HUNGARIAN JOURNAL OF
INDUSTRY AND CHEMISTRY
Vol. 49(2) pp. 53–58 (2021)
hjic.mk.uni-pannon.hu
DOI: 10.33927/hjic-2021-22

EXPERIMENTAL IMPLEMENTATION OF A RESOURCE-CONSTRAINED MULTI-PROJECT SCHEDULING PROBLEM SOLVER

Krisztián Mihály*[1], Mónika Kulcsárné Forrai[1], and Gyula Kulcsár[1]

[1]Department of Information Engineering, University of Miskolc, Egyetemváros, Miskolc, 3515, HUNGARY

Nowadays, project-based planning and execution are becoming more and more important in product life cycles; from the conceptual idea and its design to manufacturing and maintenance. Companies execute several projects simultaneously and these projects usually share the same resources resulting in conflicting situations. Since all projects have different requirements, constraints and goals that need to be achieved, the creation of a company-wide optimal schedule is difficult if all the aspects of each project are to be taken into consideration. Our paper presents an extended model and a scheduling problem-solving approach for resource-constrained, multi-project scheduling problems. In addition to defining the theoretical solution, its experimental implementation and results are presented.

Keywords: resource-constrained, multi-project, advanced project scheduling, ABAP

1. Introduction

Research into project scheduling has a solid and extensive background, moreover, it is still an active research area because – by not taking special cases into consideration – it belongs to the NP-hard (non-polynomial) problems [1]. Detailed reviews of project scheduling can be found in Refs. [2] and [3]. Scheduling problems concerning manufacturing optimization is an active research area [4]. NP-hardness renders any brute force-based calculation for practical problem sizes unrealistic and heuristic or search-based approaches are required to identify a quasi-optimal solution. In our work, an extended model for a resource-constrained multi-project scheduling problem has been developed with an advanced solver concept. The solver is based on a deterministic, rule-based, constructive schedule generation and is combined with search as well as simulation-based methods.

2. Resource-constrained project scheduling problem

Our extended model is based on the well-known resource–constrained project scheduling problem (RCPSP). First of all, the RCPSP is presented. In this paper, the following formulation is used to describe the RCPSP.

A given set of tasks (*activities, operations*) must be executed. The full set of tasks is denoted by $T = \{1, 2, 3, \ldots, n\}$. There is a given set of resource types $K = \{1, 2, 3, \ldots, m\}$. Practically speaking, a resource type can represent machines, people or executors. The capacity of any resource type is known, limited and constant in the time horizon. The capacity of a resource type is denoted by R_k, $k \in K$. Resource types are renewable, that is, after the execution of a given task, all blocked capacity becomes available again and can be used to execute a different task. There is no decreasing or increasing effect on the capacity of the task execution process (*amortization*). A deterministic method can be used to calculate the capacity of the resource type at any future time based on the initial state and actual scheduling.

Tasks are executed on the required resources. Task execution on the assigned resource types cannot be stopped or broken; once the execution has been started, the task must be completed. Pre-emption is not allowed. The processing time of each task is given. A virtual task can be defined, which does not require any resources.

The definition of the scheduling problem is subjected to two constraints: precedence constraints, that is, the prerequisite relation between tasks, and resource constraints. If the task j ($j \in T$) has immediate predecessor tasks defined by set P_j, then all tasks i ($i \in P_j$) must be completed before task j executed. It is forbidden for any circle to be located in the precedence graph.

If task j ($j \in T$) requires one or more resources, they are defined by a set of pairs $(r_{j,k})$. It is forbidden to exceed the maximum available capacity of the resource type for any given resource need $(r_{j,k}) \leq R_k$, $\forall k \in K$, $j \in T$.

Let F_j denote the completion time of task j ($j \in T$). A feasible solution can be described as a vector, in which the completion time of each task is given by the corre-

*Correspondence: altmihaly@uni-miskolc.hu sponding F_j. Let $A(t) = \{j \in T | F_j - p_j \leq t \leq F_j\}$ denote the set of tasks that are being processed at time t.

The objective of the RCPSP is to assign a feasible completion time for each task such that the makespan of the project is minimized, while the precedence and resource constraints are met.

3. Multi-project scheduling problems

The allocation of resources and scheduling tasks for multiple projects are NP hard problems that are more difficult than scheduling a single project [1].

In the literature, different classical optimization methods have been used to solve multi-project scheduling problems. A zero-one programming approach has been proposed [2] and an integer programming problem for generating the schedule as well as a simulation method for testing heuristic rules to choose the best schedule introduced [3]. Deckro et al. formulated the multi-project scheduling problem as a general integer programming model and presented a decomposition approach to solve large problems [5]. Jolayemi proposed an integer programming approach for multi-project scheduling problems and considered a special penalty function [6]. These outlined studies provide good solutions to small problems by using traditional optimization approaches. If the size of a problem is medium or large, then the classical methods cannot solve the problems within a reasonable execution time.

In the literature, several heuristic and metaheuristic methods can be found to solve multi-project scheduling problems. Researchers have developed many effective algorithms to generate feasible solutions to multi-project scheduling problems [7]. The goals of these developments were to increase the efficiency of the heuristic methods, extend the scope of the problems with new methods and reduce the computation time.

Many researchers have applied artificial intelligence methods to solve multi-project resource-constrained scheduling problems. For example, Kim et al. proposed a combined genetic algorithm to create a schedule [8] in order to minimize the project completion time. Kumanan et al. applied a genetic algorithm-based approach for generating the schedule of multi-project scheduling problems [9]. The objective function of the optimization was also to minimize the project completion time. Furthermore, Damak et al. proposed a variant of a genetic algorithm using a local search strategy to solve the problem by regarding the minimization of any project delays as the optimization goal [10].

After reviewing the related literature, it can be concluded that efficient and flexible methods need to be developed to improve the quality and robustness of the solutions of resource-constrained multi-project scheduling problems.

4. An extended model

The RCPSP describes a project along with its tasks and boundary conditions. As an extension of this problem, a case when different shared resources need to be assigned not only to individual projects but also to a set of projects executed in parallel has to be modelled. In our model, tasks that belong to many projects simultaneously are permitted. The projects can be differentiated from each other. On the one hand, the composition and constraints of each project can be unique, but on the other hand, projects can also have individually defined goals (objective functions).

In our model, a model transformation technique can be used. The essence of this possible transformation is that two new virtual tasks can be added to the set of tasks in the following way:

1. A new virtual task can be created as the starting task of the global project GS $\in T$. The GS task is defined as a predecessor task to all the tasks which did not have a predecessor task in the original problem.

2. Another new virtual task can also be added to the model. This task becomes the finalizing task of the global project GT $\in T$. All the tasks, which have no successor tasks in the original problem, become predecessor tasks for GT.

By applying this model transformation, the problem of projects executed in parallel is reduced to a single project scheduling problem. However, due to the different objective functions of individual projects, a new structure of objective functions has to be defined. This modelling approach is usually hard to construct in practice.

Instead of reducing the problem to a single RCPSP, the basic entities were reused in the extended problem and additional defining parameters as well as constraints introduced.

The resource types can be defined independently from the projects. One resource type can be described by its identifier and available capacity function. In the current implementation, each resource type can only have a constant basic capacity function, but the designed architecture is capable of defining capacity constraints that change over time.

The tasks, the required capacity with regard to the execution of tasks for each resource type and the precedence relations of tasks can be defined at the task level.

When creating a project, the project identification data and the task assignments can be defined. Furthermore, the weight of assigned project goals (objective functions) can also be specified. Although the list of possible objective functions in the current implementation is fixed, the applied software architecture is capable of extending and implementing further objective functions.

5. Problem-solving approach

The applied scheduling algorithm has two main components: the first is the project scheduler based on schedule

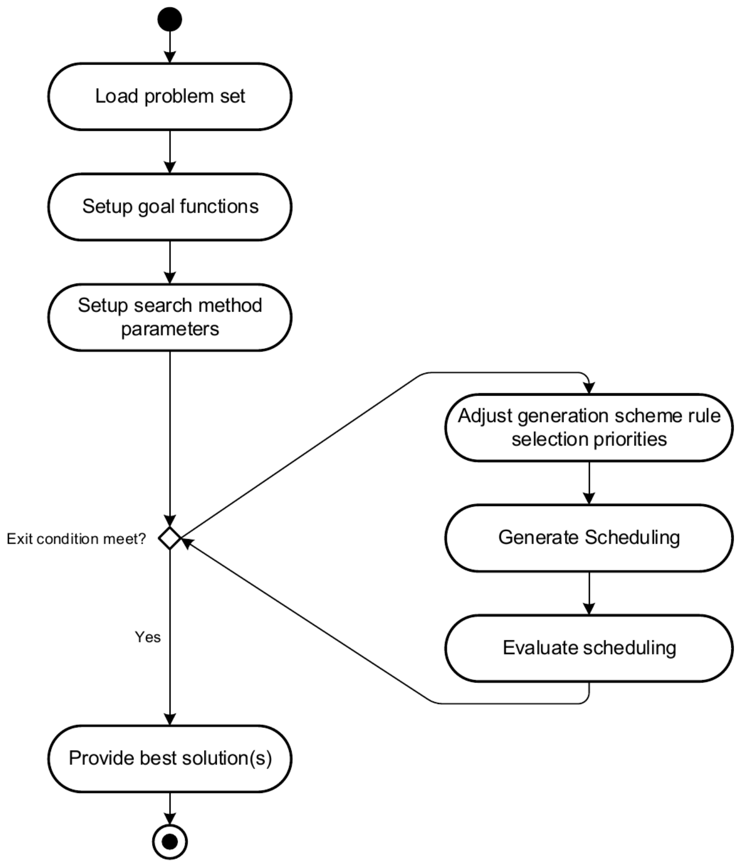

Figure 1: Problem-solving approach

generation schemes and the second is the search engine based on heuristics.

The main principle of the schedule generation scheme-based project scheduler starts with an empty schedule and during each iteration a suitable task can be selected from the set of non-scheduled tasks before being added to the schedule. In the iteration, when the next task is selected, all the given constraints are taken into consideration, which means that all prerequisite tasks are scheduled and all resource requests can be completed simultaneously. A differentiation is made between serial and parallel schedule generation schemes based on the method used to select the next executable task. The serial schedule generation scheme predominantly checks that the prerequisite tasks have been completed. The decision set contains tasks whose prerequisite tasks have all been completed. From this set of tasks, the task with the highest priority value, namely the best candidate, is selected. This task is scheduled in a way that it is added to the schedule as soon as possible, that is, when the required resource capacity can be fulfilled at the same time.

The parallel schedule 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 schedule generation scheme, not all executable tasks form the basis for selecting tasks. From the decision set, the same priority-based technique is used to select a candidate like in the serial schedule generation scheme. If more than one task meets the selection criteria, that is, they have the same priority, the schedule generation scheme randomly selects one of them.

In our approach, instead of making a random selection, a deterministic selection was applied, e.g. minimal slack, latest finish time, etc. Instead of only using one heuristic method, many task-selection heuristics are combined where the importance weighting of a heuristic component can be defined by the user. The schedule generation scheme can be influenced by parameters defined by a user or any consumer.

Using the schedule generation scheme as a simulation module, a heuristic search was implemented where the search engine alters the task-selection behavior using heuristic weights. The main flow of this problem-solving approach is depicted in Fig. 1.

In the first step, the problem set is loaded into the internal data model. The user can alter the project goal (objective functions) if the problem set needs to be changed. The user can alter the parameters of the default search module before the scheduler is started. Based on the given parameters, the search module executes simulations using the selected schedule generation scheme. The generated schedule is evaluated based on the defined goal (objective functions) and the search module can decide to reit-

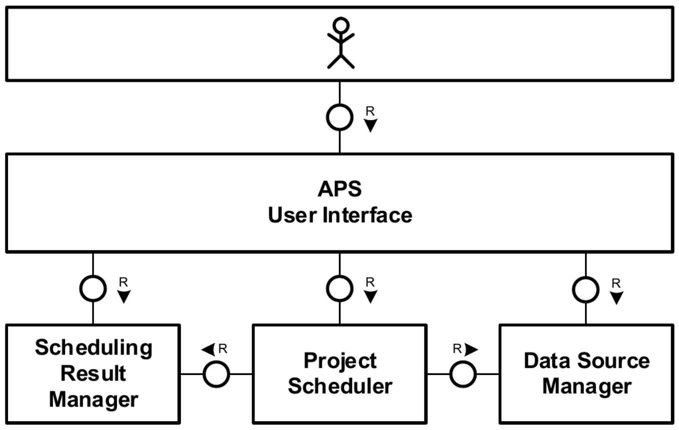

Figure 2: APS system agents

erate or exit. Among the simulations, the behavior of the schedule generation scheme is altered via selection rules.

6. Implementation

Enterprise Resource Planning (ERP) systems are predominantly used to manage all resources in an enterprise. An ERP system consists of a set of different modules responsible for coupled, separated and integrated application areas, e.g., manufacturing, finance, project management, etc. Although our study focused on a Systems Applications and Products (SAP) ERP system, our conceptual model can be used by other ERP vendors as well, after adjusting the implemented artefacts to suit the vendor-specific platform.

As with all standard project management (PM) systems, the SAP ERP PM supports the management of project-related entities such as the creation of projects, definition of tasks and dependencies, as well as definition of resources and their task assignments. In the execution phase of the project, execution reporting and collaboration with external parties are covered. In SAP ERP, this module is referred to as PM or project portfolio management (PPM) [11, 12].

A standalone advanced project scheduling (APS) component on the SAP NetWeaver platform was implemented. This high-level concept is presented in Fig. 2. The user accesses the APS user interface via the SAP GUI (graphical user interface). In this graphical user interface, the user can select the data set, decide where the projects can be selected and calibrate the scheduling parameters. When the project scheduler is started, the actual data is read from the data source, the detected scheduling algorithms are executed, and the scheduling results are stored via the scheduling result manager. The scheduling result is displayed in the APS user interface.

The main component of the system is the project scheduling agent. The internal structure of this component is depicted in Fig. 3. The scheduler consists of two disjoint sets of entities; one is visible to APS consumers, while the other is not. The aim of separating the disjoint sets is to provide a stable interface to code against it and allow the implementation used to be changed. The external interface consists of the defining interfaces of the supported project scheduling problem (CPM, RCPSP), the reading interface of the resource manager and a solver factory.

RCPSPs in the absence of resource constraints can be solved using the critical path method (CPM) algorithm. The CPM result may be an import parameter to calculate a dynamic priority rule, therefore, the RCPSP may require a CPM solver and the CPM is modelled as an individual solver entity. The RCPSP solver has different implementation alternatives which can be changed at any time. The reason for this flexibility is that the implementation alternatives can be analyzed without consuming system artefacts.

The implemented solution realizes the schedule generation-based solvers (serial SGS and parallel SGS) and extended schedule generation-based solvers such as the RCPSP heuristic solver and the multi-objective search-based solver. The resource manager is responsible for checking the feasibility of the schedules provided by the scheduling algorithms.

7. Results

The extended model and the designed scheduler have been implemented in our own SAP NetWeaver 7.5 developmental test environment. Since the model and algorithm do not contain SAP-specific elements, both can be mapped to suit any object-oriented programming language. The main reason for this decision is to enhance integration later using the SAP PM module directly on the SAP platform.

The implemented solution was tested on known basic problems and our numerical results compared with the known results of the test problems by applying two methodologies.

In the first case, problems were mapped using special boundary conditions onto the extended model, for which optimal scheduling algorithms are provided. It is

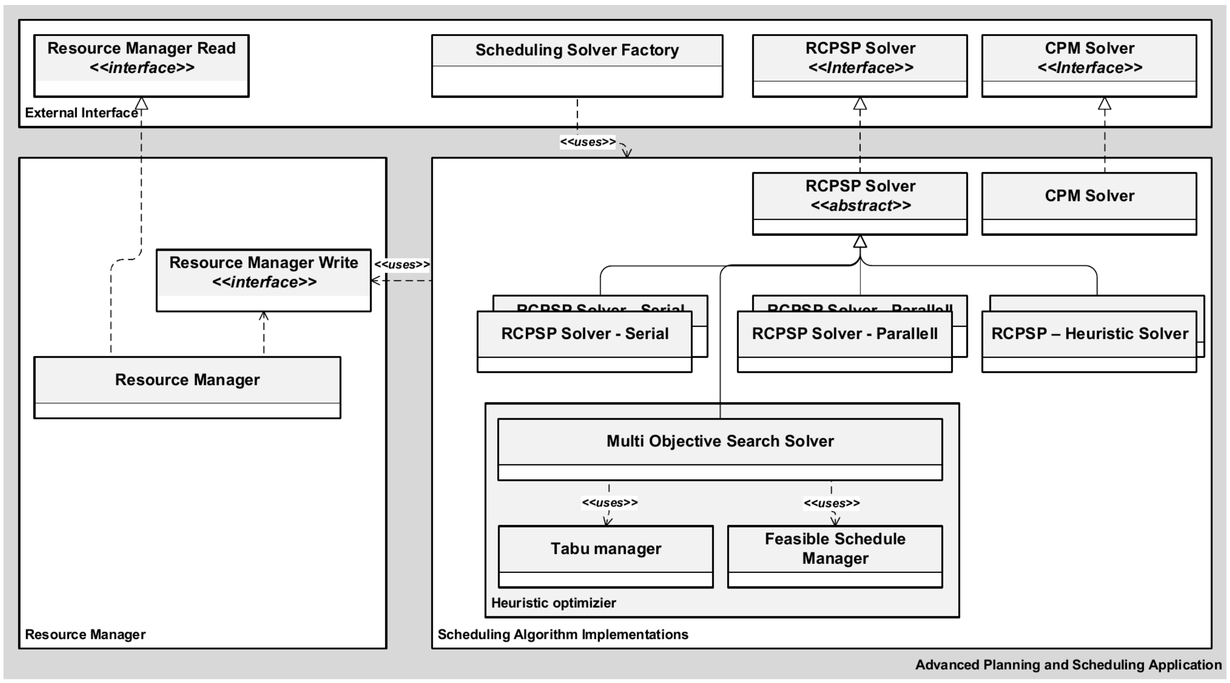

Figure 3: Project scheduler entities

known that Johnson's algorithm yields the optimal solution for permutation flow-shop scheduling problems with a makespan objective function [13]. In most cases ($> 95\%$), our heuristic-based solver identified the optimal solution to small problems (consisting of up to 20 jobs).

In the second case, the basic RCPSP was mapped onto the extended model and the results with regard to the objective function concerning the completion time of the project (Cmax) were compared with the best published results [4, 14]. Our results were compared with the best results from benchmark problems. It was observed that for small problems (consisting of up to 30 jobs), in most cases ($> 70\%$), the best known solution was identified by the scheduler.

8. Conclusions

Our research focused on modelling and solving an extended variant of resource-constrained, multi-objective, multi-project scheduling problems. The execution environment of the modelled project includes different resource types and their capacities, many projects with individual objective functions and precedence constraints, task-dependent individual processing times and resource requirements, tasks belonging to many projects, as well as project-dependent due dates.

During the research, new scheduling algorithms were developed that can flexibly solve problems while meeting all constraints. To generate detailed schedules, a predictive search algorithm was developed that includes an advanced simulation concerning the execution of the project. Multi-priority rule-based constructive algorithms were also developed by using schedule generation schemes embedded in the scheduling engine. To solve the extended problem, a new approach based on the combined usage of search and constructive methods was proposed.

In this paper, the proposed model of the investigated problem, the solution method and the key features of its implementation were described. The effectiveness of the proposed algorithms is demonstrated by presenting two validation methods.

The results show that the proposed approach is efficient and flexible. The developed model of the extended problem was formulated in such a way that the individual requirements of tasks and projects can also be taken into account. By considering several aspects simultaneously, vital practical requirements can be incorporated into the model. The management strategy as well as tactical and operational control policies can be implemented, moreover, logical decisions made by modifying the weights of aspects concerning decision-making, namely objective functions and rules.

Acknowledgements

This research was supported by the European Union and the Hungarian State, cofinanced by the European Regional Development Fund within the framework of the GINOP-2.3.4-15-2016-00004 project to promote cooperation between higher education and the industry.

REFERENCES

[1] Garey, M. R.; Johnson, D. S.: *Computers and intractability: A guide to the theory of NP-completeness.* (W. H. Freeman and Company, 1979) DOI: 10.1137/1024022

[2] Pritsker, A.; Allan, B.; Watters, L. J.; Wolfe, P. M.: Multiproject scheduling with limited resources: A zero-one programmingapproach. *Manage. Sci.*, 1969, **16**, 93–108 DOI: 10.1287/mnsc.16.1.93

[3] Mohanthy, R. P.; Siddiq, M. K.; Multiple projects multiple resources-constrained scheduling: Some studies. *Int. J. Prod. Res.* 1989, **27**, 261–280 DOI: 10.1080/00207548908942546

[4] Bálint, R.; Magyar, A.: Refrigerator Optimal Scheduling to Minimize the Cost of Operation. *Hung. J. Ind. Chem.* 2016, **44**(2), 99–104 DOI: 10.1515/hjic-2016-0012

[5] Deckro, R. F.; Winkofsky, E. P.; Hebert, J. E.; Gagnon, R.: A decomposition approach to multi-project scheduling. *Eur. J. Oper. Res.* 1991, **51**, 110–118 DOI: 10.1016/0377-2217(91)90150-T

[6] Jolayemi, J. K.: Scheduling of projects under penalty and reward arrangements: A mixed integer programming model and its variants.*Acad. Inf. Manag. Sci. J.* 2012, **15**, 29–52 ISSN: 1524-7252

[7] Gawiejnowicz, S.; Lin, B. M. T.: Scheduling time-dependent jobs under mixed deterioration. *Appl. Math. Comp.* 2010, **216**, 438–447 DOI: 10.1016/j.amc.2010.01.037

[8] Kim, K. W.; Yun, Y. S.; Yoon, J. M.; Gend, M.; Yamazaki, G.: Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling. *Comp. Ind. Eng.* 2005, **56**, 143–160 DOI: 10.1016/j.compind.2004.06.006

[9] Kumanan, S.; Jegan, J. G.; Raja, K.: Multi-project scheduling using a heuristic and a genetic algorithm. *Int. J. Adv. Manuf. Techn.* 2006, **31**, 360–366 DOI: 10.1007/s00170-005-0199-2

[10] Damak, N. J. B.; Siarry, P.; Loukil, T.: Differential evolution for solving multi-mode resource constraint scheduling problems. *Comp. Oper. Res.* 2009, **36**, 2653–2659 DOI: 10.1016/j.cor.2008.11.010

[11] SAP SE, SAP Project and Portfolio Management 2015 https://help.sap.com

[12] SAP SE, SAP S/4HANA 1709 FPS01, Project System (PS) documentation 2018 https://help.sap.com

[13] Johnson, D. B.: Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 1977, **24**(1), 1–13 DOI: 10.1145/321992.321993

[14] Kolisch, R.; Sprecher, A.: PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *Eur. J. Oper. Res.* 1996, **96**(1), 205–216 DOI: 10.1016/S0377-2217(96)00170-1