116–119

Kadri, R., & Boctor, F. (2018). An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research, 265*(2), 454–462.

Kolisch, R., & Sprecher, A. (1996). PSPLIB–a project scheduling problem library. *European Journal of Operational Research, 96*(1), 205–216.

Krüger, D., & Scholl, A. (2009). A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research, 197*(2), 492–508.

Krüger, D., & Scholl, A. (2010). Managing and modelling general resource transfers in (multi-)project scheduling. *OR Spectrum, 32*(2), 369–394.

Laurent, A., Deroussi, L., Grangeon, N., & Norre, S. (2017). A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics. *Computers & Industrial Engineering, 112*, 634–644.

Liu, Y., Zhou, J., Lim, A., & Hu, Q. (2021). Lower bounds and heuristics for the unit-capacity resource constrained project scheduling problem with transfer times. *Computers & Industrial Engineering, 161*, 107605.

Maniezzo, V., Boschetti, M. A., & Stützle, T. (2021). Preface. In V. Maniezzo, M. A. Boschetti, & T. Stützle (Eds.), *Matheuristics* (pp. 143–158). Springer.

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

Poppenborg, J., & Knust, S. (2016). A flow-based tabu search algorithm for the RCPSP with transfer times. *OR Spectrum, 38*(2), 305–334.

Rihm, T., & Trautmann, N. (2017). An assignment-based continuous-time MILP model for the resource-constrained project scheduling problem. In De Meyer A, Chai KH, Jiao R, Chen N, Xie M (eds.), Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management, pp 35–39

Sabzehparvar, M., & Seyed-Hosseini, S. M. (2008). A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *The Journal of Supercomputing, 44*(3), 257–273.