Műszaki Tudományos Közlemények vol. 13. (2020) 137–141.
DOI: Hungarian: https://doi.org/10.33895/mtk-2020.13.25
English: https://doi.org/10.33894/mtk-2020.13.25

SOLVING EXTENDED PROJECT SCHEDULING PROBLEMS

Krisztián MIHÁLY,[1] Gyula KULCSÁR[2]

University of Miskolc, Faculty of Mechanical Engineering and Informatics, Miskolc, Hungary

[1] *altmihaly@uni-miskolc.hu*

[2] *iitkgy@uni-miskolc.hu*

Abstract

Project based planning and execution is used in various phases of a product lifecycle, starting from the conceptual idea and design through to the manufacturing and maintenance. It is common, that an enterprise or an organization executes more than one project in parallel. These projects may share the same resources and may have differing goals. Based on experience the order of task execution is a key factor, having a major impact on the key performance indicators. Our paper presents an extended model and a scheduling method for resource constrained project scheduling problems.

Keywords: *project scheduling, generation scheme, multi-objective, multi-project, extended RCPSP.*

1. Project scheduling

Project-based task execution can nowadays be found in various business and industrial areas. Project based setup can be used to develop an individual product, to organize a sport event or to execute project-based manufacturing. The topic of project scheduling research has a significant background and despite the accumulated knowledge it is still an active research area, because – not taking special cases into consideration – it belongs to the NP hard problems [1].

In our research we enhanced a known resource constraint project scheduling problem type with new aspects, in order to find effective description models and project scheduling algorithms. We worked out a new scheduling concept, which is based on reactive rule-based methods, and which applied a combination of search and simulation-based methods.

2. Resource constrained project scheduling problem

Resource Constrained Project Scheduling Problem (RCPSP) can be defined as follows [2]:

Given the known set of tasks to be executed T = {1,2,3,…,n}.

Given the known set of resource types K = {1,2,3,…,m}. The capacity of a resource type is limited, meaning the capacity is not infinite for a resource type. Because of this limitation it can happen that at a given time point there is not enough capacity of a resource type to execute tasks in parallel. The capacity of a resource type is known, and there is a deterministic way to calculate it for any future time, based on the initial state and the actual scheduling. This capacity is denoted by R_k, $k \in K$ Resource types are renewable; after execution of any task all blocked capacity is free again and can be used to execute a different task (like people, machines, equipment, etc.).

To define the problem, two constraint-like conditions can be set. For all tasks the prerequisite tasks are known, which set can be noted wit $P_{i.}$ An $i \in T$ task can only be executed, if all $j \in P_i$ task have been executed. To execute a task one or more resource types may be required with predefined capacity needs. A task can be executed only if all capacity needs for all resource types can be completed simultaneously.

The goal of solving a scheduling problem is to construct an order of tasks. Following the given order during execution does not harm the capacity constraints and all pre-emptive conditions are met. An order of tasks fulfilling these requirements is called as executable project scheduling (or scheduling). If more than one executable project scheduling exists, then specialization of the scheduling problem can be done with usage of goal functions. When a scheduling problem solution is constructed, we aimed to find the best possible solution to fit best the given goal functions.

3. Extended model

We extended the model described in the second chapter with the following additional aspects:

3.1. Application of more than one goal function simultaneously

From two executable schedulings from the viewpoint of a goal function that must be minimized, we understand that the one for which the calculated value for the given goal function is less to be the better one. In the case of maximize function, the greater value means the better solution. A known and often used minimize goal function is for example the finish time of the last completed task (C_{max}), the number of finished tasks with latency and the greatest latency (L_{max}). Maximize goal function is, for example, the average resource consumption indicator. Based on real life examples it is hard to describe the suitability of a scheduling with one goal function.

In these situations, a possible technique is the construction of a new goal function, which takes more aspects into consideration. This can be the introduction of a new key performance indicator, which can be used as the target of a goal function or – practically – creation of a new, combined goal function, defined by combining and weighting the known goal functions.

In the designed solver strategy, we use the weighted relative differences of the goal function pair values technique. The main idea is that we always compare two possible schedulings. We calculate all goal function values for both solutions. For one goal function we subtract the difference of the goal function values with the greater value and we get the signed relative difference value. This can be done for all goal functions and we summarize the elementary relative differences. In this way we get the final difference resultant. If goal functions are not of equivalent importance, then the importance of one goal function can be described with its weight factor. At the end the weighted summation of the elementary relative changes defines the relative value of the two checked solutions. This technique can be used for arbitrary type and number of goal functions.

3.2. Parallelly executed projects

The RCPSP problem describes a project with their tasks and boundary conditions. As an extension of this problem we modelled the case where the resources are do not belong to tasks of one project, instead of more than one, parallel-executed projects. The projects are different from each other not just by tasks to be executed with their constraints, but that they have different goal functions.

One possible technique is that we add to the set of tasks two new tasks in the following way:

– one virtual, global GS project starting task $GS \in T$. The GS task is defined as a prerequisite task for all tasks that had no prerequisite task in the original problem;
– one virtual, global GT project finalizing task $GT \in T$. All tasks which were not prerequisite for any other task in the original problem are a prerequisite for GT.

With this modelling, the problem of the parallelly executed projects is reduced to a one project scheduling problem; but due to the different goal functions of individual projects a new goal function shall be defined. In practice this is usually hard to construct.

4. Applied model

We reused the RCPSP problem basic entities during modelling of the extended problem. We define the task, the project and the resources. When defining a project, next to the project identification data and the task assignments the weight of assigned project goal functions can be defined. The list of possible goal functions in the current implementation is fixed, but the applied software architecture is prepared to extend and implement further goal functions.

The resource types can be defined independently from the projects. One resource type can be described with their identifier and capacity availability function. In the current implementation one resource type can have only constant based function, but the designed architecture is prepared to define capacity constraints changing in the time scale.

The tasks, the task capacity needed per resource type and the task pre-emption relations can be defined at task level.

5. Applied scheduler

The applied scheduling algorithm has two main components: one scheduler based on generation schemes, and a search engine based on heuristics.

The main principle of the generation scheme-based project scheduler starts from an empty schedule and in each iteration from a non-scheduled tasks one task is added to the schedule. In the iteration when the next task is selected all the given constraints are taken into consideration, which means all prerequisite tasks were scheduled and all resource requests could be completed simultaneously. We differentiate serial and parallel generation schemes based on the method of selecting the next executable task. The serial scheduling scheme checks mainly the completion of the prerequisite tasks. The decision set contains those tasks for which all prerequisite tasks were completed. From these in the given situation the task with highest priority value (in summary the best candidate) is selected. This task is scheduled in such a way that it is added to the schedule at the earliest time, when the requested resource capacity needs can be fulfilled in the same time.

The parallel generation scheme focuses mainly on the earliest starting time. In an intermediate state the decision set contains the earliest executable tasks. In comparison with the serial generation scheme not all executable tasks are the basis of task selection. From the decision set the same priority-based technique is used to select a candidate, like the serial generation scheme.

If more than one task meets the selection criteria (they have the same priority) then the generation scheme selects randomly one of them.

In our approach instead of random selection we applied a deterministic selection, as in other publicised extensions. Instead of one heuristic method we combine more than one task selection heuristic, where the importance weights of a heuristic can be defined by the user. The generation scheme can be influenced by parameters defined by a user.

Using the generation scheme as a simulation module we implemented a heuristic search, where the search engine alters the tasks' selection heuristic weights.

6. Implementation

We developed a new scheduling software for the extended problem, whose main modules are depicted in **Figure 1.**

The modules are implemented by use of object-oriented ABAP. The modules are separated from each other with ABAP interfaces. The aim of the applied interfaces is to support the functional test automation possibilities and to support the implementation alternatives.

The realized functions and responsibilities of each module:

– *RCPSP problem*
  It stores the extended model data in the runtime. It has modelled read and write interfaces. The aim of the modelled interfaces is to separate concerns between modules;
– *RCPSP solution*
  It stores the extended model one executable scheduling. It stores the order of individual tasks in the schedule, the scheduling time of the different resource types. It has modelled read and write interfaces;
– *Generation scheme*
  It is responsible for solving the extended RCPSP problem based on serial generation scheme. It uses the RCPSP problem read interface as input, as well as the decision step controlling heuristic selection parameters with their weights. The generation scheme stores the solution via the RCPSP solution write interface;
– *Heuristic search engine*
  The heuristic search engine alters the heuristic search parameters based on the user defined goal functions and the previously generated RCPSP solution(s) and generates new RCPSP solution(s);
– *Scheduling evaluator*
  The scheduling evaluator evaluates the executable schedule based on the user defined goal functions;
– *Data source*
  More than one data source integration is available, each with different function. The aim of the benchmark data handler is to map known scheduling problems from their own format to the RCPSP format. The aim of project management system handler is to map the SAP PM module described project data to the extended model format. A further possible data source is the example database, which is used to check the functional correctness of the solver via integration tests.

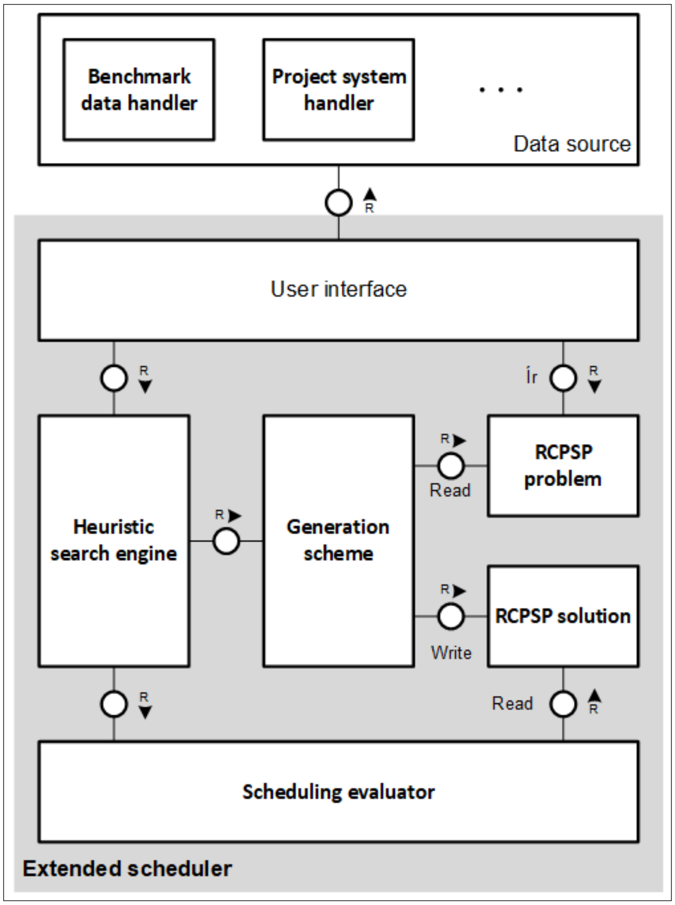

Figure 1. *Block diagram of extended project scheduler.*

7. Results

The extended model and the designed scheduler have been implemented in an own deployed SAP NetWeaver 7.50 development test environment. The model and the algorithm do not contain SAP specific elements; both can be mapped to any object-oriented programming language. The main reason of our decision is to benefit integration later with an SAP PM module directly on SAP platform.

The implemented solution was compared with known basic problems and their solution, where we applied two methodologies.

In the first case we mapped problems with special boundary conditions to the extended model, for which optimal scheduling algorithms are provided. It is known, that for the non-pre-emption Flow Shop problem the Johnson algorithm gives the optimal solution in the case of the C_{max} goal function [3]. Our heuristic-based solver found, in most cases (> 95 %) the optimal solution for small problems.

In the second case the base RCPSP problem was mapped to the extended model and we compared the results for C_{max} goal function with publicized best results [4], [5]. We compared our results with the benchmark problems best results. We observed that for small problems in most cases (> 70 %) the best-known solution was found by the scheduler.

8. Conclusions

In our paper we summarized the extended, resource constrained, multi-project, multi-objective main characteristics.

The extended model is capable of solving known problems. Regarding the difference of dynamic aspects, the specialized algorithms provide better performance indicators.

The proposed solution concept can solve a high variety of scheduling problems. This was proved with the implemented software. Most of the classical scheduling problems can be defined as a specialized case of the extended model. It means that the software can solve these problems. This is very important for practical aspects. For the different companies such an application does not require individual development, it is not necessary to extend the solution at the source code level. We continue the development of the algorithms based on the results.

The main direction of further development is to work out additional generation schemes to have more granular scheduling strategies. Another planned direction is to extend the heuristic algorithms with new modification operators. Last, but not least we work on a new comparison algorithm where a new task is selected in the task selection step.

References

[1] Garey M. R., Johnson D. S.: *Computers and Intractability; A Guide to the Theory of NP-Completeness.* 1st Edition, Series of Books in the Mathematical Sciences, W. H. Freeman; 1979.

[2] Kolisch R, Hartmann S.: *Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*. In: Weglarz J. (ed) *Project scheduling. Recent Models, Algorithms and Applications*. International Series in Operations Research & Management Science 14., Springer, 1999, 147–78.

[3] Johnson, D. B.: *Efficient Algorithms for Shortest Paths in Sparse Networks*. Journal of the ACM, 24/1. (1977) 1–13. https://doi.org/10.1145/321992.321993

[4] Kolisch R., Sprecher A.: *PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program*. European Journal of Operational Research, 96/1. (1996). 205–216. https://doi.org/10.1016/S0377-2217(96)00170-1

[5] Project Scheduling Problem Library – PSPLIB: *Datasets*. (accessed on: 2019. december 1.) http://www.om-db.wi.tum.de/psplib/data.html