scientific data

Check for updates

OPEN

DATA DESCRIPTOR

Compound Matrix-Based Project Database (CMPD)

Zsolt T. Kosztyán[1,3] ✉ & Gergely L. Novák[2,3]

The impact of projects is vital, from business operations to research to the national economy. Therefore, management science and operation research have extensively studied project scheduling and resource allocation for over six decades. Project databases were proposed to test algorithms, including simulated or real, single or multiprojects, and single-mode or multi-mode projects. However, the dozens of project databases are extremely heterogeneous regarding the file structure and the features of the modeled projects. Furthermore, the efficiency and performance of project scheduling and resource allocation algorithms are susceptible to the characteristics of projects. Therefore, the proposed Compound Matrix-Based Project Database (CMPD) collects and consolidates the most frequently used project databases. The proposed Unified Matrix-Based Project-Planning Model (UMP) sparse matrix-based model enables the addition of new features to existing project structures, such as completion priorities, structural flexibility, and quality parameters, to broaden the scope of considered projects and to take account of flexible approaches, such as agile, extreme, and hybrid projects.

Background & Summary

Overall, projects contribute almost 20% of a country's GDP[1,2]. Therefore, for approximately six decades, management science and operations research has extensively studied project scheduling issues[3,4]. A novel project scheduling or resource allocation algorithm cannot be published until it is compared with other algorithms in existing project databases.

Most project databases are capable of storing (1) fixed[5,6] project structures; (2) Different types of completion modes[7], including (a) time demands and (b) resource demands; and (3) single[8] and multiple[5] project structures. In addition, several smaller project databases store data that can be assigned to resources rather than activities, for example, the use of skills[9]. However, these databases are not compatible with several others.

The main shortcomings of these databases are that (1) they are quite heterogeneous in terms of file structures and project characteristics[10]; (2) Important features, e.g., quality and priorities, are not included; (3) It is difficult to add data that cannot be directly linked to activities, e.g., skills, organizational hierarchies, responsibilities, etc.; (4) They completely neglect flexibility issues of the project, such as completion priorities and flexible dependencies.

To address this gap, we employed a recently published[10] matrix-based UMP model that can store (1) single- and multimodal projects, (2) individual and multiprojects, and (3) fixed and flexible projects. In addition, features such as quality parameters, costs, and nonrenewable resources can be assigned to tasks as new domains (submatrices). This matrix approach allows further submatrices such as skills[11] and maintainable system parameters[12] to be specified. With the proposed parsers[13], 12 existing, most frequently used project databases (including 23 datasets) are parsed into the proposed unified matrix-based project database, CMPD. The database includes not only single-mode but also multimode data, as well as single- and multiproject data. To validate the proposed CMPD, structural, time-related, and resource-related indicators are implemented[14] to ensure adequate modeling of existing project structures in the proposed matrix-based database.

Project scheduling is an integral part of project management that involves the allocation of resources over time to perform a set of activities with dependencies. The classic resource-constrained project scheduling problem (RCPSP) and its extensions for multiple projects (RCMPSP) and multiple completion modes (MRCPSP) or both (MRCMPSP) are well known in the literature and are suitable for various practical scenarios. Recent extensions incorporating multiple skills[11], flexible resource profiles[15], task priorities, and flexible dependencies[10] have gained significant attention. These advancements have highlighted the necessity of additional attributes in

[1]Department of Quantitative Methods, University of Pannonia, Egyetem str. 10, Veszprém, H-8200, Hungary. [2]Continental Automotive Hungary Ltd., Házgyári str. 6-8., Veszprém, H-8200, Hungary. [3]These authors contributed equally: Zsolt T. Kosztyán, Gergely L. Novák. ✉e-mail: kosztyan.zsolt@gtk.uni-pannon.hu

Projects	Databases	# Instances	# New instances
Single	Boctor[20] https://www.om-db.wi.tum.de/psplib/dataob.html	240	12,480
	Kolisch[21] https://github.com/novakge/project-parsers/tree/master/data	680	8,840
	MMLIB[24] https://www.projectmanagement.ugent.be/sites/default/files/datasets/MMRCPSP/MMLIB.zip	4,320	294,840
	PAT[55] https://www.om-db.wi.tum.de/psplib/dataob.html	110	1,430
	PSPLIB[22] https://www.om-db.wi.tum.de/psplib/getdata_sm.html	13,222	461,448
	Real-life[6] https://www.projectmanagement.ugent.be/sites/default/files/datasets/Empirical/DSLIB.zip	133	1,729
	RG[8] https://www.projectmanagement.ugent.be/sites/default/files/files/datasets/AboutData.zip	2,280	29,640
Multiple	BY[5] https://www.projectmanagement.ugent.be/sites/default/files/datasets/RCMPSP/BY.zip	12,320	160,160
	MPLIB1[7] https://www.projectmanagement.ugent.be/sites/default/files/datasets/RCMPSP/MPLIB.zip	4,550	59,150
	MPLIB2[56] https://www.projectmanagement.ugent.be/sites/default/files/datasets/RCMPSP/MPLIB.zip	35,085	456,105
	MPSPLIB[25] http://www.mpsplib.com/data/mp_all.zip	140	1,820
	RCMPSPLIB[26] https://www.projectmanagement.ugent.be/sites/default/files/datasets/RCMPSP/RCMPSPLIB.zip	26	338

Table 1. Summary of existing project databases, extended with flexible instances.

project scheduling models and the importance of model standardization[16]. Applications beyond projects and other industries could also benefit from the progression of new models[17,18]. For an overview of all problem variants and their characteristics, we refer to the survey of Hartmann and Briskorn[16].

Project databases have long been studied in the project scheduling context, starting with the early Patterson[19] set but constructed without well-defined problem parameters; subsequently, Boctor[20] and other popular artificial databases, such as SMCP/SMFF[21], PSPLIB[22], RG[8,23], and MMLIB[24], play a significant role in benchmarking algorithms. A set of real-life project plans was also collected by Batselier *et al.*[6]. Databases containing multiple projects running in parallel were also established, including MPSPLIB[25], BY[5], RCMPSPLIB[26], and MPLIB[7]. Some of the databases also support multiple completion modes (PSPLIB[22], Boctor[20], and MMLIB[24]). We refer to Table 1 for a list of the selected databases and their references, along with the number of existing and newly added instances.

The PSPLIB dataset is still considered the most popular dataset in recent RCPSP literature[27]. A survey[28] considering the RCMPSP variant highlighted the MPSPLIB dataset as the most commonly used benchmark set.

There are other databases that mostly target different RCPSP variants or candidates for later release. We reviewed only the most important studies without a complete list, which is outside the scope of this paper. The MT dataset[29] is mainly used for schedule risk analysis and earned value management and contains project structures that can be combined with additional resource data; this dataset is called ResSet, which results in the NetRes dataset[30]. DC1[31] and DC2[32] are studied within the context of the RCPSP with discounted cash flows. The CV set[33] and the sD set[27] contain RCPSP instances that are difficult to solve. MISTA2013[34] is a dataset and generator for the multimode resource-constrained multiple project scheduling problem (MRCMPSP) and combines instances from the PSPLIB. The BL[35] and PACK[36] datasets are also modifications of the PSPLIB and were designed for the context of highly disjunctive and cumulative scheduling of RCPSP, respectively. The AT dataset[37] was one of the early sets generated with well-defined problem parameters. The ASLIB[38] dataset contains instances for the resource-constrained project scheduling problem with alternative subgraphs (RCPSP-AS). The MSLIB and SSLIB[39] databases were proposed for the multiskilled resource-constrained project scheduling problem (MSRCPSP). The RACP30[40] dataset was proposed in the context of the resource availability cost problem (RACP).

Most of the existing databases and available methods support only a fixed logic plan or consider a limited number of scheduling alternatives[4,17,41–45]. This approach is intuitive for traditional project management methods, which aim to minimize changes and follow rigid project plans[46,47]. However, agile, hybrid, and extreme project management methods address uncertainty by frequently adapting task priorities and dependencies[48,49]. To overcome the limitations of fixed project plans and to support the features of emerging project management approaches, the Flexible Structures Generator (FSG) enables the respecification of task priorities and dependencies, allowing existing project structures to be flexible. As a result, existing project databases can be extended with both traditional and flexible project structures for further research.

Methods

The database comprises 12 libraries, 23 datasets, and 73,106 instances. An additional 1,561,086 flexible instances were generated using the FSG method. The original databases were collected via a thorough literature review process conducted by the authors, targeting databases of the popular (multimode) resource-constrained (multi) project scheduling problem types, (M)RC(M)PSP. As a result, additional data sources were identified and collected, broadening the list mentioned in existing surveys[28,50]. To maintain data quality, relevant academic papers in project management and scheduling were selected to support the database's integrity and reliability. Some less popular datasets have already been collected and are under preparation for intended future releases.

The unified model for storing project data instances. The proposed *unified matrix-based project planning model* (UMP) can represent all features of widely accepted databases, i.e., individual and multiple projects, single and multimodal completions, and renewable and nonrenewable resources. It contains two mandatory and four supplementary domains (marked with dashed lines), as shown in Fig. 1.

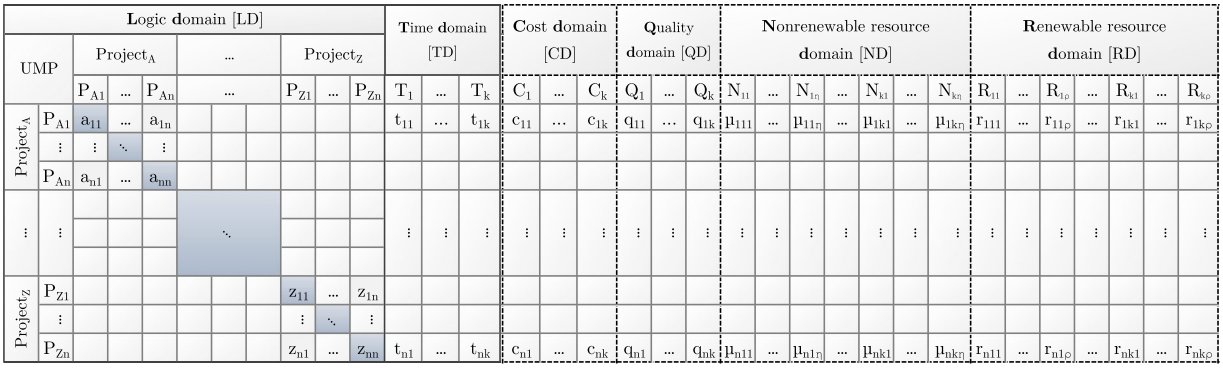

Fig. 1 Structure of the unified matrix-based project planning model (UMP).

LD The logic domain is an n by n matrix, where n is the number of tasks. Each cell contains a value in the interval [0,1].

TD The time domain is an n by k matrix with positive real values, where k is the number of completion modes.

The first mandatory domain is the logic domain, $\mathbf{LD} \in [0,\ 1]^{n\times n}$. The diagonal values in **LD** represent the task priority values. If the diagonal value is 0, the task will not be completed; if the diagonal value is 1, the task is mandatory. If the diagonal value is between 0 and 1, the task is supplementary, indicating that, depending on the decision, it will be either completed or omitted/postponed. The out-diagonal values represent the dependencies between tasks or projects (programs).

The additional supplementary domains are as follows:

CD The cost domain is an n by k nonnegative matrix of the task costs

QD The quality domain is an n by k, nonnegative matrix of the task quality parameters, where the quality parameters are in [0,1]

ND The nonrenewable resource domain is an n by k η nonnegative matrix of nonrenewable resource demands, where η is the number of types of nonrenewable resources

RD The renewable resource domain is an n by k ρ nonnegative matrix of renewable resource demands, where ρ is the number of types of renewable resources

The proposed model thus enables the representation of various projects and features, including flexibility.

Generating flexible structures. Four types of structures are generated for each flexibility level. *The maximal* structures are the equivalents of the original instances. In the case of *minimal* structures, all flexible dependencies and supplementary tasks are excluded; for *minimax*, all supplementary tasks with flexible dependencies are removed; and for *maximin* structures, only their flexible dependencies are removed.

An example of the construction process of flexible structures from existing instances is shown in Fig. 2 for *minimal* structures.

The left side of Fig. 2 shows the original logic domain: the flexibility parameter (fp) is set to 0.4 in this case. In the first step, fixed dependencies/mandatory tasks (denoted by the "X" symbol) become flexible (denoted by "? ", where "?" indicates a value between 0 and 1). The right side of Fig. 2 shows the minimal structure of the project. The center of Fig. 2 shows three possible outcomes from $\binom{10}{4}$. Because the number of "X" symbols is 10, we have $fp = 0.4$. Outcome i retains all tasks but cuts almost all dependencies, while outcome j retains only one task from the original project. In the general case, several dependencies are cut, and several tasks are omitted, e.g., in outcome k. The FSG algorithm has several steps. It processes project instances by iterating through all directories and loading the necessary input variables. For each fixed task $l_{ii} = 1$ and all fixed dependencies $l_{ij} = 1$, $(i \neq j)$ in the logic domain (LD), a matrix with uniform random values rv_{ij} from the range of [0,1] is generated. In the next step, these values are evaluated depending on the type of structure for the given flexibility parameter (fp):

maximal (original): All tasks and dependencies are retained, and fp is set to 0:

$$l_{ij}^{\max} = 1 \tag{1}$$

maximin: tasks are retained, and dependencies are updated:

$$l_{ij}^{\text{maximin}} = \begin{cases} \lceil rv_{ij} \rceil \text{if } i = j \text{ and } rv_{ij} \leq fp, \\ \lfloor rv_{ij} \rfloor \text{if } i \neq j \text{ and } rv_{ij} \leq fp, \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

Minimax: dependencies are kept, and tasks are updated:

$$l_{ij}^{\text{minimax}} = \begin{cases} \lfloor rv_{ij} \rfloor \text{if } i = j \text{ and } rv_{ij} \geq fp, \\ \lceil rv_{ij} \rceil \text{if } i \neq j, \lfloor rv_{ii} \rfloor = \lfloor rv_{jj} \rfloor = 1 \text{ and } rv_{ij} \leq fp, \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

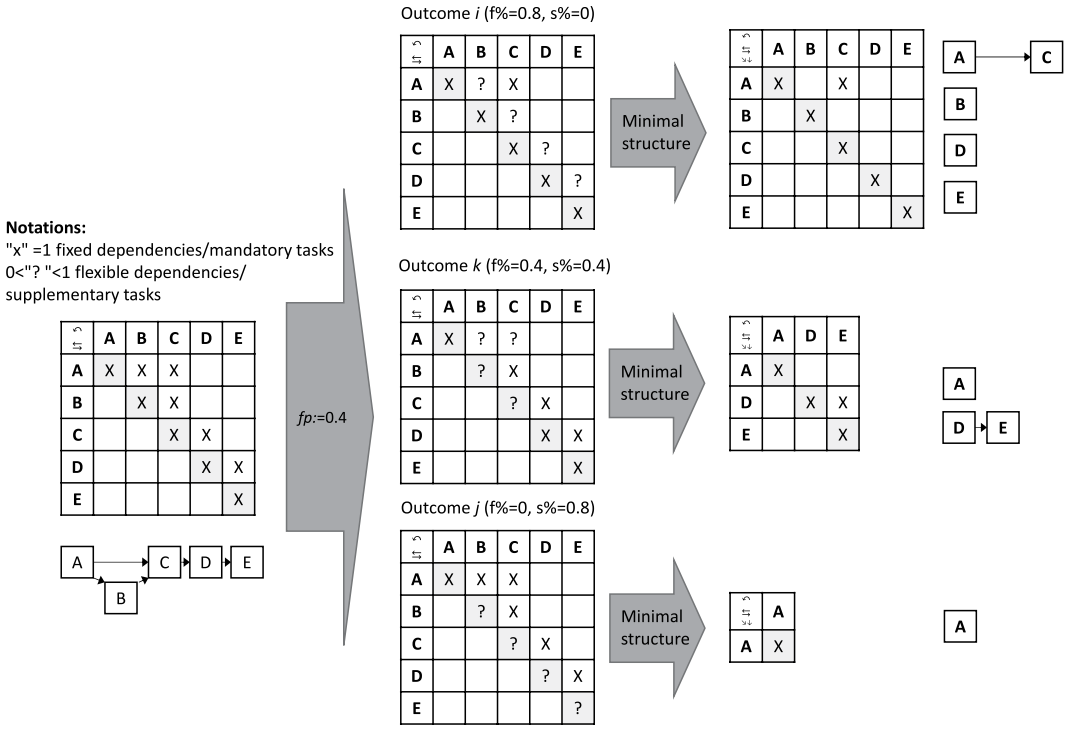

Fig. 2 Example of generating flexible and minimal structures.

minimal: tasks and dependencies are replaced

$$l_{ij}^{\min} = \lfloor rv_{ij} \rfloor, \text{ if } rv_{ij} \leq fp, \tag{4}$$

where $l_{ij}^{\max}$, l_{ij}^{maximin}, l_{ij}^{minimax}, $l_{ij}^{\min}$, are the (i, j) cells of the logic domains of the maximal (original), maximin, minimal, and minimax structures, respectively, with $i, j = 1, 2, \ldots, n$. The $\lceil \cdot \rceil (\lfloor \cdot \rfloor)$ operators denote the rounding up (rounding down) of real numbers to the closest integer. The resulting flexible structures are saved in a designated directory. The random seed of the pseudorandom number generator was fixed for reproducibility. The various structure types add backward compatibility and provide a connection between traditional and flexible project plans and approaches.

Data Records

Since the data originate from the reviewed academic literature, redundancy and quality concerns are mitigated. The database incorporates data from various sources and formats by employing the described unified model. Table 2 lists the main characteristics of the selected databases.

Data profiling was conducted for each database format through examination. None of the databases showed interpretation issues or a lack of extractable data. The methodologies employed by the original authors in generating or collecting the databases were studied in advance to understand the characteristics, methodology, and assumptions of their data. The original data were assessed for important quality characteristics, such as accuracy, consistency, completeness, and currency[51]. Additional consistency checks were executed in the preprocessing phase, ensuring that no contradictory conclusions could be drawn from the original data. Each instance contains descriptive information that can be recalculated from the data itself. These variables are the number of activities and the number of (non)renewable resources. In addition, logical rules can be directly applied for verification and to identify possible conflicts within the data. The number of (non)renewable resources is directly related to the dimension of the constraint vector, while the number of columns in the resource and cost vectors increases proportionally with the number of available modes. Some instances contain the number of precedences or the critical path length, which can be calculated from task precedences and durations. The topological ordering of the logic network, including testing for a lack of cycles in the graphs, was also verified during the process. In the case of generated data, the designed parameter ranges described in the original papers were cross-checked with the help of indicators. Outliers were assessed as individual cases through a detailed examination of the localized data. No missing entries or other anomalies were identified in any of the instances.

To seamlessly integrate diverse data into our model, automated scripts are employed. The necessary conversions or transformations are automatically performed by the developed toolset, which is provided as part of the repository. The provided scripts are designed to interpret and extract all possible attributes and information from each original dataset, ensuring reliable and reproducible data transformation. Format descriptors are collected at the code repository under the 'docs' folder. Instances generated by standard project generators, such as ProGen[21] and RanGen 1[52] and 2[8], of the collected datasets are also supported by the parser. For convenient

Dataset	Set(s)	Instances	# Tasks	# Projects	Type	Method	Mode(s)	Attributes	File extension(s)	Generator
Boctor[20]	Boctor50	120	50	Single	Artificial	Generated	Multi (4)	Time, renewable resources	*.prb	Unpublished
	Boctor100	120	100							
Kolisch[21]	SMCP	200	10, 20, 30, 40	Single	Artificial	Generated	Single	Time, renewable resources	*.rcp	ProGen[21]
	SMFF	480	30							
MMLIB[24]	MMLIB50	540	50	Single	Artificial	Generated	Multi (3)	Time, re/non-renewable resources	*.mm	RanGen1[52]
	MMLIB100	540	100							
	MMLIB+	3,240	50, 100				1*Multi (9)			
PAT[19]	Patterson	110	5–49	Single	Artificial	Generated	Single	Time, renewable resources	*.prb	Unpublished
PSPLIB[22]	SM [j]	2,040	30, 60, 90, 120	Single	Artificial	Generated	Single	Time, re/non-renewable resources	*.sm	ProGen
	MM [c,j,m,n,r]	11,182	10, 12, 16, 20, 30	Single	Artificial	Generated	Multi (1–5)	Time, re/non-renewable resources	*.mm	ProGen
Real-life[6]	—	133	26–151	Single	Empirical	Collected	Single	Time, cost, re-, non-renewable resources	*.p2x	—
RG30[8]	Set1	900	30	Single	Artificial	Generated	Single	Time, renewable resources	*.rcp	RanGen2[8]
	Set2	180	30							
	Set3	240	30							
	Set4	240	30							
	Set5	240	30							
RG300[8]	—	480	300	Single	Artificial	Generated	Single	Time, renewable resources	*.rcp	RanGen1
BY[5]	Rep1-Rep20	12,320	20	Multi (3)	Artificial	Generated	Single	Time, cost, renewable resources	*.xlsm (*.rcmp)	MNPGbrowning2010random
MPLIB1[7]	Set1	833	360	Multi (6)	Artificial	Generated	Single	Time, renewable resources	*.rcmp	Modified MNPG
	Set2	1,463	720	Multi (12)						
	Set3	2,254	1,440	Multi (24)						
MPLIB2[56]	Set1	10,125	500, 1,000	Multi (10, 20, 30)	Artificial	Generated	Single	Time, renewable resources	*.rcmp	Modified MNPG
	Set2	8,640	500, 1,000, 1,500	Multi (10, 20, 30)						
	Set3	8,640	1,000	Multi (20)						
	Set4	7,680	1,000	Multi (20)						
MPSPLIB[25]	—	140	60–2,400	Multi (2, 5, 10, 20)	Artificial	Generated, combined	Single	Time, renewable resources	*.sm, *.xml (*.rcmp)	Various (ProGen / unpublished)
RCMPSPLIB[26]	—	26	60–450	Multi (2–10)	Artificial	Generated, combined	Single	Time, renewable resources	*.txt (*.rcmp)	Various (ProGen, MNPG, unpublished)

Table 2. Summary of all supported databases and their main attributes.

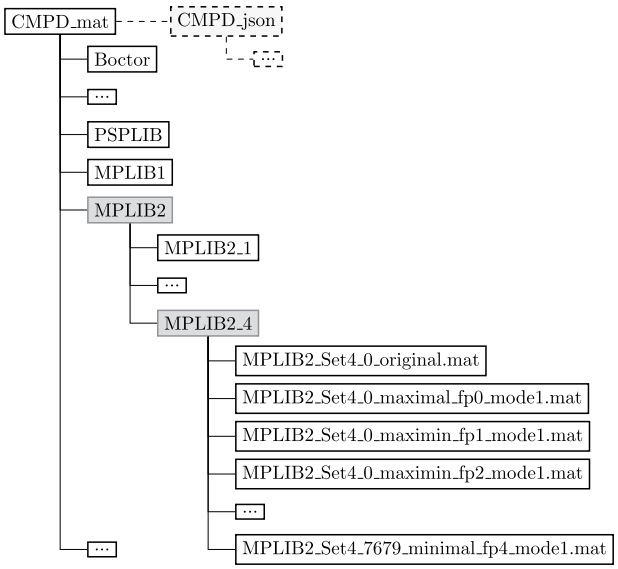

Fig. 3 Database directory structure and filenames.

Name	Type	Min. size [row,col.]	Max size [row,col.]	Value range
constr	double	[1, 3]	[1, Inf]	[−1, Inf]
sim_type	double	[1, 1]	[1, 1]	[1, 3]
struct_type	string	[1, 1]	[1, 1]	[0, 8]
release_dates	double	[1, 1]	[1, Inf]	[0, Inf]
num_projects	double	[1, 1]	[1, 1]	[1, Inf]
num_activities	double	[1, 1]	[1, Inf]	[1, Inf]
num_r_resources	double	[1, 1]	[1, Inf]	[0, Inf]
num_modes	double	[1, 1]	[1, 1]	[1, Inf]
PDM	double	[1, 1]	[Inf, Inf]	[0, Inf]
mode	double	[1, 1]	[1, 1]	[0, Inf]
fr	double	[1, 1]	[1, 1]	[−1, 1]
sr	double	[1, 1]	[1, 1]	[−1, 1]
fp	double	[1, 1]	[1, 1]	[0.0, 0.4]

Table 3. Data dictionary for all CMPD instances.

access to the released version of the CMPD, including flexible instances, please refer to the deposit at Figshare[53]. For databases containing a significant number of files or larger datasets, users can generate instances on their local computers, provided they meet the required hardware and software prerequisites.

The CMPD reflects library and dataset folder names similar to those in the literature within its folder structure. To distinguish the new output format, instances are converted and saved using a predefined naming convention. Each folder contains the standardized output format of the original and flexible instances as MAT files, ensuring consistency. The example folder structure and filenames are shown in Fig. 3.

The libraries are stored in the *CMPD_mat* folder, and *CMPD_json* mirrors it in the widely adopted JSON format. Data libraries can have multiple datasets as subfolders, containing instances as separate files. The naming convention for flexible instances follows the pattern: CMPD_<format>\<library>\<dataset>\<instance#>_<structure_type>_fp<#>_mode<#>.<extension>, where the type of structure can be one of {maximal,maximin,minimax,minimal}; the 'mode' specifies the execution mode of a particular instance; and 'fp' is the flexibility parameter in the range of {0,1,2,3,4}, used to generate the instance, and the extension is either ".mat" or ".json". For the sake of completeness, the original instances are also saved without the 'fp' and 'mode' suffices.

Technical Validation

To ensure the accuracy, reliability, and consistency of the data, several actions were taken. Unit tests were created during the development and verification process to verify the functionality of the data conversion and generation. The data consistency was checked with an automated test suite ensuring that all the instances conformed to the defined data dictionary provided in Table 3.

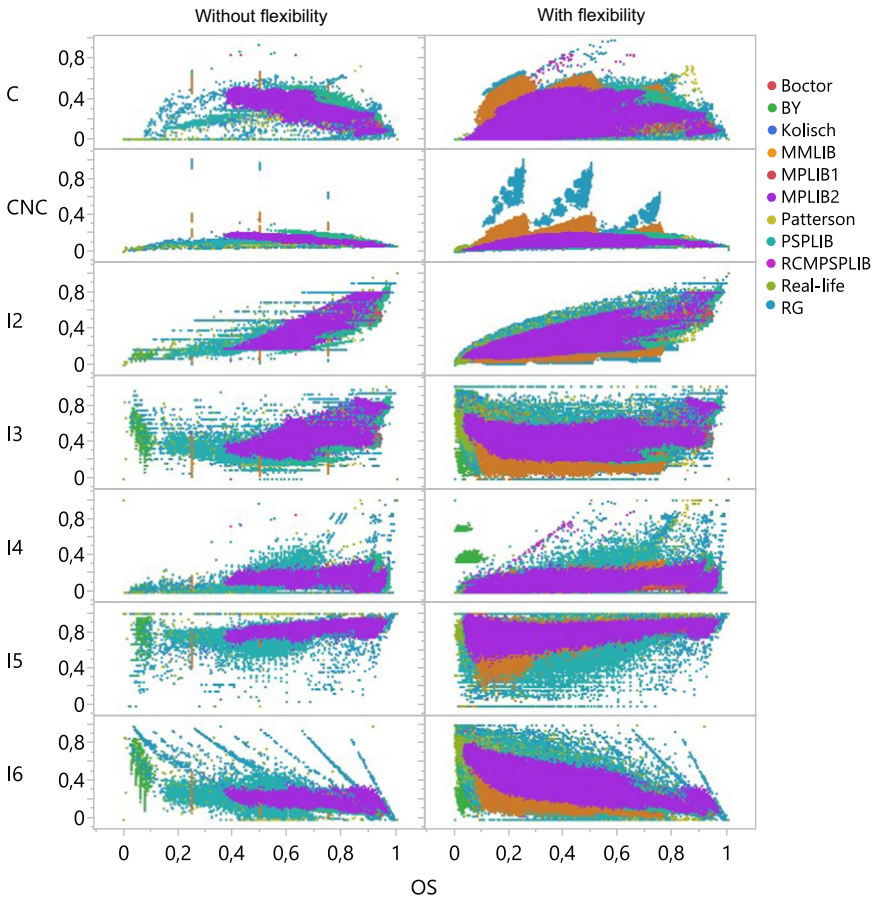

Fig. 4 Topological feature space of all databases concerning flexibility.

The test cases are designed to follow an incremental approach, starting with generic tests, such as checking the folder structure, size and number of files, and adherence to naming conventions. Equivalent tests are further executed on the level of variables, extended with specific cases for variable type, size, invalid or missing entries, and value ranges, according to the provided metadata. The logical relationships between variables are also tested. The matrices and submatrices were verified for size definitions given by the UMP. Possible errors, including exceptions, were handled by either the built-in software libraries or additionally implemented by design. Interactive debugging sessions and fault injection techniques were used to identify any potential exceptions in the parsing process for the different formats.

Reviews were also conducted to check the quality and integrity of the data. Project-related indicators were also used to assess the equivalence of the original and converted data and to compare them with the results from the literature. Subsequent generations of the database were compared to ensure reproducibility on both the Unix and Windows platforms. In addition, joint reviews by experts and paired programming were applied during the development process.

Extensive statistical analyses and comparisons between the datasets were performed to validate the data. These analyses provided an understanding of each dataset's common and unique characteristics. All the databases were checked for the coverage of numerous indicators using scatterplots. Figure 4 shows an example of the comparison between different network-related indicator values for the original and flexible structures. We refer to Kosztyán *et al.*[10] for a detailed description of the applied indicators. The order strength (OS) indicator provided the most uniform coverage of values and was therefore selected for the horizontal axis, while the complexity of network coefficient (CNC) indicator was normalized to the [0,1] range for comparison. Databases such as MPLIB, MMLIB, and RG dominate all feature spaces, while BY covers a smaller but unique area. PSPLIB shows relatively good coverage even without introducing flexibility. Complexity decreased with flexibility, as indicated by C and CNC, bringing value to lower regions, and the seriality of task execution (I2) decreased. In general, the new flexible structures widened the indicator ranges and provided a more diverse set of values that have never been tested by project scheduling and resource allocation algorithms before.

The article[10] associated with the dataset discusses the main results and findings of further evaluations. During the validation process, potential sources of errors, such as formatting differences or missing data entries, were considered and addressed to ensure the validity and reliability of the dataset.

Example: "CMPD_mat\PSPLIB\j30sm\j301_1_NTP_maximal_fp0_mode1.mat"				
Variable name	**Class**	**Size**	**Bytes**	**Description**
PDM	double	30×36	8640	Project Domain Matrix [LD,TD,{CD,QD,ND,RD}]
num_activities	double	1×1	8	Number of tasks
num_projects	double	1×1	8	Number of projects
num_r_resources	double	1×1	8	Number of (renewable) resources
num_modes	double	1×1	8	Number of completion modes (1,…,w)
fp	double	1×1	8	Flexibility parameter {0,0.1,0.2,0.3,0.4}
fr	double	1×1	8	Rate of flexible dependencies [−1, 1]
sr	double	1×1	8	Rate of supplementary tasks [−1, 1]
mode	double	1×1	8	Selected completion mode (0,…,w)
constr	double	1×5	40	Constraints vector [C_{time}, C_{cost},{$C_{quality}$}, {$C_{resource}$}, Cscore]
release_dates	double	1×1	8	Earliest allowed start time of projects
struct_type	string	1×1	150	Type of structure {original,maximal,maximin,minimax,minimal}

Table 4. Variables and their attributes within an instance.

Usage Notes

By loading the database in MATLAB or an open-source alternative, the GNU Octave[54] environment is straightforward, as determined by using either the drag&drop functionality or the built-in 'load' function. The data instances are stored as ".MAT" container or ".JSON" formatted files, each containing the following minimum set of standardized variables:

- *PDM*: This variable contains a matrix with specific domains available for the instance.
- *num_activities*: This variable represents the number of activities in a project. A multiproject is a vector of activity numbers for each project.
- *num_r_resources*: This variable represents the number of renewable resource types.
- *constr*: This variable stores the constraints set for the particular instance.

The instances might contain other optional variables depending on the applicability and actual content. For example, '*fp*' stores the flexibility parameter used by FSG, while '*num_modes*' indicates the number of execution modes available for the original instance. A detailed view of all the variables and their attributes that are stored in the instances is given in Table 4.

Once the instances are loaded in the workspace, variables can be accessed using their respective names, or it is also possible to access and change variables in the MAT files without loading them into memory.

If necessary, the MAT files can be manipulated and saved during the research process. Additionally, it is possible to extend the database with calculated indicator values, providing additional data to work with. The database itself is designed to ease future expansions, enabling the inclusion of new libraries, datasets, and instances. The structured nature of the database enables easy versioning, which can be managed through the popular GitHub platform and MathWorks site. To ensure the integrity of future updates and prevent any negative impacts or regressions, automated unit tests and use cases are implemented as part of the maintenance process. Users can run all available tests using the *'runtests'* command executed in the source code folder. The source files and original databases are securely stored and made accessible through a public GitHub repository. Any academic or professional contributions to the repository and database management are handled within the GitHub platform, which facilitates discussion, issue reporting, and pull request processes and is maintained by key users.

Code availability

The source code is tracked in the Git versioning system and can be publicly accessed from the repository at https://github.com/novakge/project-parsers and https://github.com/novakge/project-indicators without registration. It is licensed under the terms GNU General Public License v3.0. A runnable (reproducible) code capsule can be found at Code Ocean. The code is tested against MATLAB R2020a or later releases with the Global Optimization Toolbox. A developer manual, including examples, is located in the repository's Readme file.

Received: 31 August 2023; Accepted: 15 March 2024;
Published online: 27 March 2024

References

1. Denizer, C., Kaufmann, D. & Kraay, A. Good countries or good projects? macro- and microcorrelates of world bank project performance. *Journal of Development Economics* **105**, 288–302, https://doi.org/10.1016/j.jdeveco.2013.06.003 (2013).
2. World Bank. *The little data book on financial inclusion 2012* (World Bank Publications, 2012).
3. Brucker, P., Drexl, A., Möhring, R., Neumann, K. & Pesch, E. Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research* **112**, 3–41, https://doi.org/10.1016/S0377-2217(98)00204-5 (1999).
4. Franco-Duran, D. M. & Garza, J. M. D. L. Review of resource-constrained scheduling algorithms. *Journal of Construction Engineering and Management* **145**, 03119006, https://doi.org/10.1061/(ASCE)CO.1943-7862.0001698 (2019).
5. Browning, T. R. & Yassine, A. A. A random generator of resource-constrained multi-project network problems. *Journal of Scheduling* **13**, 143–161, https://doi.org/10.1007/s10951-009-0131-y (2010).

6. Batselier, J. & Vanhoucke, M. Construction and evaluation framework for a real-life project database. *International Journal of Project Management* **33**, 697–710, https://doi.org/10.1016/j.ijproman.2014.09.004 (2015).
7. Van Eynde, R. & Vanhoucke, M. Resource-constrained multi-project scheduling: benchmark datasets and decoupled scheduling. *Journal of Scheduling* **23**, 301–325, https://doi.org/10.1007/s10951-020-00651-w (2020).
8. Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B. & Tavares, L. V. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research* **187**, 511–524, https://doi.org/10.1016/j.ejor.2007.03.032 (2008).
9. Myszkowski, P. B., Laszczyk, M., Nikulin, I. & Skowroński, M. Imopse: a library for bicriteria optimization in multi-skill resource-constrained project scheduling problem. *Soft Computing* **23**, 3397–3410, https://doi.org/10.1007/s00500-017-2997-5 (2019).
10. Kosztyán, Z. T., Novák, G., Jakab, R., Szalkai, I. & Hegedüs, C. A matrix-based flexible project-planning library and indicators. *Expert Systems with Applications* **216**, 119472, https://doi.org/10.1016/j.eswa.2022.119472 (2023).
11. Kosztyán, Z. T., Bogdány, E., Szalkai, I. & Kurbucz, M. T. Impacts of synergies on software project scheduling. *Annals of Operations Research* **312**, 883–908, https://doi.org/10.1007/s10479-021-04467-5 (2022).
12. Kosztyán, Z. T., Pribojszki-Németh, A. & Szalkai, I. Hybrid multimode resource-constrained maintenance project scheduling problem. *Operations Research Perspectives* **6**, 100129, https://doi.org/10.1016/j.orp.2019.100129 (2019).
13. Kosztyán, Z. T. & Novák, G. L. Project dataset parsers to matrix-based formats. https://www.codeocean.com/, 10.24433/CO.0837444.v1 (2022).
14. Kosztyán, Z. T. & Novák, G. L. Project indicators and flexibility generator for matrix-based datasets. https://www.codeocean.com/, 10.24433/CO.5304543.v1 (2022).
15. Naber, A. & Kolisch, R. Mip models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research* **239**, 335–348, https://doi.org/10.1016/j.ejor.2014.05.036 (2014).
16. Hartmann, S. & Briskorn, D. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research* **297**, 1–14, https://doi.org/10.1016/j.ejor.2021.05.004 (2022).
17. Čapek, R., Šůcha, P. & Hanzálek, Z. Production scheduling with alternative process plans. *European Journal of Operational Research* **217**, 300–311, https://doi.org/10.1016/j.ejor.2011.09.018 (2012).
18. Zimmermann, A. & Trautmann, N. A list-scheduling heuristic for the short-term planning of assessment centers. *Journal of scheduling* **21**, 131–142, https://doi.org/10.1007/s10951-017-0521-5 (2018).
19. Patterson, J. H. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management science* **30**, 854–867 (1984).
20. Boctor, F. F. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *The international journal of production research* **31**, 2547–2558, https://doi.org/10.1080/00207549308956882 (1993).
21. Kolisch, R., Sprecher, A. & Drexl, A. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science* **41**, 1693–1703, https://doi.org/10.1287/mnsc.41.10.1693 (1995).
22. Sprecher, A. & Kolisch, R. PSPLIB-a project scheduling problem library. *European Journal of Operational Research* **96**, 205–216, https://doi.org/10.1016/S0377-2217(96)00170-1 (1996).
23. Debels, D. & Vanhoucke, M. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research* **55**, 457–469, https://doi.org/10.1287/opre.1060.0358 (2007).
24. Peteghem, V. V. & Vanhoucke, M. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research* **235**, 62–72, https://doi.org/10.1016/j.ejor.2013.10.012 (2014).
25. Homberger, J. A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research* **14**, 565–589, https://doi.org/10.1111/j.1475-3995.2007.00614.x (2007).
26. Vázquez, E. P., Calvo, M. P. & Ordóñez, P. M. Learning process on priority rules to solve the RCMPSP. *Journal of Intelligent Manufacturing* **26**, 123–138, https://doi.org/10.1007/s10845-013-0767-5 (2015).
27. Coelho, J. & Vanhoucke, M. New resource-constrained project scheduling instances for testing (meta-)heuristic scheduling algorithms. *Computers & Operations Research* **153**, 106165, https://doi.org/10.1016/j.cor.2023.106165 (2023).
28. Gómez Sánchez, M., Lalla-Ruiz, E., Fernández Gil, A., Castro, C. & Voß, S. Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research* **309**, 958–976, https://doi.org/10.1016/j.ejor.2022.09.033 (2023).
29. Vanhoucke, M. Using activity sensitivity and network topology information to monitor project time performance. *Omega* **38**, 359–370, https://doi.org/10.1016/j.omega.2009.10.001 (2010).
30. Vanhoucke, M. & Coelho, J. A tool to test and validate algorithms for the resource-constrained project scheduling problem. *Computers & Industrial Engineering* **118**, 251–265, https://doi.org/10.1016/j.cie.2018.02.001 (2018).
31. Vanhoucke, M., Demeulemeester, E. L. & Herroelen, W. On maximizing the net present value of a project under renewable resource constraints. *Management Science* **47**, 1113–1121, https://doi.org/10.1287/mnsc.47.8.1113.10226 (2001).
32. Vanhoucke, M. A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows. *International Journal of Production Research* **48**, 1983–2001, https://doi.org/10.1080/00207540802010781 (2010).
33. Coelho, J. & Vanhoucke, M. Going to the core of hard resource-constrained project scheduling instances. *Computers & Operations Research* **121**, 104976, https://doi.org/10.1016/j.cor.2020.104976 (2020).
34. Wauters, T. *et al*. The multi-mode resource-constrained multi-project scheduling problem. *Journal of Scheduling* **19**, 271–283, https://doi.org/10.1007/s10951-014-0402-0 (2016).
35. Baptiste, P. & Pape, C. L. Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. *Constraints* **5**, 119–139, https://doi.org/10.1023/A:1009822502231 (2000).
36. Carlier, J. & Néron, E. On linear lower bounds for the resource constrained project scheduling problem. *European Journal of Operational Research* **149**, 314–324, https://doi.org/10.1016/S0377-2217(02)00763-4. Sequencing and Scheduling (2003).
37. Alverez-valdes, E. & Tamarit, J. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis, advances in project scheduling (1989).
38. Servranckx, T. & Vanhoucke, M. A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs. *European Journal of Operational Research* **273**, 841–860, https://doi.org/10.1016/j.ejor.2018.09.005 (2019).
39. Snauwaert, J. & Vanhoucke, M. A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research* **307**, 1–19, https://doi.org/10.1016/j.ejor.2022.05.049 (2023).
40. Van Peteghem, V. & Vanhoucke, M. An artificial immune system algorithm for the resource availability cost problem. *Flexible services and manufacturing journal* **25**, 122–144, https://doi.org/10.1007/s10696-011-9117-0 (2013).
41. Creemers, S., Reyck, B. D. & Leus, R. Project planning with alternative technologies in uncertain environments. *European Journal of Operational Research* **242**, 465–476, https://doi.org/10.1016/j.ejor.2014.11.014 (2015).
42. Servranckx, T. & Vanhoucke, M. Strategies for project scheduling with alternative subgraphs under uncertainty: similar and dissimilar sets of schedules. *European Journal of Operational Research* **279**, 38–53, https://doi.org/10.1016/j.ejor.2019.05.023 (2019).
43. Kellenbrink, C. & Helber, S. Scheduling resource-constrained projects with a flexible project structure. *European Journal of Operational Research* **246**, 379–391, https://doi.org/10.1016/j.ejor.2015.05.003 (2015).
44. Tao, S. & Dong, Z. S. Multi-mode resource-constrained project scheduling problem with alternative project structures. *Computers & Industrial Engineering* **125**, 333–347, https://doi.org/10.1016/j.cie.2018.08.027 (2018).

45. Hauder, V. A., Beham, A., Raggl, S., Parragh, S. N. & Affenzeller, M. Resource-constrained multi-project scheduling with activity and time flexibility. *Computers & Industrial Engineering* **150**, 106857, https://doi.org/10.1016/j.cie.2020.106857 (2020).
46. Ciric, D. *et al*. Agile vs. traditional approach in project management: Strategies, challenges and reasons to introduce agile. *Procedia Manufacturing* **39**, 1407–1414, https://doi.org/10.1016/j.promfg.2020.01.314. 25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing August 9-14, 2019 | Chicago, Illinois (USA) (2019).
47. Pellerin, R. & Perrier, N. A review of methods, techniques and tools for project planning and control. *International Journal of Production Research* **57**, 2160–2178, https://doi.org/10.1080/00207543.2018.1524168 (2019).
48. Ciric Lalic, D., Lalic, B., Delić, M., Gracanin, D. & Stefanovic, D. How project management approach impact project success? from traditional to agile. *International Journal of Managing Projects in Business* **15**, 494–521, https://doi.org/10.1108/IJMPB-04-2021-0108 (2022).
49. Wysocki, R. K. *Effective project management*, 8 edn (John Wiley & Sons, Nashville, TN, 2019).
50. Vanhoucke, M., Coelho, J. & Batselier, J. An overview of project data for integrated project management and control. *Journal of Modern Project Management* **3**, 6–21 (2016).
51. Batini, C. *et al*. Data and information quality. *Cham, Switzerland: Springer International Publishing* https://doi.org/10.1007/978-3-319-24106-7 (2016).
52. Demeulemeester, E. L., Vanhoucke, M., Herroelen, W. & Rangen:, A. random network generator for activity-on-the-node networks. *Journal of Scheduling* **6**, 17–38, https://doi.org/10.1023/A:1022283403119 (2003).
53. Kosztyán, Z. T. & Novák, G. L. Compound matrix-based database. *Figshare* https://doi.org/10.6084/m9.figshare.23937978 (2023).
54. GNU Octave. https://www.octave.org. Version 8.2.0 (2023).
55. Patterson, J. H. Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly* **23**, 95–123, https://doi.org/10.1002/nav.3800230110 (1976).
56. Van Eynde, R. & Vanhoucke, M. New summary measures and datasets for the multi-project scheduling problem. *European Journal of Operational Research* **299**, 853–868, https://doi.org/10.1016/j.ejor.2021.10.006 (2022).

Acknowledgements

This work was implemented by the TKP2021-NVA-10 project with support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the 2021 Thematic Excellence Programme funding scheme. Gergely L. Novák's cooperation was supported by the Doctoral Student Scholarship Program of the Co-Operative Doctoral Program of the Ministry of Innovation and Technology financed by the National Research, Development, and Innovation Fund. Zsolt T. Kosztyán's research contribution, supported by Project no. K 142395, has been implemented with support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the K_22 "OTKA" funding scheme.

Author contributions

G.L.N. implemented the parsers; Z.T.K. analyzed the results. All the authors reviewed the manuscript.

Funding

Open access funding provided by University of Pannonia.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Z.T.K.

Reprints and permissions information is available at www.nature.com/reprints.

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

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2024