Journal of Global Optimization (2020) 76:211–241
https://doi.org/10.1007/s10898-019-00832-8

Multimode resource-constrained project scheduling in flexible projects

Zsolt T. Kosztyán[1] · **István Szalkai**[2]

Received: 3 May 2018 / Accepted: 21 September 2019 / Published online: 4 October 2019

Abstract
Flexible agile and extreme project management methods have become increasingly popular among practitioners, particularly in the IT and R&D sectors. In contrast to the theoretically and algorithmically well-established and developed trade-off and multimode methods applied in traditional project management methods, flexible project scheduling methods, which are applied in agile, hybrid, and especially extreme project management, lack a principled foundation and algorithmic handling. The aim of this paper is to fill this gap. We propose a matrix-based method that provides scores for alternative project plans that host flexible task dependencies and undecided, supplementary task completion while also handling the new but unplanned tasks. In addition, traditional multimode resource-constrained project scheduling problems are also covered. The proposed method can bridge the flexible and traditional approaches.

Keywords Multimode resource-constrained project scheduling problem · Flexible project management approaches · Matrix-based project planning

1 Introduction

The importance of scheduling and resource allocation problems in project management was recognized over five decades ago (see excellent reviews in Brucker et al. [2], Habibi et al. [11]). From the beginning of the 1960s until recently, researchers usually assumed trade-off functions between both the time and the cost and between the time and the resources (see, e.g., [20]) (see Fig. 1a) and later, also among the time, the cost and the quality [1,18].

This research is supported by a János Bolyai Fellowship, Hungarian Academy of Science and by the Research Centre at Faculty of Business and Economics (No. PE-GTK-GSKK A095000000-1) of University of Pannonia (Veszprém, Hungary).

✉ Zsolt T. Kosztyán
kzst@gtk.uni-pannon.hu

[1] Department of Quantitative Methods, Faculty of Business and Economics, University of Pannonia, Egyetem str. 10, Veszprém, Hungary

[2] Department of Mathematics, Faculty of Information Technology, University of Pannonia, Egyetem str. 10, Veszprém, Hungary

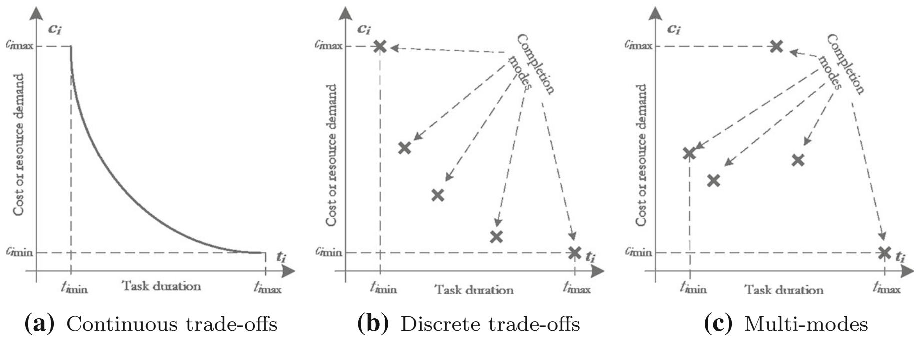


Fig. 1 Trade-off methods and multimode problems

Two main versions of trade-off methods are specified. First, from the 1960s to the 1980s, continuous time–cost relationship problems were addressed extensively in the literature (see, e.g., [9,17]). Researchers later investigated the discrete version of trade-off methods (see, e.g., [5,7,25]). Despite the ease of use, the main concern of the concept of continuous trade-off problems (CTP) is that it is usually hard to specify continuous trade-off functions between the time and the cost, the time and resources, or the cost and the quality. In real projects, decision makers instead choose from technologies (in addition to completion modes) (see Fig. 1b, c).

Moreover, this situation is better described by discrete trade-off problems (DTP) (see Fig. 1b). At the same time, CTP can be solved by fast polynomial algorithms (see, e.g., [10,19]), while the discrete version of trade-off problems, with the exception of a few kinds of project networks [5] is an NP-hard problem [6]; therefore, the discrete version of time–quality–cost trade-off problems are also NP-hard problems [25]. To solve DTPs, heuristic and metaheuristic algorithms are usually applied [23,26]. The recent multimode resource-constrained project scheduling problems (MRCPSP) (see an excellent review in Habibi et al. [11]), do not assume any trade-off functions between time and resources or between time and cost (see Fig. 1c). Trade-off methods are successfully used in construction projects [20]; in addition, the discrete version of trade-off methods and MRCPSPs are also successfully used in new product development projects [29] and investment projects [12], but they are difficult to use in software development projects and R&D projects where the project plans are more flexible. The main limitation of these concepts is that they both assume a fixed logic plan, such as a fixed set of tasks and a fixed sequence of completion. The proposed algorithm extends trade-off methods and MRCPSPs in order to handle flexible projects.

Previously, Wysocki found that in a [31] study of the practices of software project managers, only 20% of IT projects were managed by a traditional project management (TPM) methodology. Generally, methods for investment and construction projects cannot be directly applied to software development or R&D projects, as these are managed by agile project management (APM) approaches. Currently, hybrid (i.e., combinations of traditional and agile and extreme) approaches are becoming increasingly popular (see, e.g., [16]). However, flexible approaches are thus far not privileges for software development projects [3]. Rapidly changing environments increasingly enforce flexible approaches. Scheduling algorithms can support decision makers to manage projects; however, there are very few algorithmic methods that can support the flexible approaches. Therefore, it is important to study how to extend project scheduling methods to handle flexible and changing environments. Scheduling methods,

Table 1 Comparison of the traditional and the flexible approaches

Approaches	Features			
	Project structure	New tasks	Multimodes	Constraints
Traditional (TPM)	Fixed	Not allowed	Handled	Fixed
Agile (APM)	Flexible	Not allowed	Not handled	Fixed
Application management (AM)	Flexible	Allowed	Not handled	Fixed
Extreme (XPM)	Flexible	Allowed	Not handled	Flexible
Hybrid (HPM)	Flexible	Allowed	Handled	Optional

as agents, can also imitate decision makers; therefore, not only the methods but also the scheduling and management approaches can be modeled (see Table 1).

While a project manager who follows a traditional project management (TPM) approach can use trade-off or multimode methods to reduce task duration, the agile and extreme project manager tries to restructure the project. If the project structure is flexible (see Table 1), the project duration can be reduced without increasing the project cost by reducing the number of flexible dependencies. However, in real project situations, decision makers can choose from different kinds of technologies; therefore, the TPM and APM approaches should be integrated. Agile approaches usually split the projects into smaller so-called sprints that are usually 2–6 weeks. The content of sprints is specified by the customer and the developers together. However, when running a sprint, unplanned new tasks and new requirements can be involved only until the next sprint.

The extreme project management (XPM) approach must handle the new tasks and new requirements during the implementation of the project. Application management, which can be considered as a combination of project management and process management from a scheduling point of view, also allows management to involve new and unforeseen tasks in the project plan. However, while extreme project management can confirm the extra costs and the increased project duration due to the extra tasks, application management instead emphasizes adhering to a fixed budget and fixed time-frame.[1]

Flexible approaches require flexible project structures and flexible constraints; however, in addition to the opportunity of reorganizing the project, different kinds of technology (completion modes) should also be considered; therefore, traditional and flexible approaches should be combined into hybrid project management approaches (see, e.g., [16,21,27]). Nevertheless, hybrid approaches should be supported by algorithmic methods in order to help decision makers ensure the project's success.

There are different combinations of agile and traditional project management approaches (see, e.g., [16,21,27]). However, to the best of our knowledge, there is no exact algorithm that can be used to solve hybrid multimode problems that can handle unplanned tasks and dependencies. Nevertheless, R&D and IT projects, such as introducing and setting up new information systems, may require reorganizing part of the project, and R&D projects may require handling unplanned tasks, particularly in the development phase. However, decreasing the time demands of mandatory tasks and those of the new unplanned tasks may also be an important requirement. Neither the agile nor the extreme approach can handle this situation properly; nor can the traditional approach. Traditional approaches, or network-based methods, assume static logic plans, but the reorganization of projects may produce insuf-

[1] In application management the term of 'time-frame' is used instead of deadline.

ficient reductions in project duration and/or supplementary tasks, and important tasks may excluded from the project due to budget constraints and/or project deadlines. A hybrid project management (HPM) approach can combine the traditional, agile and extreme approaches; however, these kinds of HPM approaches are not yet supported by project planning methods. The proposed algorithm combines the agile, extreme and traditional approaches. This method extends the traditional multimode resource-constrained project scheduling problem by allowing the restructuring and reorganizing of projects and handling unplanned new tasks.

The proposed hybrid time–cost and hybrid time–quality–cost trade-off models multimode methods manage flexible project plans and allow us to restructure or reorganize these project plans in order to satisfy customer and management claims. In contrast to the traditional project scoring and selection methods, there is no need to specify all project alternatives in order to select the most desirable project scenario or the one with the shortest duration or lowest cost.

To handle flexible project plans, matrix-based techniques will be used instead of traditional network-based project planning techniques. This method has already been successfully used to model agile projects [15]. The basis of the proposed matrix-based method is a project domain matrix (PDM) (see [15]) with unplanned tasks. The modified PDM is an $n + u$ by $m + u$ matrix, where n is the number of planned tasks; u is the number of unplanned tasks; $m = n + t + c + q + r$; t is the number of possible durations; c is the number of possible (direct) costs; q is the number of possible quality parameters; and r is the number of possible resource demands of tasks. The PDM has five domains. The first domain is the logic domain (LD) that is described as an n by n project expert matrix (PEM) (see [14]) or numerical dependency structure matrix (NDSM) (see [24]).[2] Since the PEM has specified and semispecified versions, the PDM is *specified* if and only if the LD is specified; otherwise, the PDM is *semispecified*. The other domains are the time domain (TD), cost domain (CD), quality domain (QD) and resource domain (RD). If the demands are deterministic, we say that the PDM is *deterministic*; otherwise, the PDM is *nondeterministic*. In this study, the hybrid resource-constrained multimode project scheduling problems (HMRCPSP) are considered: the TD, CD, QD, and RD can contain deterministic values but must have at least two completion modes. Therefore, this version is a deterministic multimodal PDM. While the basis of the proposed model is the PDM, the basis of the proposed method is the expert project ranking (EPR) algorithm (see [15]), which can evaluate specified and semispecified deterministic PDMs. However, the EPR method cannot handle the resource-constrained multimode project scheduling problem, and the unplanned tasks and requirements. Therefore, while EPR can be used to schedule a flexible project plan and can thus be used in agile project management approaches, it cannot address multimodes and extra, unplanned tasks, and therefore cannot be directly used in extreme or hybrid project management.

This paper proposes a hybrid resource constrained multimode project scheduling problem by handling unplanned tasks to bridge APM, XPM and TPM and handle both the challenges of project and the application managements.

2 Solving hybrid multimode resource-constrained project scheduling problems

In this section, a resource-constrained hybrid multimode resource-constrained project scheduling problem (HMRCPSP) is first specified. Then, a matrix-based model represen-

[2] The NDSM does not represent supplementary tasks but can represent flexible dependencies; however, the PEM can represent both flexible dependencies and supplementary tasks.

tation will be proposed. At the end of this section, an exact algorithm is proposed for a hybrid multimode resource-constrained project scheduling problem. The decisions for finding the optimum will be directed by *score functions* and *matrices* (**P**, **Q**) and time–quality–cost–resource functions, so we need several definitions and notations before proceeding.

2.1 Definitions and problem statements

Definition 1 We call any finite set $A = \{a_1, \ldots, a_n\}$ the set of **possible activities** or **tasks** in the project. The subset of **supplementary tasks** is $\widetilde{A} = \{\widetilde{a}_1, \ldots, \widetilde{a}_\sigma\} \subseteq A$, where $\widetilde{A}$ is any fixed subset of A. Then, $\overline{A} = A \setminus \widetilde{A}$ is the subset of **mandatory tasks**. Another subset is $\hat{A} = \{\widehat{a}_1, \ldots, \widehat{a}_p\} \subseteqq A$, the set of **planned tasks,** so $\check{A} = A \setminus \hat{A}$ is the subset of **unplanned tasks.** Then, $\overline{\hat{A}} = \hat{A} \cap \overline{A}$ is the set of *planned mandatory* tasks, $\overline{\check{A}} = \check{A} \cap \overline{A}$ is the set of *unplanned mandatory* tasks, $\widetilde{\hat{A}} = \hat{A} \cap \widetilde{A}$ is the set of *planned supplementary* tasks, and $\widetilde{\check{A}} = \check{A} \cap \widetilde{A}$ is the set of *unplanned supplementary* tasks.

Clearly $A = \overline{\hat{A}} \cup \overline{\check{A}} \cup \widetilde{\hat{A}} \cup \widetilde{\check{A}}$ is a disjoint union (a partition of A).
The mathematical model in Sect. 2 does not distinguish supplementary and planned tasks, similarly, not the mandatory and unplanned ones. This distinction will be interesting in the simulation Sect. 3.

Definition 2 The **set of** (project) **scenarios** is $\Xi\,(A) := \left\{S \subseteq A : \overline{A} \subseteq S\right\}$. When we consider a fixed element $S \in \Xi\,(A)$, we call S a **realized** (project) **scenario**.

The traditional project management approach allows only mandatory tasks. There is no opportunity to remove tasks or exclude dependencies, while agile approaches prioritize tasks and can handle mandatory and supplementary tasks. While mandatory tasks must be realized, supplementary tasks can be omitted from the project. Decisions about *supplementary* task realization always have two options: include or exclude. When specifying project constraints, usually only the planned tasks are considered; however, the extreme project management approach allows new, unplanned tasks to be included in the projects. $S \subseteqq A$ denotes the set of tasks that will be fulfilled by the algorithm. The number of possible project scenarios is 2^σ, where $\sigma = \left|\widetilde{A}\right|$.

Definition 3 Any function $P : A \to [0, 1]$ is called a **score function of** task **completion** if $P\,(a_i) = 1$ for $a_i \in \overline{A}$ and $P\,(a_i) \in [0, 1)$ for $a_i \in \widetilde{A}$.
The function $Q : A \to [0, 1]$ is called a **score function of** task **exclusion** if $Q\,(a_i) = 0$ for $a_i \in \overline{A}$ and $Q\,(a_i) \in (0, 1]$ for $a_i \in \widetilde{A}$.

The task completion and exclusion scores can mean either probabilities or importance values.

Example 1 If every task completion score is a probability value, then $Q = 1 - P$.

Definition 4 For any associative and *monotone*[3] operation $\otimes$ on $\mathbb{R}^+$, we define the **aggregation function** $\otimes : \Xi\,(A) \to \mathbb{R}$ as

$$\otimes\,(S) := \bigotimes_{a \in S} P\,(a) \quad \otimes \quad \bigotimes_{a \in A \setminus S} Q\,(a)\,, \tag{1}$$

[3] $\otimes$ is **monotone** if $x \le y$ and $u \le v$ implies $x \otimes u \le y \otimes v$ for $x, y, u, v \in \mathbb{R}^+$.

where $\otimes$ indicates Σ or Π (addition or multiplication, respectively) or $\vee$ or $\wedge$ (maximum or minimum, respectively).

Example 2 If scores are probability values, then we have $\otimes = \Pi$.
If scores indicate the importance of task completion, then $\otimes = \Sigma$.

In PHASE ONE, we decide which tasks will be included in the project scenario by maximizing $\otimes(S)$, fulfilling certain *time*, *quality* and *cost* requirements.

In PHASE TWO, we decide *the order in which* we will complete these tasks. This order is a *relation* of tasks, and our algorithm receives three kind of relations (dependencies) as inputs: **no dependency**, **required** and **flexible**. We have to *resolve* (i.e., decide) all flexible relations.

Definition 5 The relation triplet $(\prec, \sim, \bowtie)$ is a **relation representation** of a hybrid project plan if for any $i, j \leq n$ and $i \neq j$, we let
(i) $a_i \prec a_j$ represent the **strict** or **required** dependencies between tasks a_i and a_j, i.e., a_j may not be started unless activity a_i has been completed;
(ii) $a_i \sim a_j$ represent **no dependency** between tasks a_i and a_j, i.e., the starting time of a_j is not affected by a_i; and
(iii) $a_i \bowtie a_j$ represents **flexible** dependencies between tasks a_i and a_j.

Remark 1 It is important to note that $\prec$ is naturally a transitive, irreflexive and asymmetric relation, i.e. a **strict partial ordering**, which excludes *circles* like $a_1 \prec a_2 \prec \cdots \prec a_1$. Therefore, by a standard topological ordering algorithm, we may assume that $a_i \prec a_j \implies i < j$.

While *strict* dependencies $a_i \prec a_j$ between tasks a_i and a_j must be realized (in a sequential way), and $a_i \sim a_j$ indicates indifference (we may choose either sequential or parallel realization), *flexible* dependencies $(\bowtie)$ must be resolved. In this case, we can decide whether these tasks will be completed in a *sequential* way, i.e., $a_i \prec a_j$ will be declared, or in a *parallel* way, $a_i \sim a_j$. When we decide that these tasks should be completed in either a sequential or a parallel way, we say that the flexible dependency is **realized** (or **decided**). Since for any $i, j, i \neq j$ *exactly one* of $a_i \prec a_j$, $a_i \succ a_j$, $a_i \sim a_j$ or $a_i \bowtie a_j$ holds, and for $i = j$ ("in the diagonal"), none of these four is valid; then, we have the following result:

Proposition 1 *For any binary relations $\prec, \sim, \bowtie$ on A, the triplet $(\prec, \sim, \bowtie)$ is a relation representation of a hybrid project plan if and only if $\{\prec, \prec^{-1}, \sim, \bowtie\}$ is a partition of $A \times A \setminus \iota$ ($\prec^{-1}$ is the* reverse *of $\prec$, i.e., $\succ$, and ι is the diagonal) and $\prec$ is a strict partial ordering.*

The easiest way to input and modify *all* data is by using a special $n \times n$ *matrix* $\mathbf{M}$, which we call a *hybrid logic plan*. ($[\mathbf{M}]_{i,j}$ denotes the *entry* in row i and column j.)

Definition 6 The **matrix representation** of an **input** for a hybrid logic plan means any $n \times n$ matrix with entries $\emptyset$, X, ?, i.e., $\mathbf{M} \in \{X, \emptyset, ?\}^{n\times n}$, assuming the following requirements:
(i) for any $i \leq n$, $[\mathbf{M}]_{i,i} =$ "X" for $a_i \in \overline{A}$ and $[\mathbf{M}]_{i,i} =$ "?" for $a_i \in \widetilde{A}$; and
(ii) for any $i, j \leq n, i \neq j$ $[\mathbf{M}]_{i,j} =$ "X" $\iff a_i \prec a_j$, $[\mathbf{M}]_{i,j} =$ "$\emptyset$" $\iff a_i \sim a_j$ and $[\mathbf{M}]_{i,j} =$ "?" $\iff a_i \bowtie a_j$.

In the diagonal, $[\mathbf{M}]_{i,i} =$ "X", "$\emptyset$", "?" represents that the task a_i will be executed, will not be executed, or will be decided later, respectively. Similarly, $[\mathbf{M}]_{i,j} =$ "?" for any $i \neq j$

indicates flexible task dependency, which we have to decide in the algorithm.[4] The algorithm will change each "?″ to either "X" or "∅" in $\mathbf{M}$ step-by-step in the diagonal in PHASE ONE, and in the off-diagonal in PHASE TWO. From Remark 1, we know that considering the entries X, $\mathbf{M}$ is an upper triangular matrix both in the input and in all subsequent steps of the algorithm. In what follows, $\mathbf{M}$ is any *such* matrix $\mathbf{M} \in \{X, \emptyset, ?\}^{n\times n}$.

Proposition 2 *For any realized project scenario $S \subseteq A$, the* diagonal *of the matrix representation does not contain "?": $\mathbf{M}_{i,i}$ = "X" for $a_i \in S$ and $\mathbf{M}_{i,i}$ = "∅" for $a_i \in A \setminus S$. Further, $\mathbf{M}$ does not contain "?" at all when all flexible dependencies are decided.*

Since we plan to omit all tasks $a \in A \setminus S$, we might also omit the rows and columns of $\mathbf{M}$ not belonging to S. This might be important in computer runs (saving time and memory) and could be defined axiomatically. We leave these details to the readers.

Similar to Definition 3, we are given *score values* for each dependency.

Definition 7 Any two *matrices* $\mathbf{P}, \mathbf{Q} \in [0, 1]^{n\times n}$ are called **score functions** of the *input* matrix $\mathbf{M}$ if
(i) for every $i \le n$, we have $[\mathbf{P}]_{i,i} = P(a_i)$ and $[\mathbf{Q}]_{i,i} = Q(a_i)$;[5]
(ii) for every $i, j \le n$, $i \ne j$ we have:
$[\mathbf{M}]_{i,j}$ = "X" $\iff$ $[\mathbf{P}]_{i,j} = 1$ and $[\mathbf{Q}]_{i,j} = 0$,
$[\mathbf{M}]_{i,j}$ = "∅" $\iff$ $[\mathbf{P}]_{i,j} = 0$ and $[\mathbf{Q}]_{i,j} = 1$, and
$[\mathbf{M}]_{i,j}$ = "?" $\iff$ $0 < [\mathbf{P}]_{i,j}, [\mathbf{Q}]_{i,j} < 1$.
Let us emphasize that *(i)* and *(ii)* above are requirements for the *input* matrix $\mathbf{M}$. While the "?" entries in $\mathbf{M}$ are changing during the algorithm run, $\mathbf{P}$ and $\mathbf{Q}$ remain unchanged.

The off-diagonal values of $[\mathbf{P}]_{i,j}$ represent the score of realizing the task dependency between a_i and a_j, while $[\mathbf{Q}]_{i,j}$ represents the score of resolving the task dependency between a_i and a_j.

From the diagonal of $\mathbf{M}$, we can provide sharp bounds to $\otimes(S)$ (see (1)) in each step of the algorithm.

Definition 8 For any $\mathbf{M}, \mathbf{N} \in \{X, \emptyset, ?\}^{n\times n}$, we say that
(i) $\mathbf{N}$ is an **in-/out-**diagonal **extension** of $\mathbf{M}$ if all the symbols "X" and "∅" in $\mathbf{M}$ remain unchanged in $\mathbf{N}$, and some (possibly none) of the "?" *inside/outside* of the diagonal of $\mathbf{M}$ are changed to "X" or "∅". In this case, $\mathbf{M}$ is a **restriction** of $\mathbf{N}$.
(ii) For $i, j \le n$ and $[\mathbf{M}]_{i,j}$ = "?", we denote by $\mathbf{M}[i, j = X]$ and $\mathbf{M}[i, j = \emptyset]$ the matrices if only $\mathbf{M}_{i,j}$ has been changed to either "X" or "∅".
(iii) $\mathbf{N}$ is an **in-/out-** diagonal **closure** of $\mathbf{M}$ if $\mathbf{N}$ contains no "?s" in/out of the diagonal and $\mathbf{N}$ in/out extends $\mathbf{M}$.

When replacing a "?", the algorithm makes newer *extensions* of the recent (or input) matrix, and our goals in PHASE ONE and PHASE TWO are suitable *in-* and *out-closures* of the input matrix, respectively. *In-closures* represent scenarios, while *out-closures* decide all flexible dependencies.

[4] The reader is allowed to replace the symbols "∅", "X" and "?" with any numbers. In Table 2, we use 0 for ∅, 1 for X and numbers from (0, 1) for ?.

[5] P and Q were introduced in Definition 3.

Definition 9

$$\otimes_{diag}(\mathbf{M}) := \bigotimes_{[\mathbf{M}]_{i,i}=X} P(a_i) \otimes \bigotimes_{[\mathbf{M}]_{i,i}=\emptyset} Q(a_i), \tag{2}$$

$$\otimes_{diag}^{\min}(\mathbf{M}) := \otimes_{diag}(\mathbf{M}) \otimes \bigotimes_{[\mathbf{M}]_{i,i}=?} \min\{P(a_i), Q(a_i)\}, \tag{3}$$

$$\otimes_{diag}^{\max}(\mathbf{M}) := \otimes_{diag}(\mathbf{M}) \otimes \bigotimes_{[\mathbf{M}]_{i,i}=?} \max\{P(a_i), Q(a_i)\}. \tag{4}$$

For some $\otimes$ (e.g., multiplying probabilities), $\otimes_{diag}^{\max}(\mathbf{M})$ may be smaller than $\otimes_{diag}(\mathbf{M})$ (see also Definition 7). However, we have the following inequalities.

Theorem 1 *For any* $\mathbf{M}$*,*

$$\otimes_{diag}^{\min}(\mathbf{M}) \le \otimes_{diag}^{\max}(\mathbf{M}). \tag{5}$$

For any in-extension $\mathbf{N}$ *of* $\mathbf{M}$*,*

$$\otimes_{diag}^{\min}(\mathbf{M}) \le \otimes_{diag}^{\min}(\mathbf{N}) \le \otimes_{diag}^{\max}(\mathbf{N}) \le \otimes_{diag}^{\max}(\mathbf{M}), \tag{6}$$

and for any $i \le n$*,*

$$\otimes_{diag}^{\min}(\mathbf{M}) = \min\left\{\otimes_{diag}^{\min}(\mathbf{M}[i,i=X])\ ,\ \otimes_{diag}^{\min}(\mathbf{M}[i,i=\emptyset])\right\}, \tag{7}$$

$$\otimes_{diag}^{\max}(\mathbf{M}) = \max\left\{\otimes_{diag}^{\max}(\mathbf{M}[i,i=X])\ ,\ \otimes_{diag}^{\max}(\mathbf{M}[i,i=\emptyset])\right\}. \tag{8}$$

For any in-closure $\mathbf{N}$ *of* $\mathbf{M}$*,*

$$\otimes_{diag}^{\min}(\mathbf{N}) = \otimes_{diag}(\mathbf{N}) = \otimes_{diag}^{\max}(\mathbf{N}) \tag{9}$$

and

$$\otimes_{diag}^{\min}(\mathbf{M}) \le \otimes_{diag}(\mathbf{N}) \le \otimes_{diag}^{\max}(\mathbf{M}). \tag{10}$$

For any $\mathbf{M}$*, there are* $\mathbf{N}_1$ *and* $\mathbf{N}_2$ in-closures *such that*

$$\otimes_{diag}^{\min}(\mathbf{M}) = \otimes_{diag}(\mathbf{N}_1) \quad \textit{and} \quad \otimes_{diag}(\mathbf{N}_2) = \otimes_{diag}^{\max}(\mathbf{M}). \ \square \tag{11}$$

Corollary 1 *If* $\mathbf{M}$ *contains no "?" in the diagonal, then for the represented scenario* $S \subseteq A$*, we have*

$$\otimes(S) = \otimes_{diag}(\mathbf{M}). \tag{12}$$

In PHASE THREE, we will have to decide *how to handle* the elements of A by choosing elements from the sets of **modes** called *protocols*.

Definition 10 **(i)** Any finite set of quadruplets $W = \{(t_i, q_i, c_i, \mathbf{r}_i) : i = 1, \ldots, k\}$, of *positive* real numbers and the vector $\mathbf{r}_i = \{r_{i,1}, .., r_{i,\rho}\}$ is called a **discrete multimode protocol** (**DMMp**). We write $t_{\min}$, $t_{\max}$, $q_{\min}$, $q_{\max}$, $c_{\min}$, $c_{\max}$, $r_{\min}$ and $r_{\max}$ instead of $\min_i t_i$, $\max_i t_i$, $\min_i q_i$, $\max_i q_i$, $\min_i c_i$, $\max_i c_i$, $\min_i \min_j r_{i,j}$ and $\max_i \max_j r_{i,j}$ respectively.
If for each $a \in A$ we are given a protocol W_a, then we call the set $\mathcal{W} = \{W_a : a \in A\}$ a **discrete multimode problem** (**DMMP**) on A.
(ii) Any *positive, continuous, strictly decreasing function* $w : [t_{\min}, t_{\max}] \to [q_{\min}, q_{\max}] \times [c_{\min}, c_{\max}] \times [r_{\min}, r_{\max}]$ is called a **continuous multimode protocol** (**CMMp**) with resource demands, where $0 < t_{\min}, t_{\max}, q_{\min}, q_{\max}, c_{\min}, c_{\max}, r_{\min}$, and $r_{\max}$ are also assumed. If for each $a \in A$, we are given a protocol w_a. Then, we call the set $\mathcal{W} = \{w_a : a \in A\}$ a continuous multimode **problem** (**CMMP**).

We interpret $(t, q, c, \mathbf{r}) \in W_a$ or $w_a(t) = (q, c, \mathbf{r})$ as paying **cost** c with **quality** q and with **resource r** to solve the element $a \in A$ in **time** t using the **mode** assigned to $(t, q, c, \mathbf{r})$. For a parallel explanation of discrete and continuous problems, we write $(t, q, c, \mathbf{r}) \in w_a$ *in both cases*. If ($t_i > t_j \Rightarrow c_i \le c_j, r_{i_1} \le r_{j_1}, .., r_{i_\rho} \le r_{j_\rho}$) and ($c_k > c_l \Rightarrow q_k \le q_l$) are satisfied for all $i, j, k, l = 1, .., n$, then a special case of multimode problems, namely, **time–quality–resource–cost trade off problems** are considered. The elements $t^a_{\min}$, $t^a_{\max}$, $q^a_{\min}$, $q^a_{\max}$, $c^a_{\min}$, $c^a_{\max}$, $r^a_{\min}$ and $r^a_{\max}$ may be different in different protocols w_a (W_a) for each $a \in A$ in general. The cases $t_{\min} = t_{\max}$, $q_{\min} = q_{\max}$, $c_{\min} = c_{\max}$ or $r_{\min} = r_{\max}$ are also allowed.
The final goal of our algorithm is an optimal project *schedule*.

Definition 11 Let $S \in \Xi(A)$ be any realized project scenario; $\mathcal{W}$ is either a CMMP or a DMMP. A **project schedule** is a set

$$\overrightarrow{w} = \left\{\left(t^a, q^a, c^a, \mathbf{r}^a\right) : a \in S\right\}, \tag{13}$$

where $(t^a, q^a, c^a, \mathbf{r}^a) \in w_a$ for $a \in S$.

We are now ready to provide upper and lower *bounds* for time and cost in each stage (in any PHASE) of the algorithm, that is, for any matrix $\mathbf{M} \in \{X, \emptyset, ?\}^{n\times n}$.

Definition 12 **(i)** For any **M** and $\mathcal{W}$ DMMP or CMMP, the minimal **cost bound** is

$$C_{\min}(\mathbf{M}, \mathcal{W}) := \sum_{[\mathbf{M}]_{i,i}=\text{“}X\text{”}} c^a_{\min}. \tag{14}$$

(ii) For any project schedule $\overrightarrow{w}$ the **total** *project* **cost** of $\overrightarrow{w}$ is

$$\mathbf{c}\left(\overrightarrow{w}\right) := \sum_{(t^a, q^a, c^a, \mathbf{r}^a)\in w_a\ ,\ a\in S} c^a. \tag{15}$$

To quantify project quality, both quality parameters and the task completion scores ($[\mathbf{P}]_{i,i}$) are considered.

Definition 13 **(i)** For any **M** and $\mathcal{W}$ DMMP or CMMP, the maximal **(relative) quality bound** is

$$Q_{\max}(\mathbf{M}, \mathcal{W}) := 1. \tag{16}$$

(ii) For any project schedule $\overrightarrow{w}$, the **total** *project* **quality** of $\overrightarrow{w}$ is

$$\mathbf{q}\left(\overrightarrow{w}\right) := \frac{\sum_{(t^a, q^a, c^a, \mathbf{r}^a)\in w_a\ ,\ a\in S} q^a}{\sum_{a\in A} q^a_{\max}}. \tag{17}$$

For time bounds, we must not forget the $\prec$ dependencies.

Definition 14 **(i)** For any *real* **path**

$$\overrightarrow{P} = \text{“}a_{i_1} \prec a_{i_2} \prec \cdots \prec a_{i_k}\text{”} \tag{18}$$

($\mathbf{M}_{i_j, i_{j+1}} = \text{“}X\text{”}$ for $1 \le j < k$), the minimal **time bound** of the path is

$$T_{\min}\left(\overrightarrow{P}, \mathcal{W}\right) := \sum_{a\in \overrightarrow{P}} t^a_{\min}. \tag{19}$$

(ii) $\overrightarrow{P}$ is the **longest minimum path** of $\mathbf{M}$ if $T_{\min}\left(\overrightarrow{P}, \mathcal{W}\right)$ is *maximal*, assuming that $\overrightarrow{P}$ contains mandatory tasks only (i.e., assuming $\mathbf{M}_{i,i} =$ "X" whenever $a_i \in \overrightarrow{P}$).

We denote this maximum by

$$T_{\min}(\mathbf{M}, \mathcal{W}) := \max_{P} T_{\min}\left(\overrightarrow{P}, \mathcal{W}\right). \tag{20}$$

Thus, $\overrightarrow{P}$ is called a **critical path**, and $\left\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\right\}$ is the set of **critical activities**.

(iii) For any project schedule $\overrightarrow{w}$, the **total** *project* **time** of $\overrightarrow{w}$ is

$$\mathbf{t}\left(\overrightarrow{w}\right) := \sum_{(t^a, q^a, c^a, \mathbf{r}^a) \in w_a\ ,\ a \in \overrightarrow{P}} t^a, \tag{21}$$

where $\overrightarrow{P}$ is any longest minimum path.

The length and definition of the longest minimum path do not depend on the project schedule $\overrightarrow{w}$ since t^a_{min} are summed in Eq. 21. Critical paths are in fact longest minimum paths. Clearly, $t(\overrightarrow{w}) \geq T_{min}(\mathbf{M}, \mathcal{W})$ for any $\mathcal{W}$ and $\overrightarrow{w}$. A *longest minimum path* in any $\mathbf{M}$ can be found by a standard algorithm within $O(n + d)$, where n is the number of tasks, and d is the number of dependencies.

Definition 15 Denote $A(\overrightarrow{w}, t) \subseteq A$ the set of running activities in time t for the schedule $\overrightarrow{w}$. The **maximal resource demands** for resource k are:

$$r_k(\overrightarrow{w}) := \max_t \sum_{a_i \in A(\overrightarrow{w}, t)} r_{i,k}, \qquad (k := 1, .., \rho) \tag{22}$$

and the **total resource vector** is

$$\mathbf{r}\left(\overrightarrow{w}\right) := \left(r_1\left(\overrightarrow{w}\right), \ldots, r_\rho\left(\overrightarrow{w}\right)\right). \tag{23}$$

Theorem 2 *For any* $\mathbf{M}$ *and for any* in- *or* out-closure $\mathbf{N}$ *of* $\mathbf{M}$*,*

$$C_{\min}(\mathbf{M}, \mathcal{W}) \leq C_{\min}(\mathbf{N}, \mathcal{W}), \tag{24}$$

$$Q_{\max}(\mathbf{M}, \mathcal{W}) \geq Q_{\max}(\mathbf{N}, \mathcal{W}) \tag{25}$$

and

$$T_{\min}(\mathbf{M}, \mathcal{W}) \leq T_{\min}(\mathbf{N}, \mathcal{W}). \tag{26}$$

Further, for any project schedule $\overrightarrow{w}$ *(for the scenario* S *, determined by the diagonal of* $\mathbf{N}$*)*

$$C_{\min}(\mathbf{N}, \mathcal{W}) \leq \mathbf{c}\left(\overrightarrow{w}\right), \tag{27}$$

$$Q_{\max}(\mathbf{N}, \mathcal{W}) \geq \mathbf{q}\left(\overrightarrow{w}\right) \tag{28}$$

and

$$T_{\min}(\mathbf{N}, \mathcal{W}) \leq \mathbf{t}\left(\overrightarrow{w}\right). \tag{29}$$

For $\mathbf{M} \in \{X, \emptyset\}^{n \times n}$ and $\overrightarrow{w}$, we also use the following notations for **total project quality**, **cost**, **time** and **resource** demand:

$$TPC\left(\mathbf{M}, \overrightarrow{w}\right) := \mathbf{c}\left(\overrightarrow{w}\right), \tag{30}$$

$$TPQ\left(\mathbf{M}, \overrightarrow{w}\right) := \mathbf{q}\left(\overrightarrow{w}\right) \tag{31}$$

and

$$TPT\left(\mathbf{M},\overrightarrow{w}\right) := \mathbf{t}\left(\overrightarrow{w}\right) \tag{32}$$

and

$$\mathbf{TPR}\left(\mathbf{M},\overrightarrow{w}\right) := \left[r_1(\overrightarrow{w})\ ,..,r_\rho(\overrightarrow{w})\right]^T. \tag{33}$$

Definition 16 A project **structure** or **net** means a *deterministic logic plan*, i.e., every flexible relation is realized. The triplet $\mathcal{X} = (S, \prec, \sim)$ represents the project structure of a given scenario $S \subseteq A$.

The matrix representation of a project *structure* contains no "?" symbols at all.

In PHASE TWO, we have to address the *score values* of the *dependencies* on A using the off-diagonal elements of P and Q (see Definition 7).

Definition 17 For any associative operation $\otimes$ on $\mathbb{R}$, we define the **aggregation function for project structures** as

$$\otimes_{nd}(\mathbf{M}) := \bigotimes_{\mathbf{M}_{i,j}=\text{``}X\text{''},\ i\neq j} \mathbf{P}_{i,j} \otimes \bigotimes_{\mathbf{M}_{i,j}=\text{``}\emptyset\text{''},\ i\neq j} \mathbf{Q}_{i,j} \tag{34}$$

and its extreme values

$$\otimes_{nd}^{\min}(\mathbf{M}) := \otimes_{nd}(\mathbf{M}) \otimes \bigotimes_{\mathbf{M}_{i,j}=?,i\neq j} \min\left\{\mathbf{P}_{i,j},\mathbf{Q}_{i,j}\right\}, \tag{35}$$

$$\otimes_{nd}^{\max}(\mathbf{M}) := \otimes_{nd}(\mathbf{M}) \otimes \bigotimes_{\mathbf{M}_{i,j}=?,i\neq j} \max\left\{\mathbf{P}_{i,j},\mathbf{Q}_{i,j}\right\}. \tag{36}$$

If $\mathbf{M}$ is the matrix representation of a realized project structure, $\otimes_{nd}(\mathbf{M})$ gives the *score value of* this *project structure*.[6]

2.1.1 Calculating constraints

In all versions of the multimode functions, the maximal/minimal time, quality and cost demands can be determined for all activities. In this way, the maximal/minimal total project time (TPT), total project quality and maximal/minimal total project cost (TPC) can be determined. This feature will be used when calculating the maximal and minimal demands for a project scenario and the maximal/minimal duration of a project structure. In the case of resource-constrained HMRCPSP problems, activities can be supplementary, and the dependencies can be flexible. Unplanned tasks and dependencies can modify the project plan. If we exclude a task from the project, we also exclude the time/quality and cost/resource demands of this activity.

$\text{TPC}_{\min}$:	Minimal value of total project cost ($\text{TPC}_{\min}$) occurs if only mandatory tasks are included in the project with the minimal possible cost.
$\text{TPC}_{\max}$:	Maximal value of total project cost ($\text{TPC}_{\max}$) occurs if all the mandatory and supplementary tasks are included in the project with the maximal possible cost.

[6] The abbreviation *nd* in the index means "*no diagonal*".

$\text{TPQ}_{\min}$:	Minimal value of total project quality ($\text{TPQ}_{\min}$) occurs if only mandatory tasks are included in the project with the minimal possible quality parameter.
$\text{TPQ}_{\max}$:	Maximal value of total project quality ($\text{TPQ}_{\max}$) occurs if all the mandatory and supplementary tasks are included in the project with the maximal possible quality parameter.
$\mathbf{TPR}_{\min}$:	Minimal value of total project resources ($\mathbf{TPR}_{\min}$) occurs if only mandatory tasks are included in the project with minimal possible resource demands.
$\mathbf{TPR}_{\max}$:	Maximal value of total project resources ($\mathbf{TPR}_{\max}$) occurs if all the mandatory and supplementary tasks are included in the project with maximal resource demands.
$\text{TPT}_{\min}$:	Minimal value of total project time ($\text{TPT}_{\min}$) occurs, if only mandatory tasks and fixed dependencies are included to the project, with minimal possible duration.
$\text{TPT}_{\max}$:	Maximal value of total project cost ($\text{TPT}_{\max}$) occurs if all the mandatory and supplementary tasks and all the flexible and fixed dependencies are included in the project with the maximal possible duration.
$\text{TPS}_{\min} = \otimes_{diag}^{\min}$:	Minimal value of total project score ($\text{TPS}_{\min}$) occurs if all the mandatory and supplementary tasks are included, where the score of exclusion is higher than the score of inclusion.
$\text{TPS}_{\max} = \otimes_{diag}^{\max}$:	Maximal value of total project score ($\text{TPS}_{\min}$) occurs if all the mandatory and supplementary tasks are included, where the score of inclusion is higher than the score of exclusion.
$\otimes_{nd}^{\min}$:	Minimal value of dependency scores occurs if all the fixed and flexible dependencies are included, where the score of exclusion is higher than the score of inclusion.
$\otimes_{nd}^{\max}$:	Maximal value of dependency scores occurs if all the fixed and flexible dependencies are included, where the score of inclusion is higher than the score of exclusion.

Instead, PHASES 1 through 3 are directed by (37)–(56), where the constants C_c, C_t, C_q, $\mathbf{C_r}$, C_{diag} and C_{nd} might be varied upon request; however, we assume that $C_t \in [\text{TPT}_{\min}, \text{TPT}_{\max}]$, $C_c \in [\text{TPC}_{\min}, \text{TPC}_{\max}]$, $C_q \in [\text{TPQ}_{\min}, \text{TPQ}_{\max}]$, $\mathbf{C_r} \in [\mathbf{TPR}_{\min}, \mathbf{TPR}_{\max}]$, $C_{diag} \in [\otimes_{diag}^{\min}, \otimes_{diag}^{\max}]$, $C_{nd} \in [\otimes_{nd}^{\min}, \otimes_{nd}^{\max}]$.

2.1.2 Possible target function

If all constraints are specified, we are ready to specify target functions in all phases of the algorithm. These targets can be different in all phases. We specified several possible target functions, but other target functions can be specified for other applications

$\otimes \to$ max:	Maximize score values in order to select the most desired project plan.
TPT$\to$ min:	Minimize project duration.
TPC$\to$ min:	Minimize project cost.
TPQ$\to$ max:	Minimize project quality.

We are now ready to define the *problems* we will solve in PHASES ONE, TWO and THREE.

2.1.3 Hybrid multimode resource-constrained project scheduling problem

In this problem, we try to select the most desired project plans with minimal project duration, while keeping all the time/cost/resource/quality constraints.

Problem 1 PHASE ONE: Let A be a finite set of activities and $\mathbf{M}$ be a matrix representation of A. Let $C_c, C_t, C_{diag} \in \mathbb{R}^+$ be given such that $C_{\min}(\mathbf{M}, \mathcal{W}) \leq C_c$, $T_{\min}(\mathbf{M}, \mathcal{W}) \leq C_t$ and $C_{diag} \leq \otimes_{diag}^{\max}(\mathbf{M})$. Now, find a *scenario* $S \subseteq A$, i.e., an *in-closure* $\mathbf{M}'$ of $\mathbf{M}$ such that

$$\otimes_{diag}\left(\mathbf{M}'\right) \rightarrow \max \tag{37}$$

assuming

$$C_{\min}\left(\mathbf{M}', \mathcal{W}\right) \leq C_c, \tag{38}$$
$$T_{\min}\left(\mathbf{M}', \mathcal{W}\right) \leq C_t, \tag{39}$$
$$\otimes_{diag}\left(\mathbf{M}'\right) \geq C_{diag}, \tag{40}$$
$$Q_{max}\left(\mathbf{M}', \mathcal{W}\right) \geq C_q. \tag{41}$$

The role of (40) is to stop the algorithm from searching for the maximum in (37) when (40), together with (38), (39) and (41), cannot be achieved.

Problem 2 PHASE TWO: Let $\mathbf{M}'$ be a solution to Problem 1 PHASE ONE, i.e., a matrix representation of a project scenario $S \subseteq A$. Let $C_t, C_{nd} \in \mathbb{R}^+$ be given such that $T_{\min}\left(\mathbf{M}', \mathcal{W}\right) \leq C_t$ and $C_{nd} \leq \otimes_{nd}^{\max}\left(\mathbf{M}'\right)$. Now, find a *structure*, i.e., an *off-closure* $\mathbf{M}''$ of $\mathbf{M}'$ such that

$$\otimes_{nd}(\mathbf{M}'') \rightarrow \max \tag{42}$$

assuming

$$T_{\min}\left(\mathbf{M}'', \mathcal{W}\right) \leq C_t, \tag{43}$$
$$\otimes_{nd}\left(\mathbf{M}''\right) \geq C_{nd}. \tag{44}$$

The role of (44) is to stop the algorithm from searching for the maximum in (42) when (44), together with (43), cannot be achieved.

After PHASE TWO, we are faced with a traditional time–cost trade-off problem; therefore, in PHASE THREE, we can specify different kinds of objective functions in Problem 3: PHASE THREE/1 and /2.

Problem 3 PHASE THREE: Let $\mathbf{M}$ be a solution to Problem *1* PHASE TWO, i.e., a matrix representation of a given project structure $\mathcal{X} = (S, \prec, \sim)$. Let $C_c, C_t \in \mathbb{R}^+$ be given such that $C_{\min}\left(\mathbf{M}'', \mathcal{W}\right) \leq C_c$ and $T_{\min}\left(\mathbf{M}'', \mathcal{W}\right) \leq C_t$. Further, let $\mathbf{C}_r$ be any nonnegative[7] vector of dimension ρ such that $\mathbf{C_r} \in [\mathbf{TPR}_{\min}, \mathbf{TPR}_{\max}]$.

Problem 3 PHASE THREE**/1)** Find a project schedule $\overrightarrow{w}$ such that

$$\mathbf{t}\left(\overrightarrow{w}\right) \rightarrow \min \tag{45}$$

assuming

$$\mathbf{c}\left(\overrightarrow{w}\right) \leq C_c, \tag{46}$$
$$\mathbf{q}\left(\overrightarrow{w}\right) \geq C_q, \tag{47}$$
$$\mathbf{r}\left(\overrightarrow{w}\right) \leq \mathbf{C_r}. \tag{48}$$

[7] For vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x} < \mathbf{y}$, we mean *componentwise* relations, i.e., $x_i \leq y_i$ and $x_i < y_i$ for $i = 1, \ldots, \rho$.

Problem 3 PHASE THREE/**2)** Find a project schedule $\overrightarrow{w}$ such that

$$\mathbf{c}\left(\overrightarrow{w}\right) \rightarrow \min \tag{49}$$

assuming

$$\mathbf{t}\left(\overrightarrow{w}\right) \leq C_t, \tag{50}$$
$$\mathbf{q}\left(\overrightarrow{w}\right) \geq C_q, \tag{51}$$
$$\mathbf{r}\left(\overrightarrow{w}\right) \leq \mathbf{C_r}. \tag{52}$$

Problem 3 PHASE THREE/**3)** Find a project schedule $\overrightarrow{w}$ such that

$$\mathbf{q}\left(\overrightarrow{w}\right) \rightarrow \max \tag{53}$$

assuming

$$\mathbf{t}\left(\overrightarrow{w}\right) \leq C_t, \tag{54}$$
$$\mathbf{q}\left(\overrightarrow{w}\right) \geq C_q, \tag{55}$$
$$\mathbf{r}\left(\overrightarrow{w}\right) \leq \mathbf{C_r}. \tag{56}$$

Note that the requirements for all the constants in each phase ensure that this phase has at least one solution and that this matrix can be handled in the next phase.

2.2 Modeling flexible project plans

Although the proposed model can be used for both discrete and continuous versions of multimode resource-constrained project scheduling problems, in simulations, we addressed both continuous and discrete versions of MRCPSP. For *practical reasons*, we use a specific, deterministic, multimodal project domain matrix (PDM) to model the resource-constrained hybrid multimode resource allocation problem. The PDM is an n by m matrix (see, e.g., Table 2), where $n = |A|$ is the number of tasks, ρ is the number of resources, $m = n+(3+\rho)\omega$ and $\omega = |W|$ is the number of modes [15]. The **PDM** has five domains: the logic domain (LD), time domain (TD), quality domain (QD), cost domain (CD) and resource domain (RD) (see Table 2). In the initial step, $\mathbf{LD} := \mathbf{P}$, where $\mathbf{P} \in [0, 1]^{n\times n}$ is the score matrix. The diagonal $[\mathbf{PDM}]_{i,i} = [\mathbf{LD}]_{i,i} = [\mathbf{P}]_{i,i}$ represents the task completion scores, and those out of the diagonal $\mathbf{PDM}_{i,j} = \mathbf{LD}_{i,j} = \mathbf{P}_{i,j}$ $(i \neq j)$ represent the task dependency scores between task a_i and task a_j. Empty cells represent the unplanned tasks and their unknown requirements. While specifying constraints, the demands of unplanned tasks are not specified. In the simulation, two contract strategies are compared, the agile strategy, which schedules unplanned task to the next sprint, but the constraints are assumed to be fixed, and the extreme strategy, which involves new tasks with their demands and modifies the constraints considering the unplanned task demands.

TD, QD and **CD** contain of the *corresponding data* of multicompletion modes $\overrightarrow{w_i} \in W$ $(i \leq \omega)$ as columns, (minimum, maximum) values while the resource domain (**RD**) is built up similarly for each resource (the columns are for $r_{i,j}$ for $i \leq \omega$, $j \leq \rho$, see Table 2).

Kosztyáan [15] shows how to handle and resolve cycles in a PDM. Therefore, without loss of generality, we can assume that there is no cycle in the project net. In other words, the LD of the PDM can be rearranged as an upper triangular matrix.

Example 3 Table 2 shows an example of a project plan. Table 2(a) shows the original project plan, where the PDM is already ready to mark new/unplanned tasks. As shown in Table 1,

Table 2 Project Domain Matrix for planned mandatory and supplementary tasks (a) and considering unplanned tasks (b)

PDM	Logic domain					Time domain			Cost domain			Quality domain			Resource domain					
	A	B	C	D	***E***	t_1	t_2	t_3	c_1	c_2	c_3	q_1	q_2	q_3	r_{11}	r_{21}	r_{31}	r_{12}	r_{22}	r_{32}
(a) *Considering only planned tasks*																				
A	0.8	1.0	0.8	0.2		4	5	6	2	4	3	0.8	0.9	0.8	2	3	0	3	0	3
B	0.0	1.0	0.0	0.4		2	3	1	3	3	5	0.8	0.9	0.9	3	5	2	2	2	1
C	0.0	0.0	0.9	0.0		8	6	4	5	5	4	0.7	0.9	0.9	0	1	0	1	0	2
D	0.0	0.0	0.0	0.4		9	9	9	7	7	7	0.9	0.9	0.9	2	2	2	0	0	0
E																				
Constraints						18			19			0.75			10			6		
(b) *Considering unplanned tasks, with fixed and flexible constraints*																				
A	0.8	1.0	0.8	0.2	*0.0*	4	5	6	2	4	3	0.8	0.9	0.8	2	3	0	3	0	3
B	0.0	1.0	0.0	0.4	*0.0*	2	3	1	3	3	5	0.8	0.9	0.9	3	5	2	2	2	1
C	0.0	0.0	0.9	0.0	*0.4*	8	6	4	5	5	4	0.7	0.9	0.9	0	1	0	1	0	2
D	0.0	0.0	0.0	0.4	*0.0*	9	9	9	7	7	7	0.9	0.9	0.9	2	2	2	0	0	0
E	*0.0*	*0.0*	*0.0*	*0.0*	*1.0*	*1*	*2*	*3*	*3*	*3*	*2*	*1.0*	*1.0*	*1.0*	*2*	*2*	*1*	*2*	*2*	*1*
Original constraints						18			19			0.75			10			6		
Modified constraints						21			22			0.75			12			8		

two contract strategies can handle the unplanned tasks. Following application management (AM), the original constraints are kept, and tasks can be completed provided all demands are not greater than the constraints. Extreme project management modifies the constraint to involve the unplanned tasks (see the last two rows in Table 2), whereas if handling unplanned tasks and dependencies are not allowed, then only task B is the mandatory task. Therefore, $\mathrm{TPT}_{\min} = 1$, $\mathrm{TPC}_{\min} = 2$, $\mathbf{TPR}_{\min} = [2, 1]^T$ can be specified without calculating all possible scenarios. According to Eq. (17), $\mathrm{TPQ}_{\max} = 1$, $\mathrm{TPQ}_{\min} = 0.8/(4 \times 0.9) = 0.22$, while if the unplanned *task E* is also considered, than minimal values of the total project cost are $\mathrm{TPC}_{\min} = 2$ and $\mathrm{TPQ}_{\min} = (0.8 + 1.0)/(4 \times 0.9 + 1.0) = 0.39$. The minimal duration $\mathrm{TPT}_{\min} = 1$ and resource demands $\mathbf{TPR}_{\min} = [2, 1]^T$ are not changed. Therefore, if the extra costs/duration/resource demands of the unplanned task can be accepted and the constraint can be modified, it may be beneficial for project scheduling.

2.3 The algorithms

2.3.1 Phase one

We are given the matrix $\mathbf{M}_0 \in \{X, \emptyset, ?\}^{n \times n}$. Suppose that all the "?" symbols in the diagonal are in the first σ rows (columns). The algorithm sequentially changes these symbols to either "X" or "$\emptyset$" in *this* order; such a change is called a **step**. These changes are not final, and the original matrix $\mathbf{M}_0$ is also saved. We look for the optimum similar to a "back-and-forth" method, saving information in the **buffer** B (a set) of possible ways that we might investigate later. After replacing as many elements of $\mathbf{M}$ as we can (satisfying Eqs. (38), (39), and (41)), we go back to the cases in which B has higher scores $\otimes_{diag}$ than $\mathbf{M}$ has. (All the possible variations of $\mathbf{M}_0$ form a binary tree of size 2^σ with root $\mathbf{M}_0$.)

Here, $\mathbf{M}$ denotes the actual matrix *before* the next replacement, so $[\mathbf{M}]_{j,j} = \text{"?"} \iff i \le j \le \sigma$ for some $1 \le i \le \sigma$, and denote by $\mathbf{M}[i, i = Y]$ the matrix *after* replacing $\mathbf{M}_{i,i}$ with Y, where $Y \in \{X, \emptyset\}$.

Before replacing $[\mathbf{M}]_{i,i}$, we save the *other* possibility, which we do not follow in the present step, in B. The elements of B are of the form

$$\mathbf{b} = \left(i,\ [\mathbf{M}]_{1..n,1..n}\ ,\ Y,\ \otimes_{diag}^{\max}\left(\mathbf{M}[i, i = Y]\right)\right) \tag{57}$$

$$= \left(i, \overrightarrow{m}, Y, \otimes_{\mathbf{b}}\right), \tag{58}$$

where i denotes the element in the diagonal of $\mathbf{M}$ we are replacing, $\overrightarrow{m} = \mathbf{M}_{1..n,1..n}$ is the actual content of the diagonal of $\mathbf{M}$ (specifically, $\mathbf{M}_{i,i} = \text{"?"}$), $Y \in \{X, \emptyset\}$ and $\otimes_b = \otimes_{diag}^{\max}(\mathbf{M}[i, i = Y])$ is the"ideal" score we may achieve by replacing $\mathbf{M}_{i,i}$ with Y.[8] B may contain several elements with the same $\otimes_{\mathbf{b}}$ value, but at this moment, we do not know which value can be realized later, i.e., satisfy (38)–(41)

Remark 2 B contains only $\mathbf{M}[i, i = Y]$ extensions that have not yet been investigated but fulfill the bounds of (38)–(41). More precisely, for their *extension*,

$$C_{\min}(\mathbf{M}[i, i = Y], \mathcal{W}) \le C_c, \tag{59}$$

$$T_{\min}(\mathbf{M}[i, i = Y], \mathcal{W}) \le C_t, \tag{60}$$

$$Q_{\max}(\mathbf{M}[i, i = Y], \mathcal{W}) \ge C_q, \tag{61}$$

$$\otimes_{diag}^{\max}(\mathbf{M}[i, i = Y]) \ge C_{diag}. \tag{62}$$

[8] $\otimes_{\mathbf{b}}$ in (58) denotes a nonnegative real number, and $\otimes_{diag}^{\max}$ was defined in (4).

Remark 3 Before starting step i for $1 < i$, at the end of step $i-1$, we have inserted Y into the $i-1$-th entry of $\mathbf{M}$. Since we are now extending this configuration, B does *not* contain the corresponding record $\mathbf{b} = \left(i-1, \overrightarrow{m}, Y, \otimes_{\mathbf{b}}\right)$.

The algorithm starts a **new cycle** whenever it goes back to an element of B and starts to replace "?" from the $i = i_0 + 1$-th entry of the diagonal. Problem 1 may have a solution if, for at least in one cycle, we are able to step i to σ (satisfying (38)–(41)). Of course, we store all the in-closures $\mathbf{M}'$ of $\mathbf{M}_0$ that were found by the algorithm and may be optimal solutions to Problem 1. If B contains an element $\mathbf{b}$ with a higher score than $\mathbf{M}'$ has (i.e., $\otimes_{\mathbf{b}} > \otimes_{diag}\left(\mathbf{M}'\right)$), then we start a new cycle from $\mathbf{b}$. During this cycle, i cannot be increased to σ, $\otimes_b = \otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = Y\right]\right)$ may fall below $\otimes_{diag}\left(\mathbf{M}\right)$, or we might obtain a solution better than $\mathbf{M}'$.

START Let $i_0 := 0$, $i = 1$, $B := \emptyset$.

GENERAL STEP $(1 \le i \le \sigma)$, $\mathbf{M}$ is the actual matrix. Let

$$\mathbf{b}_X^i := \left(i,\ \mathbf{M}_{1..n,1..n},\ X,\ \otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = X\right]\right)\right), \tag{63}$$

$$\mathbf{b}_\emptyset^i := \left(i,\ \mathbf{M}_{1..n,1..n},\ \emptyset,\ \otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = \emptyset\right]\right)\right). \tag{64}$$

Case (i) Neither $\mathbf{b}_\emptyset^i$ nor $\mathbf{b}_X^i$ fulfills (59)–(62) and $B = \emptyset$. Then, STOP since Problem 1 has no solution.

Case (ii) Neither $\mathbf{b}_\emptyset^i$ nor $\mathbf{b}_X^i$ fulfills (59)–(62) but $B \neq \emptyset$. Recall that in Cases (i) and (ii), B may contain elements of type $\mathbf{b} = \left(j, \overrightarrow{m}, Y, \otimes_{\mathbf{b}}\right)$ only if $j < i$ by Note 2 and (57), (58), and (63), (64). In Case ii), choose any element $\mathbf{b} \in B$ such that $\otimes_{\mathbf{b}}$ is maximal (in B). Then, reset the diagonal of $\mathbf{M}$ according to $\overrightarrow{m}$, set $i := j$, delete $\mathbf{b}$ from B, and proceed to the General Step.

Case (iii) Exactly *one* of $\mathbf{b}_X^i$ or $\mathbf{b}_\emptyset^i$ fulfills (59)–(62), say, $\mathbf{b}_Y^i$. Let $\mathbf{M}_{i,i} :=$ "Y" and go to Step Increasing i.

Case (iv) Both $\mathbf{b}_X^i$ and $\mathbf{b}_\emptyset^i$ fulfill (59)–(62).

If $\otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = X\right]\right) \le \otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = \emptyset\right]\right)$, then let $\mathbf{M}_{i,i} :=$ "$\emptyset$", $B := B \cup \left\{\mathbf{b}_X^i\right\}$, and go to Step Increasing i.

If $\otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = X\right]\right) > \otimes_{diag}^{\max}\left(\mathbf{M}\left[i, i = \emptyset\right]\right)$, then let $\mathbf{M}_{i,i} :=$ "X", $B := B \cup \left\{\mathbf{b}_\emptyset^i\right\}$, and go to Step Increasing i.

STEP INCREASING (i) If $i < \sigma$, then let $i := i + 1$, and go to the General Step. In the case $i = \sigma$, go to the Check Step.

CHECK STEP $(i = \sigma)$ First, save the recent $\mathbf{M}$ with its $\otimes_{diag}\left(\mathbf{M}\right)$. If B contains an element $\mathbf{b} = \left(j, \overrightarrow{m}, Y, \otimes_{\mathbf{b}}\right)$ such that

$$\otimes_{\mathbf{b}} > \otimes_{diag}\left(\mathbf{M}\right), \tag{65}$$

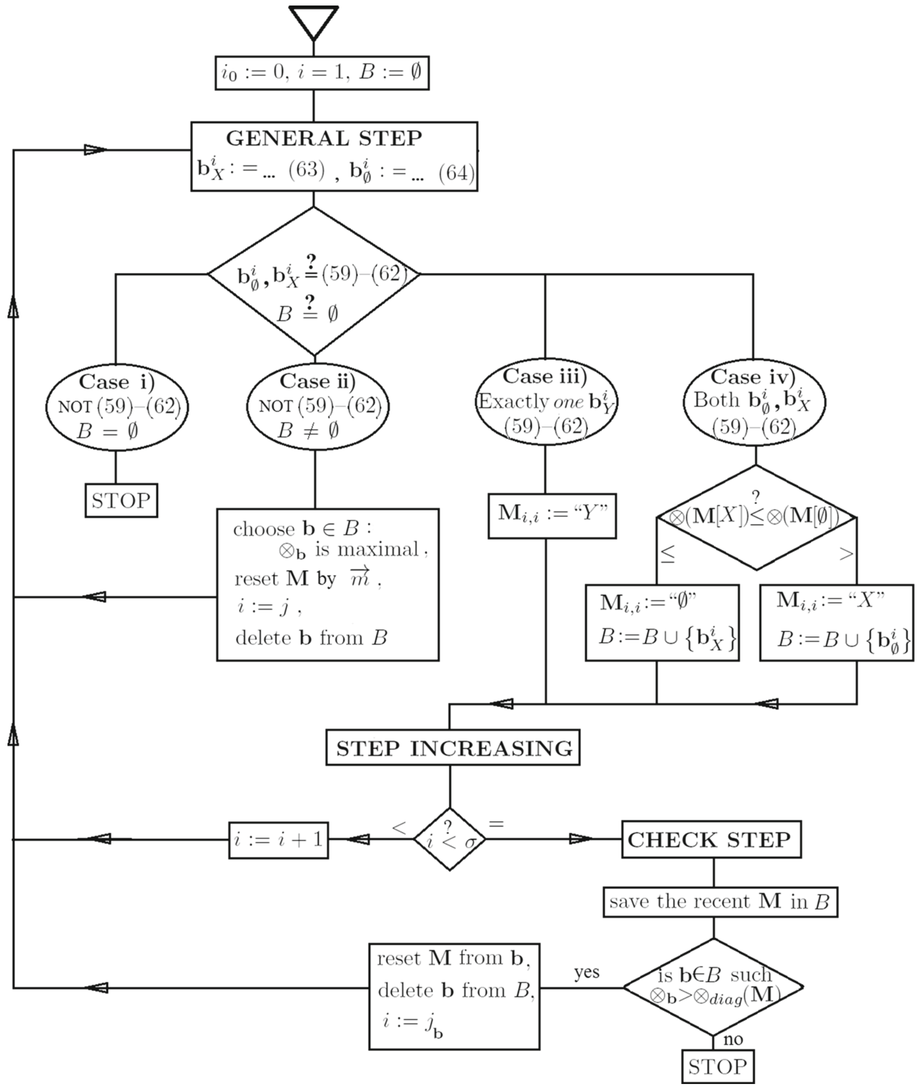
then go back to $\mathbf{b}$ and start a new cycle, i.e., reset the diagonal of $\mathbf{M}$ according to $\overrightarrow{m}$, set $i := j$, delete $\mathbf{b}$ from B, and go to the General Step.

END of the Algorithm.

(The algorithm is visualized in Fig. 2.)

Remark 4 If we want to find *all* optimal solutions to Problem 1, then replace (65) with

$$\otimes_{\mathbf{b}} \ge \otimes_{diag}\left(\mathbf{M}\right). \tag{66}$$



$i_0 := 0,\ i = 1,\ B := \emptyset$
GENERAL STEP
$\mathbf{b}_X^i := \dots$ (63), $\mathbf{b}_\emptyset^i := \dots$ (64)
$\mathbf{b}_\emptyset^i, \mathbf{b}_X^i \overset{?}{=} (59)$–$(62)$
$B \overset{?}{=} \emptyset$
Case i) NOT (59)–(62) $B = \emptyset$
STOP
Case ii) NOT (59)–(62) $B \neq \emptyset$
choose $\mathbf{b} \in B$: $\otimes_{\mathbf{b}}$ is maximal, reset $\mathbf{M}$ by $\overrightarrow{m}$, $i := j$, delete $\mathbf{b}$ from B
Case iii) Exactly *one* $\mathbf{b}_Y^i$ (59)–(62)
$\mathbf{M}_{i,i} :=$ "Y"
Case iv) Both $\mathbf{b}_\emptyset^i, \mathbf{b}_X^i$ (59)–(62)
$\otimes(\mathbf{M}[X]) \overset{?}{\leq} \otimes(\mathbf{M}[\emptyset])$
$\leq$
$>$
$\mathbf{M}_{i,i} :=$ "$\emptyset$" $B := B \cup \{\mathbf{b}_X^i\}$
$\mathbf{M}_{i,i} :=$ "X" $B := B \cup \{\mathbf{b}_\emptyset^i\}$
STEP INCREASING
$i := i + 1$
$<$
$i \overset{?}{<} \sigma$
$=$
CHECK STEP
save the recent $\mathbf{M}$ in B
is $\mathbf{b} \in B$ such $\otimes_{\mathbf{b}} > \otimes_{diag}(\mathbf{M})$
yes
no
reset $\mathbf{M}$ from $\mathbf{b}$, delete $\mathbf{b}$ from B, $i := j_{\mathbf{b}}$
STOP

Fig. 2 Flowchart of the proposed algorithm

Of course, we must first save (in an *output buffer*) the solution(s) we have found so far. Then, we pick the *next* element $\mathbf{b} \in B$ in the buffer, reset the diagonal of $\mathbf{M}$ according to $\overrightarrow{m}$, set $i := j$, delete $\mathbf{b}$ from B, and go to the General Step.
We call this algorithm the **Hybrid Project Ranking** Algorithm.

Theorem 3 *The saved matrices (in the Check Step) of the above algorithm are exactly the optimal solutions to Problem* 1. *Specifically, there are no saved matrices if and only if* (38)–(40) *in Problem* 1 *has no solution at all.*

Proof In each step, the algorithm chooses the best of (at most) two possibilities but buffers the other for further investigation. The value $\otimes_{diag}(\mathbf{M})$ for each $\mathbf{M}$ is a sharp *upper bound* for

Table 3 Results of PHASE ONE for Example 3

(a) Considering only planned tasks

PDM	Logic Domain					Time Domain			Cost Domain			Quality Domain			Resource Domain					
	A	**B**	**C**	■	***E***	t_1	t_2	t_3	c_1	c_2	c_3	q_1	q_2	q_3	r_{11}	r_{21}	r_{31}	r_{12}	r_{22}	r_{32}
A	X	1.0	0.8	■		4	5	6	2	4	3	0.8	0.9	0.8	2	3	0	3	0	3
B	0.0	X	0.0	■		2	3	1	3	3	5	0.8	0.9	0.9	3	5	2	2	2	1
C	0.0	0.0	X	■		8	6	4	5	5	4	0.7	0.9	0.9	0	1	0	1	0	2
■	■	■	■	∅	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
E				■																
					Constraints:	18			19			0.75			10			6		

(b) Considering unplanned tasks, with fixed and flexible constraints

PDM	Logic Domain					Time Domain			Cost Domain			Quality Domain			Resource Domain					
	A	**B**	**C**	■	***E***	t_1	t_2	t_3	c_1	c_2	c_3	q_1	q_2	q_3	r_{11}	r_{21}	r_{31}	r_{12}	r_{22}	r_{32}
A	X	1.0	0.8	■	*0.0*	4	5	6	2	4	3	0.8	0.9	0.8	2	3	0	3	0	3
B	0.0	X	0.0	■	*0.0*	2	3	1	3	3	5	0.8	0.9	0.9	3	5	2	2	2	1
C	0.0	0.0	X	■	*0.4*	8	6	4	5	5	4	0.7	0.9	0.9	0	1	0	1	0	2
■	■	■	■	∅	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
E	*0.0*	*0.0*	*0.0*	■	*X*	*1*	*2*	*3*	*3*	*3*	*2*	*1.0*	*1.0*	*1.0*	*2*	*2*	*1*	*2*	*2*	*1*
					Original constraints:	18			19			0.75			10			6		
					Modified constraints:	21			22			0.75			12			8		

further continuation of **M**. Therefore, all the buffered possibilities with smaller $\otimes$ than those of the finished (and saved) matrices (in the Check Step) could be deleted from the buffer. Since the algorithm checks *each* remaining element of the buffer (see the Check Step), at the end, we must obtain each optimal solution. □

The result of PHASE ONE is to find [depending on the meaning of the completion scores and the aggregation functions (see Definition 4)] the most desired or most probable project scenario. Table 3 shows the results of PHASE ONE of Example 3.

Example 4 At the end of the result of PHASE ONE, only two options are allowed: tasks will be included in ("X" see on Table 3) or (because of the low priority) will be excluded from (see "∅" on Table 3) the project plan. Table 3 shows that in the case of excluding a task from a project plan, all of its demands, completion modes and dependencies are also excluded from the project plan. Similar to the *original project plan*, minimal and maximal values of demands can be specified without calculating all possible project structures. $\mathrm{TPT}_{\min} = max(4+1, 1) = 5$ if only planned tasks are considered, and $\mathrm{TPT}_{\min} = max(4+1, 1, 1) = 5$ if both planned and unplanned tasks are considered. $\mathrm{TPC}_{\min} = 2 + 3 + 4 = 9$ if only planned tasks are considered, and $\mathrm{TPC}_{\min} = 2 + 3 + 4 + 2 = 11$ if both planned and unplanned are considered. Because task D is excluded, if only planned tasks are considered: $\mathrm{TPQ}_{\max} = 3\times0.9/4\times0.9 = 0.75$, while if both planned and unplanned tasks are considered, $\mathrm{TPQ}_{\max} = (3 \times 0.9 + 1)/(4 \times 0.9 = 0.75 + 1) = 0.80$, which shows that if unplanned tasks can be completed, the total project quality can increase.

The goal of PHASE TWO is to find the most desired/most probable project structure within a project scenario. However, the project structure always depends on the result of PHASE ONE. Therefore, the "highest" scoring project structure can be interpreted only within a specified project scenario.

2.3.2 PHASE TWO

In PHASE TWO, the goal is to find the most probable or most desired project structure *within* the specified project scenario. The algorithm and most of the notation for PHASE TWO are

Table 4 Results of PHASE TWO for Example 4

(a) Considering only planned tasks

PDM	Logic Domain					Time Domain			Cost Domain			Quality Domain			Resource Domain					
	A	**B**	**C**	■	***E***	t_1	t_2	t_3	c_1	c_2	c_3	q_1	q_2	q_3	r_{11}	r_{21}	r_{31}	r_{12}	r_{22}	r_{32}
A	X	X	X	■		4	5	6	2	4	3	0.8	0.9	0.8	2	3	0	3	0	3
B	$\emptyset$	X	$\emptyset$	■		2	3	1	3	3	5	0.8	0.9	0.9	3	5	2	2	2	1
C	$\emptyset$	$\emptyset$	X	■		8	6	4	5	5	4	0.7	0.9	0.9	0	1	0	1	0	2
■	■	■	■	$\emptyset$	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
E				■																
Constraints:						18			19			0.75			10			6		

(b) Considering unplanned tasks, with fixed and flexible constraints

PDM	Logic Domain					Time Domain			Cost Domain			Quality Domain			Resource Domain					
	A	**B**	**C**	■	***E***	t_1	t_2	t_3	c_1	c_2	c_3	q_1	q_2	q_3	r_{11}	r_{21}	r_{31}	r_{12}	r_{22}	r_{32}
A	X	X	X	■	$\emptyset$	4	5	6	2	4	3	0.8	0.9	0.8	2	3	0	3	0	3
B	$\emptyset$	X	$\emptyset$	■	$\emptyset$	2	3	1	3	3	5	0.8	0.9	0.9	3	5	2	2	2	1
C	$\emptyset$	$\emptyset$	X	■	$\emptyset$	8	6	4	5	5	4	0.7	0.9	0.9	0	1	0	1	0	2
■	■	■	■	$\emptyset$	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
E	$\emptyset$	$\emptyset$	$\emptyset$	■	X	*1*	*2*	*3*	*3*	*3*	*2*	*1.0*	*1.0*	*1.0*	*2*	*2*	*1*	*2*	*2*	*1*
Original constraints:						18			19			0.75			10			6		
Modified constraints:						21			22			0.75			12			8		

the same as in PHASE ONE. We have to determine the "?" symbols outside of the diagonal of **M** in a fixed (but arbitrary) order. *Before* each replacement, we save the other possibility in a buffer, similar to (57), checking the conditions corresponding to (43) and (44), such as (59)–(62), which correspond to (38)–(40). In each step, we have to refresh $T_{\min}(\mathbf{M}, \mathcal{W})$ and $\otimes_{nd}(\mathbf{M})$. The properties of the PHASE TWO algorithm can be proved along the lines of Theorem 3 and Sect. 2.3.4. That is, the algorithm in PHASE 2 finds all optimal solutions since it saves and investigates all real candidates in the buffer. The complexity is investigated in Sect. 2.3.4.

Example 5 Following Examples 3 and 4, Table 4 shows the results of PHASE TWO

Minimal and maximal values of demands can also be specified without calculating all possible project plans. $\text{TPT}_{\min} = 4 + \max(1, 4) = 8$ if only planned tasks are considered, and $\text{TPT}_{\min} = \max(4 + \max(4, 1), 1) = 8$. Since at PHASE TWO tasks are already not excluded from the project plan, minimal/maximal values of the cost/quality/resource are not modified.

2.3.3 PHASE THREE

After PHASE TWO we obtain a project structure, which represents a traditional multimode resource-constrained project scheduling problem (MRCPSP). In this phase, when considerations of discrete completion modes are specified, we use Creemers [4]'s algorithm to specify the solution of the MRCPSP, and the continuous version of the problem is solved by Monghasemi et al. [18]'s algorithm. If there is no feasible solution to the given algorithm, we should go back to PHASE TWO and select the next project structure from the buffer. The optimal output matrix is a kind of domain mapping matrix (DMM), see Table 5, where flexible dependency and uncertain task completion are excluded or included. Therefore, the logic domain (LD) of the output matrix is a dependency structure matrix (DSM), where "X" represents included tasks on the diagonal, and included dependencies are on the out-diagonals of the LD. The optimal output matrix (furthermore, the project schedule matrix (PSM)) contains one vector of time/cost demands (TD,CD) and one vector of quality parameters (QD). The

Table 5 Results of PHASE THREE for Example 5

	Logic domain				TD	CD	QD	RD	
PSM	**A**	**B**	**C**	***E***	t	c	q	r_1	r_2
(a) *With all tasks*									
A	X	X	X		5	3	0.9	3	0
B	∅	X	∅		1	5	0.9	2	1
C	∅	∅	X		4	4	0.9	0	2
E									
Constraints					18	19	0.75	10	6
(b) *With fixed constraints*									
A	X	X	X	∅	4	2	0.8	2	3
B	∅	X	∅	∅	1	5	0.9	2	1
C	∅	∅	X	∅	4	4	0.9	0	2
E	∅	∅	∅	*X*	*1*	*3*	*1.0*	*1*	*1*
Original constraints					18	19	0.75	10	6
Modified constraints					21	22	0.75	12	8

PSM also contains an n by r submatrix of resource demands (RD) and a vector of scheduled start times (SST).

Example 6 Following Example 3 and Examples 4–5, Table 5 shows the results of PHASE THREE.

Table 5 shows that because of the quality constraint, the lowest duration cannot be selected if unplanned tasks are not considered (see Table 5a). In this case, TPT $= 5 + \max(1, 4) = 9$, TPC $= 3 + 5 + 4 = 12$, TPQ $= 3 \times 0.9/4 \times 0.9 = 0.75$, and $\mathbf{TPR} = [3, 2]^T$. At the same time, if unplanned tasks are considered, than the lowest duration of tasks can be selected. In this case, TPT is not increased. TPT $= \max(4+\max(1, 4), 1) = 8$. TPC $= 2+5+4+3 = 14$ if every task is completed as early as possible $\mathbf{TPR} = [3, 4]^T$, while TPQ $= (0.8 + 0.9 + 0.9 + 1.0)/(4 \times 0.9 + 1.0) = 0.78$. Example 5 shows that unplanned tasks do not necessarily increase the time demands, and because the growth of relative quality occurs, customers can be more satisfied.

2.3.4 Algorithmic complexity

Briefly, in PHASE ONE, $\otimes_{diag}(\mathbf{M})$ is calculated and $T_{\min}(\mathbf{M}, \mathcal{W})$ is refreshed (see Definition 14), so each cycle is at most *quasilinear*, but we have no bounds on the *total* size of the buffer. Similarly, in PHASE TWO, we have to calculate $T_{\min}(\mathbf{M}, \mathcal{W})$ and $\otimes_{nd}(\mathbf{M})$, which implies a similar upper bound on time as in PHASE ONE. Perhaps some extreme counterexamples may cause exponential running time, but practical runs (see Sect. 3) provided quadratic runs, both in PHASE ONE and in PHASE TWO.

In more detail, in general, the nth-largest value can be determined within $O(n \log n)$ computation time (e.g., [22]); however, our decision tree is a special binary heap where a quasilinear search algorithm can be specified. In Sect. 2.3, we saw that the best project structures can be found within $O(s + d)$, where s is the number of supplementary tasks to be completed (PHASE ONE), and d is the number of flexible task dependencies (PHASE TWO). If there are no supplementary tasks to be completed, the number of possible project structures depends only on the number of flexible dependencies. If there are d flexible dependencies, then there are 2^d possible project plans (PHASE TWO). In case of acyclic project networks, the maximal number of flexible dependencies is $n(n-1)/2$, and in this case, the number of possible project structures is $2^{n(n-1)/2}$ (PHASE TWO). If there are s supplementary tasks, then 2^s project scenarios can be specified. In a special case, if $s = n$ and $d = n(n-1)/2$, then there are 2^n project scenarios (PHASE ONE) and $\sum_{j:=0}^{n} \binom{n}{j} 2^{j(j-1)/2}$ project plans (PHASE TWO).

The computational demand of the proposed hybrid algorithm with respect to the fulfilled PEM=LD of a specified deterministic PEM is $O(d) = O(n(n-1)/2) \approx O(n^2)$ in the case of a fulfilled upper triangular matrix (NDSM) with only mandatory tasks, and the run time $O(d+s) = O(n+n(n-1)/2) \approx O(n^2)$ is similar when regarding a fulfilled upper triangular matrix with all supplementary tasks (PHASE ONE). For example, when $n = 50$, completely filled upper triangular NDSM and PEM specify $5, 78 \cdot 10^{368}$ possible project plans, while our algorithm finds a project structure within $O(50^2)$ steps (PHASE TWO). However, at the end of the project selection process, where we have a feasible project structure, the proposed algorithm has to call a method for the multimode resource-constrained project scheduling problem. In this case, the complexity of the (MRCPSP) method and the complexity of project selection are multiplied. As MRCPSP are NP-hard problems, the hybrid versions of these problems are also NP-hard problems.

3 Method of testing project management approaches

The main goal of this section is to test the proposed algorithm and compare it to several state-of-the art algorithms. At the same time, since applied methods can imitate decision makers, different kinds of project management and scheduling approaches can be compared. In this section, the proposed algorithm is tested on the modified MMLIB [28] project database and on generated projects, which were generated by ProGEN [13]. Flexible and traditional approaches are implemented by algorithms from the literature review. Both the continuous version of time–quality–cost trade-off methods and the discrete version of multimode resource-constrained project scheduling problems are compared. The implemented algorithms imitate decision makers (i.e., project managers), who can select a technology (completion mode) from the set of multimodes and/or who can reorganize the project. The main question was to investigate scenarios in which flexible approaches produce more feasible projects.

3.1 Data sources and selected initial project plans

The aim of selecting and generating initial project plans is to meet as much as possible the expectations for flexible project management approaches, especially the features of agile, extreme and hybrid projects.

1. Vanhoucke [30] showed that flexible projects usually contain more parallel tasks; therefore, according to Vanhoucke [30], the number of parallel tasks is greater than the number of serial tasks.[9] Nevertheless, one of the most popular agile approaches, the KANBAN method, limits the number of parallel WIP tasks, and allows only 3–5 WIPs. Therefore, in the simulation, the number of WIPs must be lower than 5.

[9] Following the simulations of Vanhoucke [30], $i_2 = (m-1)/(n-1) \in [0.2, 0.3]$, where m is the number stages in a topological ordered network and n is the number of tasks. $i_2 = 1$ if all tasks are completed in a serial manner, and $i_2 = 0$ if all tasks are completed in parallel.

2. Projects are usually separated into smaller autonomous subprojects (sprints) (see, e.g., [8]), that should completed within 2–5 weeks; therefore, the number of tasks is limited and should not be greater than 30.
3. Projects should contain at least two types of renewable resources (e.g., in software projects, a programmer and a tester).
4. Projects should contain at least two completion modes to apply continuous trade-off methods and at least three for testing the discrete version of MRCPSP.

Two kinds of datasets are specified. The logic networks of **Dataset A** came from the standard project database [30]. Project plans of **Dataset B** are generated by the standard project generator software ProGen [13].

Use of logic network

Since all applied databases contained logic plans, the original project plans were used. Moreover, 5 project plans (j3031_7, j3035_10, j3042_1, j3031_5, j3064_10) satisfied the above mentioned criteria from Dataset A, and 5 additional project plans were generated by ProGen. Unfortunately, all databases contained neither flexible dependencies/supplementary tasks nor unplanned tasks; therefore, on one hand, score values are attached to the customized rate of tasks and dependencies. The rate of flexible dependencies and supplementary tasks were between 10 and 40% ($ff \in \{0.1, 0.2, .., 0.4\}$) (see an example in Fig. 3a). The proposed algorithm can exclude tasks and dependencies, and the original network can be changed; however, at least the mandatory tasks and fixed dependencies must be completed (see Fig. 3a). On the other hand, the original database does not contain unplanned tasks and dependencies. Therefore, the ratio of unplanned tasks and dependencies are specified between 0 and 40% ($uf \in 0.0, 0.10, .., 0.40$) (see an example in Fig. 3c). Unplanned tasks and dependencies can contain mandatory tasks and fixed dependencies (see Fig. 3d). This modified dataset is capable of checking flexible management approaches. The modified databases contain 2 datasets, which contain 5–5 original project plans. These project plans contain a flexible project plan with 4 kinds of flexible rates. Moreover, 5 kinds of ratios of unplanned tasks and dependencies can be attached to the original project plan. Therefore, $2 \times 5 \times 4 \times 5 = 200$ logic plans are investigated.

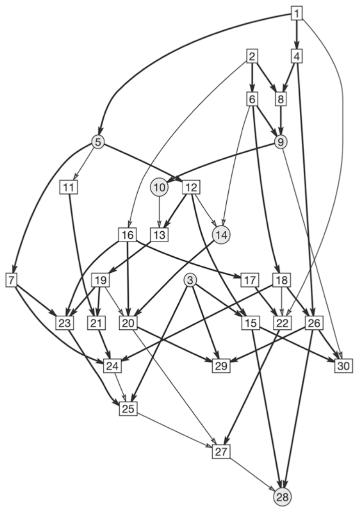
Use of demands

Since quality parameters are missing from every known project database, we have to calculate according to the quality–cost trade-off functions. Only the relative quality functions were calculated. In the case of the continuous version, a higher budget that can allow higher quality is considered.

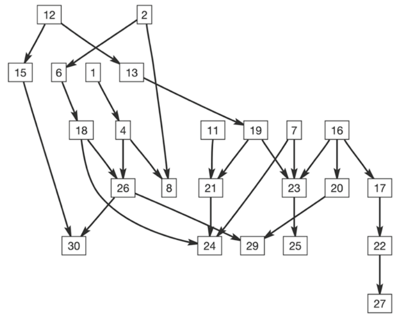
Moreover, usually an inverse proportionality can be assumed between the cost and quality. The (relative) quality parameters were simulated to the i-th task. In the case of the continuous version of trade-off methods, only two completion modes are specified for minimal/maximal values of durations/cost and resource demands. However, in the discrete version of MCRPSP, a randomized third completion mode is generated for time, cost, quality and resources; therefore, the trade-off assumptions between time–cost, time-resources and cost–quality are not satisfied in this case. Since both the discrete and the continuous version of the problem are examined, $200 \times 2 = 400$ project plans are specified.

Calculating constraints

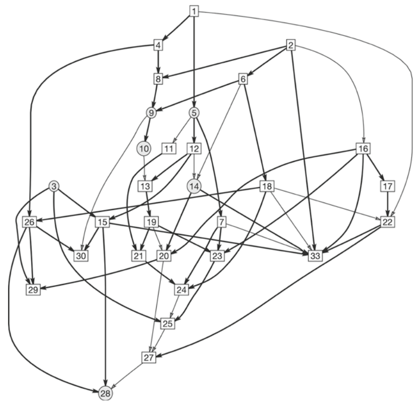
All of the traditional approaches, agile project management approaches, and application management approaches specify the budget, deadlines and resource constraints based on only



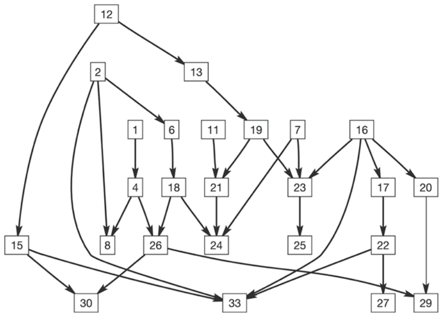
(a) Flexible dependencies and supplementary tasks are considered.



(b) Only mandatory tasks and fixed dependencies are considered.



(c) All planned and unplanned tasks and dependencies are considered.



(d) Unplanned mandatory tasks and dependencies are considered.

Fig. 3 j3530_10 networks form MMLIB [28]

the planned tasks and dependencies. However, the extreme project management approach allows the project managers to confirm the extra cost and duration of unplanned tasks, and the constraints can be modified. Therefore, in the simulation, two kinds of contract strategy are compared. In the first strategy, the constraints are specified by considering only planned tasks. The agile project management approach does not allow new, unplanned tasks and dependencies. Therefore, these new tasks will be completed in a next sprint. However, the application management approach can confirm new tasks for which the original budget constraint allows their completion.

The other contract strategy is the flexible contract strategy, where the new tasks can increase the budget and can modify the deadline.

In the simulation, the five constraints (C_t,C_c,C_q,C_s,$\mathbf{C_r}$) were between the possible minimal and maximal values of demands. Then, Eq. (67) calculates:

$$C_x = C_x\% \left(\mathrm{TPX}_{\max} - \mathrm{TPX}_{\min}\right) + \mathrm{TPX}_{\min} \tag{67}$$

where C_x represents the constraints (i.e., $C_x \in \{C_t, C_c, C_q, C_s, \mathbf{C_r}\}$) and TPX represents the total project time/cost/quality/score/resources, respectively. $C_x\% \in \{0.7, 0.9\}$, if $C_x\% \in \{C_t, C_c, C_{ri}\}, i = 1, .., \rho$ and $C_x\% \in \{1 - 0.7, 1 - 0.9\} = \{0.1, 0.3\}$, if $C_x\% \in \{C_q, C_s\}$.

Two kinds of contract strategies and 2^5 kinds of constraints on 400 project plans specified $400 \times 2 \times 2^5 = 25{,}600$ problems.

Target function

Though different kinds of target functions can be considered, we only want to specify the most desired project plan (maximal total project scores), where the completion time of the project is minimal.

Applied agents

Applied algorithms imitate decision makers. The traditional project management approach does not allow flexible dependencies and supplementary tasks; therefore, all flexible dependencies are considered as fixed and all supplementary tasks are considered as mandatory tasks. In the case of discrete completion modes, we obtain an MRCPSP problem, which was solved by Creemers [4]'s algorithm, while the continuous version of the problem was solved by Monghasemi et al. [18]'s algorithm. These algorithms imitate the decision maker who follows a traditional project management approach. Furthermore, these algorithms are called Traditional Project Management agents (TPMa).

To implement Agile Project Management agents (APMa), we used Kosztyán [15]'s Exact Project Ranking (EPR) algorithm. This algorithm does not consider different kinds of completion modes. Therefore, only minimal (so called normal) time demands and their cost/quality/resource demands are considered. Furthermore, this algorithm is called the Agile Project Management agent (APMa).

Despite that the original version of EPR does not allow the unplanned tasks and dependencies, the proposed extension of the PDM matrix and the extension of the set of activities can capably solve the problem. If the constraints are calculated based only on the planned tasks, and there is no chance to modify the constraints, the algorithm imitates the application management approach; therefore, the modified version of EPR is called an Application Management agent (AMa). However, if the extra demands can modify the constraints, than the modified EPR can imitate the decisions of the extreme project managers, and this algorithm, henceforward, is called the eXtreme Project Management agent (XPMa).

The proposed algorithm handles both the supplementary and the unplanned tasks and dependencies, and in addition, it can handle the multimode of completion modes. The decision maker can combine the multimode and the project screening techniques to reorganize the project. This algorithm can imitate the decision maker who follows the hybrid project management approach. Therefore this algorithm is, furthermore, called the Hybrid Project Management agent (HPMa).

Feasibility and the scheduling performance

On one hand, one of the main aspects of comparing the examined agents is to specify feasible project plans, namely, which agent and which project management approach can solve more problems.

On the other hand, it is also important to determine which feasible method can save more time, cost and resources. Therefore, the scheduling performances are specified as follows:

$$TPT\% = C_t/TPT - 1 \tag{68}$$

$$TPC\% = C_c/TPC - 1 \tag{69}$$

$$\overline{TPR}\% = Mean(C_{r_i}/TPR_i - 1), i = 1, .., \rho \tag{70}$$

$$TPQ\% = TPQ/C_q - 1 \tag{71}$$

$$TPS\% = TPS/C_s - 1 \tag{72}$$

All scheduling performances are between $[0, \infty]$, specifically, 0 if the total project time/cost/quality/score/resource and the given constraints are equal. The scheduling performances are 1 if half of the budget, duration, resource demands, etc. can be saved. A greater value can produce higher scheduling performance.

The verification of optimality is a considerably hard problem because in this case, we should specify all the possible project structures, and we should find an optimal solution in a fixed project structure. However, because of the problem complexity (see Sect. 2.3.4), this kind of testing works only in small problems. Therefore, we tested the proposed algorithm only in small projects, where the logic domain was 5×5.

4 Results

If unplanned tasks are not allowed, three kinds of algorithms can be compared, both for the continuous version of the time–quality–cost trade-off problem (CTQCTP) and the discrete version of the multimode resource-constrained project scheduling problem (MRCPSP). TPMa represents the traditional project management approach, where every (not only mandatory, but supplementary) task and all (not only fixed, but flexible) tasks are included in the project; therefore, we obtain either the traditional (resource-constrained) CTQCTP or an MRCPSP problem. To imitate CTQCTP, we used Creemers [4]'s algorithm, while in the case of MRCPSP, Monghasemi et al. [18]'s is implemented.

4.1 Results of feasibility test and verification optimality

Given the large number of project scenarios and project structures in the cases of flexible projects (see Sect. 2.3.4) and the NP-hardness of MRCPSP-s, the feasibility is first checked. The question was to determine which method can produce a more feasible project.

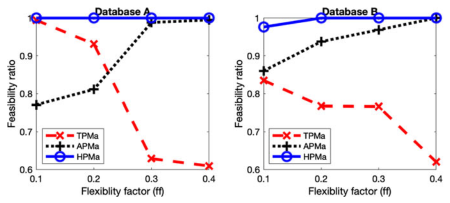
Considering all databases and all the continuous and discrete problems, the *feasibility rate* was 72.03%. Despite that the agile project management approach (implemented by Kosztyán [15]) cannot handle multimodes and trade-offs, the *feasibility rate* was 78.64%.

More feasible projects could be specified if the flexibility factor was higher (see Fig. 4).

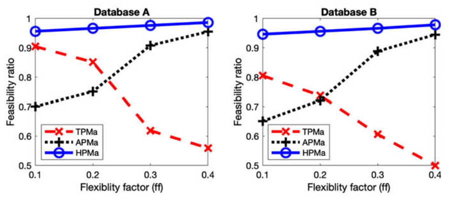
However, the proposed HPMa can produce projects that are 98.83% flexible.

Figure 4 shows that in the case of more flexible projects, the agile project management approach produces more feasible projects, while the traditional project management approaches perform better if the flexibility ratio is lower. Because the characteristics of project management agents are not different when considering dataset A and dataset B, we combined these datasets and show results on the joined dataset.

Verification of optimality of the proposed method was performed in two steps. First, without using resource constraints, flexible task dependencies, and supplementary tasks, the optimal solution of a CTQTCP problem can be specified by Creemers [4]'s algorithm. In this case, we used the original project plans where there were no supplementary tasks and



(a) For hybrid CTQCTP



(b) For hybrid MRCPSP

Fig. 4 Comparing feasibility. HPMa represents the proposed algorithm, APMa represents Kosztyán [15]'s expert project ranking algorithm, TPMa follows Creemers [4]'s algorithm for trade-off problems and Monghasemi et al. [18]'s algorithm for solving MRCPSP

flexible dependencies. Since phase three of the proposed method also applies Creemers [4]'s algorithm, the original algorithm and the proposed algorithm produced the same results. Similarly, if trade-offs are not considered, Kosztyán [15]'s algorithm can specify the optimal solution. In this case, the proposed and Kosztyán [15]'s algorithm produced the same results. After the first step, we specified random 5×5 logic domains, which is quite small to specify all possible project scenarios and all project structures. According to the complexity of the problem, the number of possible project plans is $\sum_{j:=0}^{5} \binom{5}{j} 2^{j(j-1)/2} = 1450$ (see Sect. 2.3.4). For all possible project plans without resource constraints, Kosztyán [4]'s and the proposed algorithm are compared, and both algorithms could find the optimal solution. However, this mode of verification cannot be used for larger matrices because for a 6×6 logic domain, the possible project plan is 979,841.

4.2 Comparing scheduling performance

Both XPMa and AMa are based on the modification of Kosztyán [15]'s algorithm. Figure 5 shows that HPMa produces the most feasible project. At the same time, the ratio of feasibility can be increased if the constraints are not fixed as in AM but are flexible as in XPM.

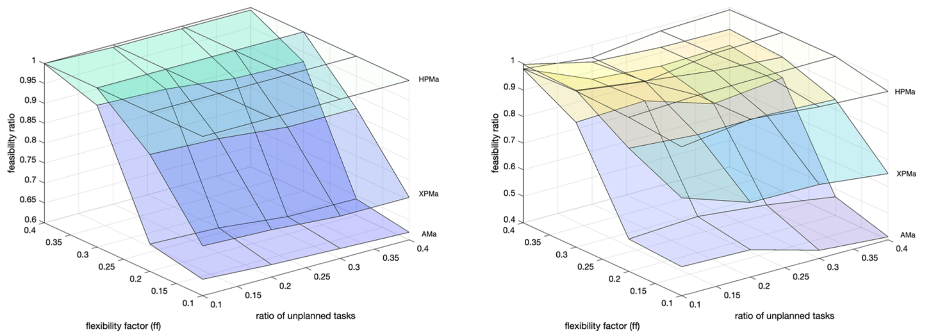
Figure 6 shows the scheduling performances for MRCPSP problems, where greater values are better (shows lower TPT, TPC, TPR and higher TPQ, TPS; see Eqs. (68)–(72)). The HPMa produces the lowest TPT because the scheduling performance of the total project time (TPT%), 2.18, is the best for the proposed algorithm, but the cost of this performance is such that the other scheduling performance indicators are better for TPMa and APMa algorithms (see Fig. 6a).

Although TPMa has produced the lowest ratio of feasible project plans, TPMa reduces the most cost demands for feasible project plans.

APMa is the most effective for increasing quality and allocating resources. Figure 6b shows that HPMa can reduce the TPT most effectively while keeping more tasks, but XPMa and AMa can reduce cost and resources most effectively.

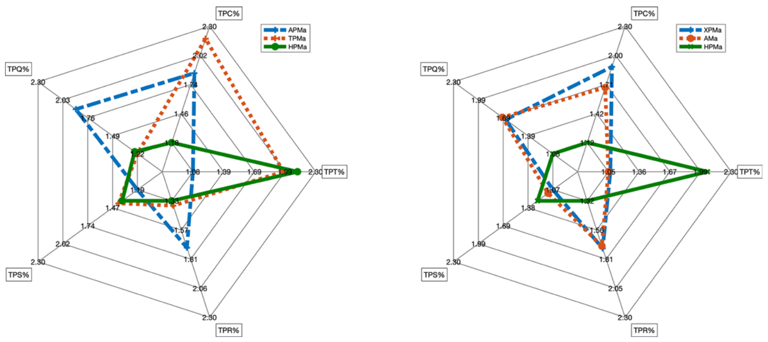
5 Discussion

In flexible project environments, flexible approaches, such as agile and extreme project management approaches, can specify more feasible projects; therefore, these approaches can be more successful than the traditional approaches. At the same time, if there are more completion modes, hybrid approaches can handle both the flexibility and the selection



(a) Hybrid TQCPSP **(b)** Hybrid MRCPSP

Fig. 5 Results of management agents in the case of unplanned tasks



(a) Without unplanned tasks **(b)** With unplanned tasks

Fig. 6 Comparing scheduling performances for project management approaches

from completion modes. This combination of approaches can produce the most feasible projects. The other advantage of the hybrid approach is that it can be used in cases of both flexible (ff is high) and nonflexible environments (ff is low). On one hand, if there is no flexible dependency and supplementary task, the HPMa is the equivalent of TPMa. On the other hand, if there is only one completion mode and we consider only planned tasks, HPMa is the equivalent of APMa. At the same time, in the flexible project environment when unplanned tasks can also be included in the project, HPMa is equivalent to AMa if constraints are fixed, but HPMa = XPMa if the constraint is flexible. Nevertheless, in the general case, if more than one completion mode (technology) is specified, in a flexible project environment HPMa can use both traditional and flexible scheduling techniques in order to specify feasible projects. Figure 5 shows that the flexible environment (higher flexibility factor) can increase the rate of flexibility more than the number of unplanned tasks. However, if extra costs and extra efforts of unplanned tasks are considered, the ratio of flexibility can be increased. Although Fig. 6 shows results only for MRCPSP, where trade-offs between time and cost, cost and quality, time and resources are not assumed, a kind of trade-off between methods can be observed. Figure 6 shows that the proposed HPMa can reduce TPT most effectively, but this method is not superior. Traditional and flexible project management approaches, such as APMa, XPMa, AMa can save more money and more resources. Flexible approaches can provide greater increases in quality and can be more effective in resource allocation. HPMa can be most effective for reducing time because this approach can reorganize the project and can select from completion modes, while other approaches use only one technique; however, other methods can save more resources and can produce more quality, while HPMa is usually closer to the constraints.

6 Summary and conclusion

The proposed hybrid multimode resource-constrained project scheduling problem approach may bridge the agile and traditional project management approaches. If there are no flexible dependencies or supplementary task completions, the problem amounts to the traditional MRCPSP problem. The hybrid project management (HPM) approach combines methods (e.g., TQCTP. MRCPSP) from traditional project management, and structuring and scoring techniques from agile project management.

The proposed algorithm is a fast, efficient method that supports the hybrid project management approach. The algorithm is able find an optimal solution according to predefined preferences over factors such as time, cost and quality. The proposed algorithm is able to handle unplanned tasks; therefore, this method can be used in a continuously changing environment. This paper shows that a hybrid approach can be an adequate alternative to flexible project management approaches. Traditional and flexible approaches can and should be combined in order to specify more feasible projects, and increase the scheduling performance of the project duration.

The developed matrix-based method and proposed exact algorithm may be important and essential components of a project expert system supporting strategic decision making, particularly in cases of large, complex, and flexible projects.

7 Limitations and future works

The proposed model extends the traditional multimode resource-constrained project scheduling problem; however, in this model, only renewable resources (e.g., human resources) and one nonrenewable resource (e.g., cost demand) are considered. In project management, renewable, nonrenewable, and semirenewable resources may also be important parameters. Therefore, this extension will be considered in future research. Another possible application of this method is in risk management and risk analysis. Supplementary and unplanned task completions can model changes in management or client claims. Flexible task dependency can model technological changes. In these cases, a more appropriate matrix-based model could be specified, and the efficiency on risk mitigation of traditional and flexible project management approaches can be compared for different kinds of project plans.

Acknowledgements Open access funding provided by University of Pannonia (PE).

References

1. Babu, A.J.G., Suresh, N.: Project management with time, cost, and quality considerations. Eur. J. Oper. Res. **88**(2), 320–327 (1996). https://doi.org/10.1016/0377-2217(94)00202-9
2. Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: notation, classification, models, and methods. Eur. J. Oper. Res. **112**(1), 3–41 (1999). http://ideas.repec.org/a/eee/ejores/v112y1999i1p3-41.html
3. Cram, W.A., Marabelli, M.: Have your cake and eat it too? Simultaneously pursuing the knowledge-sharing benefits of agile and traditional development approaches. Inf. Manag. **55**(3), 322–339 (2018). https://doi.org/10.1016/j.im.2017.08.005
4. Creemers, S.: Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. J. Sched. **18**(3), 263–273 (2015). https://doi.org/10.1007/s10951-015-0421-5
5. De, P., Dunne, E.J., Ghosh, J.B., Wells, C.E.: The discrete time–cost tradeoff problem revisited. Eur. J. Oper. Res. **81**(2), 225–238 (1995). https://doi.org/10.1016/0377-2217(94)00187-H
6. De, P., Dunne, E.J., Ghosh, J.B., Wells, C.E.: Complexity of the discrete time–cost tradeoff problem for project networks. Oper. Res. **45**(2), 302–306 (1997). https://doi.org/10.1287/opre.45.2.302
7. Demeulemeester, E.L., Herroelen, W.S., Elmaghraby, S.E.: Optimal procedures for the discrete time/cost trade-off problem in project networks. Eur. J. Oper. Res. **88**(1), 50–68 (1996). https://doi.org/10.1016/0377-2217(94)00181-2
8. Dingsøyr, T., Nerur, S., Balijepally, V.G., Moe, N.B.: A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. **85**(6), 1213–1221 (2012). https://doi.org/10.1016/j.jss.2012.02.033
9. Elmaghraby, S.E.: Activity Networks: Project Planning and Control by Network Models. Wiley, New York (1977)
10. Goldberg, A.V.: An efficient implementation of a scaling minimum-cost flow algorithm. J. Algorithms **22**(1), 1–29 (1997). https://doi.org/10.1006/jagm.1995.0805
11. Habibi, F., Barzinpour, F., Sadjadi, S.: Resource-constrained project scheduling problem: review of past and recent developments. J. Proj. Manag. **3**(2), 55–88 (2018)
12. Hazır, Ö., Erel, E., Günalay, Y.: Robust optimization models for the discrete time/cost trade-off problem. Int. J. Prod. Econ. **130**(1), 87–95 (2011)
13. Kolisch, R., Sprecher, A.: PSPLIB—a project scheduling problem library: OR software—ORSEP operations research software exchange program. Eur. J. Oper. Res. **96**(1), 205–216 (1997). https://doi.org/10.1016/S0377-2217(96)00170-1
14. Kosztyán, Z.T., Kiss, J.: PEM–a new matrix method for supporting the logic planning of software development projects. In: DSM 2010: Proceedings of the 12th International DSM Conference, Cambridge, UK (2010)
15. Kosztyán, Z.T.: Exact algorithm for matrix-based project planning problems. Expert Syst. Appl. **42**(9), 4460–4473 (2015). https://doi.org/10.1016/j.eswa.2015.01.066
16. Kuhrmann, M., Diebold, P., Mnch, J., Tell, P., Trektere, K., Mc Caffery, F., Vahid, G., Felderer, M., Linssen, O., Hanser, E., Prause, C.: Hybrid software development approaches in practice: a European perspective. IEEE Softw. **PP**(99), 1–1 (2018). https://doi.org/10.1109/MS.2018.110161245
17. Moder, J.J., Phillips, C.R., Davis, E.W.: Project management with CPM, PERT, and precedence diagramming. Van Nostrand Reinhold; Subsequent edition (1983). ISBN-10: 0442254156. ISBN-13: 978-0442254155
18. Monghasemi, S., Nikoo, M.R., Fasaee, M.A.K., Adamowski, J.: A novel multi criteria decision making model for optimizing time–cost–quality trade-off problems in construction projects. Expert Syst. Appl. **42**(6), 3089–3104 (2015). https://doi.org/10.1016/j.eswa.2014.11.032
19. Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Oper. Res. **41**(2), 338–350 (1993). https://doi.org/10.1287/opre.41.2.338
20. Orm, M.B., Jeunet, J.: Time cost quality trade-off problems: a survey exploring the assessment of quality. Comput. Ind. Eng. **118**, 319–328 (2018)
21. Rahimian, V., Ramsin, R.: Designing an agile methodology for mobile software development: a hybrid method engineering approach. In: Second International Conference on Research Challenges in Information Science, 2008. RCIS 2008, pp. 337–342 (2008). https://doi.org/10.1109/RCIS.2008.4632123
22. Sack, J.-R., Strothotte, T.: A characterization of heaps and its applications. Inf. Comput. **86**(1), 69–86 (1990). https://doi.org/10.1016/0890-5401(90)90026-E
23. Said, S.S., Haouari, M.: A hybrid simulation-optimization approach for the robust discrete time/cost trade-off problem. Appl. Math. Comput. **259**, 628–636 (2015). https://doi.org/10.1016/j.amc.2015.02.092

24. Tang, D., Zhu, R., Tang, J., Xu, R., He, R.: Product design knowledge management based on design structure matrix. Adv. Eng. Inf. **24**(2), 159–166 (2010). https://doi.org/10.1016/j.aei.2009.08.005. (Enabling technologies for collaborative design)
25. Tareghian, H.R., Taheri, S.H.: On the discrete time, cost and quality trade-off problem. Appl. Math. Comput. **181**(2), 1305–1312 (2006). https://doi.org/10.1016/j.amc.2006.02.029
26. Toğan, V., Eirgash, M.A.: Time–cost trade-off optimization of construction projects using teaching learning based optimization. KSCE J. Civ. Eng. **23**(1), 10–20 (2019). https://doi.org/10.1007/s12205-018-1670-6
27. Tyagi, M.K., Munisamy, S., Reddy, L.S.S.: Traditional and hybrid software project tracking technique formulation: state space approach with initial state uncertainty. CSI Trans. ICT **2**(2), 141–151 (2014). https://doi.org/10.1007/s40012-014-0037-5
28. Van Peteghem, V., Vanhoucke, M.: An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. Eur. J. Oper. Res. **235**(1), 62–72 (2014). https://doi.org/10.1016/j.ejor.2013.10.012
29. Vanhoucke, M.: New computational results for the discrete time/cost trade-off problem with time-switch constraints. Eur. J. Oper. Res. **165**(2), 359–374 (2005)
30. Vanhoucke, M.: Measuring the efficiency of project control using fictitious and empirical project data. Int. J. Proj. Manag. **30**(2), 252–263 (2012). https://doi.org/10.1016/j.ijproman.2011.05.006
31. Wysocki, R.K.: Effective Project Management: Traditional, Agile, Extreme, 5th edn. Wiley, New York (2009). ISBN: 0470423676. http://www.amazon.de/Effective-Project-Management-Traditional-Extreme/dp/0470423676