Production Systems and Information Engineering
Volume 10 (1), pp. 89–99
doi: 10.32968/psaie.2022.1.8.

MULTI-OBJECTIVE, MULTI-PROJECT SCHEDULING SOLVER IMPLEMENTATION USING SAP ABAP LANGUAGE

Krisztián Mihály
University of Miskolc, Hungary
Department of Information Engineering,
altmihaly@iit.uni-miskolc.hu

Mónika Kulcsárné Forrai
University of Miskolc, Hungary
Department of Information Engineering,
aitkfm@uni-miskolc.hu

Gyula Kulcsár
University of Miskolc, Hungary
Department of Information Engineering,
iitkgy@uni-miskolc.hu

Abstract. Projects became part of the daily business in many functional areas, across many industries and services. Despite of the variety of project execution implementation of the different domains it is visible that projects are sharing fundamental similiarities. Based on experiment the success of a project exuecution depends – next to maintained data quality and execution follow-up – on the ability to schedule the activities in an optimized way. Scheduling of the activities is even more complicated if an organization is executing more than one project at the same time. This paper presents our conceptual and implementation work in area of multi-project scheduling.

Keywords: Multi-project, multi-objective, project scheduling

1. Introduction

Project-based execution of product design, manufacturing, transportation, event organization or other main functionality of different line of businesses is common nowadays. The research domain of project scheduling has a solid and extensive background. The project scheduling topic has a well-known formalized problem that covers the two main interdependencies of the basic entities. This problem type is the resource-constrained project scheduling problem (RCPSP) which means one of the most important fundamental problems for project scheduling in the literature. Pritsker et al. (1969) developed a mathematical model for representing the RCPSP [1]. Blazewicz et al. (1983) have proved that the RCPSP is a strongly NP-hard problem [2]. RCPSP is used as a basic model, because it is a powerful model, but it cannot include all features of the situations that occur in practice. Therefore, many researchers have developed more general models for project scheduling problems. The classical RCPSP often are used as a starting model. In the literature, several survey papers on project scheduling have been published. Most of them focus on methods of the RCPSP model, for example [3], [4], [5], [6]. Many papers have summarized the main variants of the project scheduling problems, for example [7], [8], [9]. Schwindt and Zimmermann (2015) edited a handbook with two volumes [10], [10] that contain papers covering many important models and methods for project scheduling. In the literature, many other researchers have also published new extensions of RCPSP. For example, important overviews can be found in [11] and [12].

Definition of a project. A project can be defined from many different aspects. A widely used definition is provided by *ISO Standard 8402, 1994* [13]. This definition states that a project is an indivudual setup of coordinated and controlled system of processes, which has a defined start and end date, has a defined goal to achieve by using specified enviroment parameters, like cost, resource or deadlines. The resources often has certain limits to respect; a machine has a predefined maximum production capacity, a transportation has limits in speed and cargo size, a human can work up to 8 hours in a work day and so on.

Project scheduling. Project scheduling is required to create an execution plan based on known tasks and available environment parameters. The schedule is the created plan to achieve the predefined goals. Constructing a schedule is a complex task and it has a significant literature. Detailed reviews of the project scheduling topic can be found in [14].

Multi-objective, multi-project aspects. It is a common pattern in the pratice, that an organization is executing more than one project at the same time. The multi-project aspect was considered by other researches as well [15]. Each project has its own, individial setup regarding goals to achieve and resources to consume. Until resources or activities are not shared accross the projects, than each project can be scheduled and executed individually and independently. Once the projects are sharing resources or activities than the one project approach is hardly applicable. Mathematically a common start and end activity can be introduced to connect the projects to a *"super-project"*. The challenge of the super-project approach is to form a combined optimization function, which lead to an optimized solution on individual project level.

2. Modelling the problem

Definition of RCPSP. In the literature the *RCPSP – Resource Constrained Project Scheduling Problems* model is commonly used to describe one project with the constrainted scheduling environment. RCPSP model works with terms of task, resource types and constraints.

The set of tasks to be executed represents the activities to be completed in the project, for example an operation on a machine, an implementation task of a developer in a software engineering project and so on.

$$T = 1, 2, 3, ..., n \tag{2.1}$$

The set of resource types represents executor of the tasks, required to complete it. For example a resource type can be a group of machines or software developers of a company.

$$K = 1, 2, 3, ..., m \tag{2.2}$$

Capacity constraints for resource types, represent the individual limit of an available resource type. This value defines the quantity that can be allocated at the same time.

$$R_k, k \in K \tag{2.3}$$

Predecessor tasks represents the activities to be completed before a task can be started. For example quality check of a semi-finished product before painting can be started or review of software requirements before designing of a software can be started.

$$P_j, j \in T \tag{2.4}$$

Resource requirement represents per task the required resource types and the required capacity. For example one quality engineer and two inspector machines and is required execute a quality check.

$$(r_j, k) \tag{2.5}$$

Objective function(s) is (are) needed to evaulate the goodness of a feasible solution. A *feasible solution* is a schedule where all the given constraints are met. Two feasible solutions can be compared with evaulation of defined objective funtion(s).

Extended mathematical model. Based on RCPSP model an enhanced model has been developed. The model inherits the terminologies of the RCPSP model, such as task, resource types and constraints. As an enhancement more than one project can be defined at the same time and the individual tasks can be assigned to one or more projects. Each individual project has own assigned objective functions. Additionally further scheduling constraints on task or project level can be defined, for example definition of earliest start time for a task or a project.

3. Solving concept

The RCPSP problem belongs to NP hard problem [16]. Characteristic of NP hard problem has effect on applicable solver concept. For small problem size brute-force based implementations may work, but for pratical problem size it is not applicable. To deal with the problem in the literature different approaches are used.

3.1. Approaches

In this sub-section two mainly used approaches are briefly summarized.

Generation scheme solvers. The main principle of the generation scheme-based scheduling is to start from an empty schedule and in each iteration one egilible task is selected from the set of non-scheduled tasks. In the iteration step when the next task is selected all the given constraint is taken into consideration; all prerequisite task was scheduled and all resource requests can be completed simultaneously. The selected task is added to the schedule considering all resource constraints. The construction is finished when all tasks are scheduled. Generation scheme solvers has two main variants: serial- and parallell generation scheme.

Search-based solvers. Search-based solvers are working with a set of predefined rules to create new solution candidates. The candidate generation is followed by a simulation step. During the simulation the candidate is used to calculate the objective function values and calculate the key performance indicator values. If the generation rules can generate candidates which are not by-design feasible, then simulation can be terminated and the candidate is rejected. After the simulation the candidates are compared and based on the solver algorithm new candidates are generated. The generation and simulation steps are continued until a certain exit criteria is met and the best candidate is returned as solution.

3.2. A new combined approach

Our approach combines the generation scheme solvers and the search-based solvers. A generation scheme solver is used as simulator by a search-based solver. The search-based solver is injecting generated search parameters for the generation scheme, to influence the behavior of the task selection step externally. This approach ensures that generation scheme always creates a feasible schedule. The search-based solver can combine different candidates based on generated, feasible solutions.

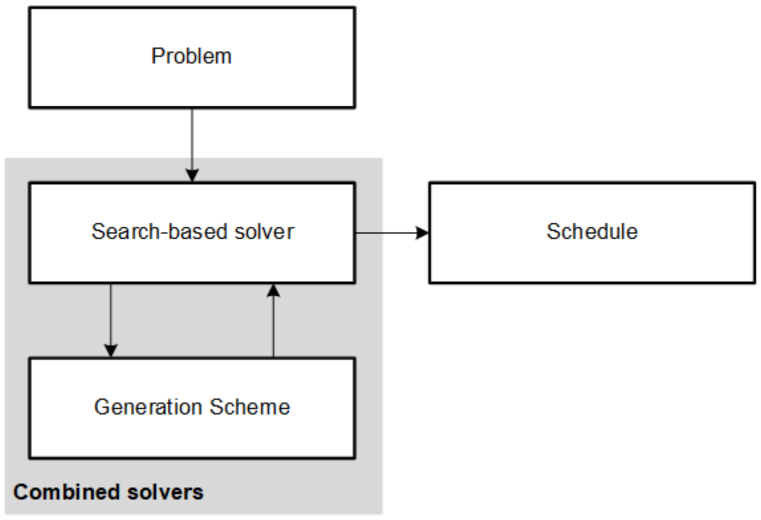


Figure 1. Combined approach overview

4. Implementation concept

4.1. Main design goals

The main desing goals on implementation level were to enable the extensibility, the maintaineability, the testability and the combination of modules. ABAP programming language was used in the implementation and in the design phase the following design driving factors were identified:

- use object-oriented modelling,
- clear separation of concerns,
- injection or changeability of dependencies,
- by design supported unit testing and integration testing,
- enable proof of concept work.

4.2. Compositional structure

The high-level overview depicted in Figure 1 and it is refined in Figure 2.

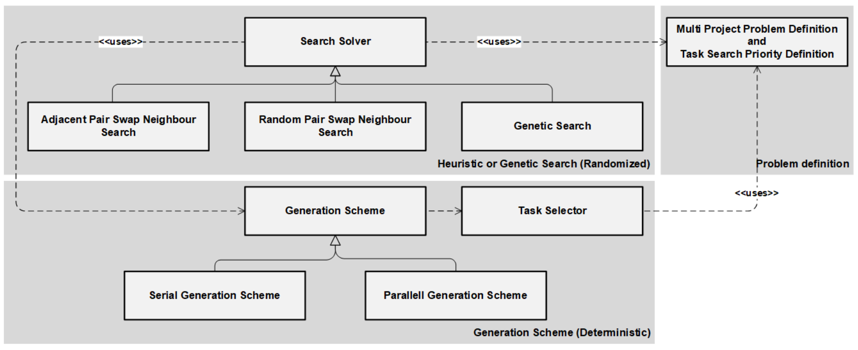

Search Solver
<<uses>>
<<uses>>
Multi Project Problem Definition and Task Search Priority Definition
Adjacent Pair Swap Neighbour Search
Random Pair Swap Neighbour Search
Genetic Search
Heuristic or Genetic Search (Randomized)
Problem definition
<<uses>>
Generation Scheme
Task Selector
Serial Generation Scheme
Parallell Generation Scheme
Generation Scheme (Deterministic)

Figure 2. Implementation main building block detailed view

The search-based solver uses a generation scheme as dependency to be used as a simulator. The implementation alternative of concrete generation scheme variant is hidden for the search. The multi-proect scheduling project definition is modelled as an own object and it can be accessed as a dependency.

Modelling of generation scheme. The generation solver is accessible for consumers as an ABAP interface. The interface defines the generation scheme contract and it is containing the accessible services, for example request for a solution for a problem and injection of search controlling dependencies, like realization of task selection logic. The generation solver can have multiple implementations. The serial and parallell generation solver has significant overlap in their algorithm, the common code is implemented in an abstract class. The variant specific implementation is implemented by dedicated realization classes, where specialities is implemented with modelled delegation pattern. The task selection logic is modelled as an individual object and contract, enabling the changeability of different heuristics or algorithms. As an example the Minimal Slack or Latest Finish Time heuristics are implemented. To offer control via externally defined heuristic parameters for search solvers the *Search Solver Defined* selection class is implemented.

Modelling of search-based solver. Search-based solvers are accessible for consumers as an ABAP interface. The interface defines the contract for search-based solvers. The interfaces contains the signature of solving an RCPSP problem and the signature is defining the dependencies as well. Different kind of search-based solvers are grouped into main types in the Figure 4. The common algorithm code can be implemented in abstract classes; for example the

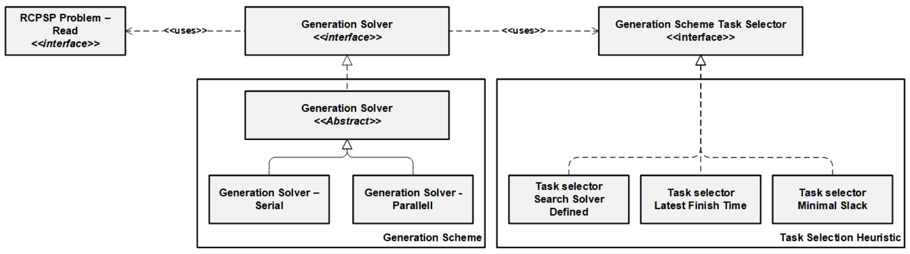

RCPSP Problem – Read <<interface>>
<<uses>>
Generation Solver <<interface>>
<<uses>>
Generation Scheme Task Selector <<interface>>
Generation Solver <<Abstract>>
Generation Solver – Serial
Generation Solver - Parallell
Generation Scheme
Task selector Search Solver Defined
Task selector Latest Finish Time
Task selector Minimal Slack
Task Selection Heuristic

Figure 3. Generation Scheme Implementation Level

heuristic search based implementations have a main frame and each implementation has its own, specific implementation of applied modification operations. The specialization is implemented usually via delegation pattern.

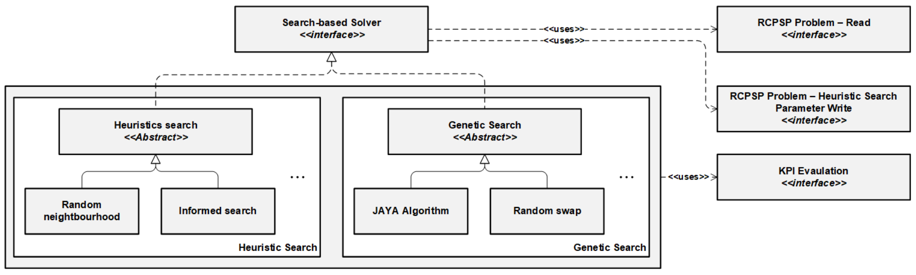

Search-based Solver <<interface>>
<<uses>>
<<uses>>
RCPSP Problem – Read <<interface>>
RCPSP Problem – Heuristic Search Parameter Write <<interface>>
Heuristics search <<Abstract>>
Genetic Search <<Abstract>>
...
...
<<uses>>
KPI Evaulation <<interface>>
Random neightbourhood
Informed search
JAYA Algorithm
Random swap
Heuristic Search
Genetic Search

Figure 4. Search-based Solver Implementation Details

4.3. Advantage of modularity

The design of the solver is based on ABAP interfaces with well defined contract and separation of responsibilty. The dependencies are modelled always as an interface. The concrete implementation of the dependency can be injected in the instantation of an implementation (constructor inject) or the dependency is part of request signature itself. With this approach the implementation is

- enabled for unit testing on fine granular unit level,
- interchangeable without major refactor of the implementation,
- open for new implementation alternatives,
- capable to define combination of algorithms,
- prepared for teaching purposes, where students can try out implementations.

The dependency injection pattern makes it possible to try out search-based implementations with different task selection options and generation scheme alternatives.

Table 1. Combination of modules

Module	**Implemented alternatives**
Generation scheme solver	Serial, Parallell
Task selection heuristic	Minimal Slack, Latest Finish Time
Heuristic Search	Neighourhood, Random swap, Critical path swap
Genetic Search	Population-based search without recombination, Random cross-over

5. Implementation results

The concept has been implemented in ABAP language using the ABAP object-oriented language elements. The scheduler has a well defined application programming interface (API) and it is accessible via ABAP objects. For testing purpose an ABAP GUI based administrator user interface has been developed. In the Figure 5 a testing application UI is depicted. The user can

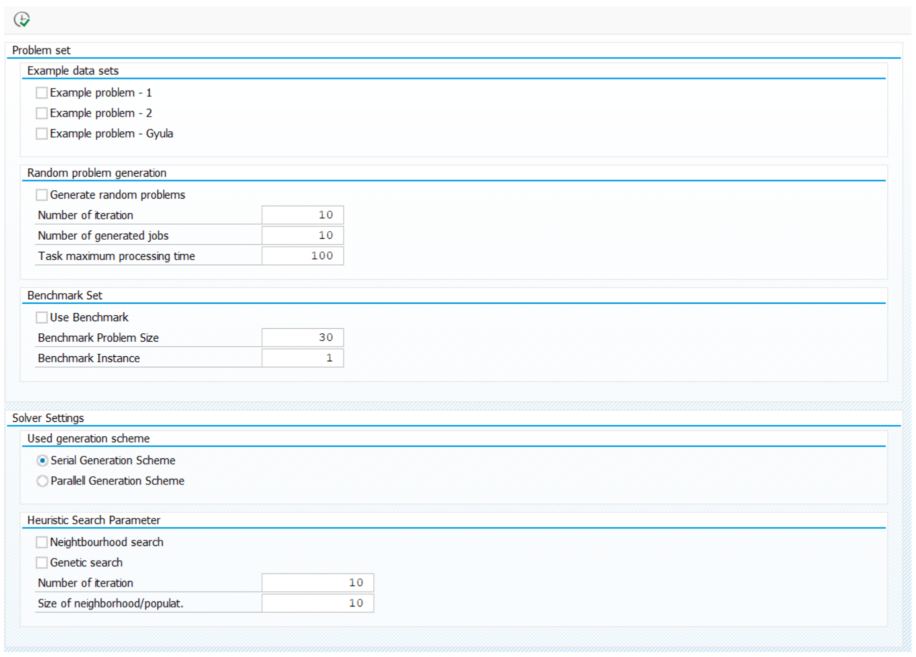
- select wich problem set to be used,
- generate random problems,
- calibrate the search modules (number of iteration, size of population, etc.),
- combine search-modules and generation scheme alterantives,
- combine task selection alternatives.

5.1. Functional correctness

Functional correctness of the implementation was tested with usage of unit tests, mapping of known optimal algorithm to RCPSP problem and solving RCPSP benchmark problems with known optimum solution.

Unit tests are implemented as ABAP unit tests to ensure functional correctness of a module [17]. The unit test is useful to try out the module with different known request – response combinations. Unit tests were used for internal modules like critical path searching, correctness of task selection, resource booking management and so on. ABAP unit test can be executed in each source code change and it helps to prevent regression if any correction is required in a module.

Mapping of less complex problems to RCPSP was used to do integration test. Well known permutation flow shop problems with 2 machines (F2||Cmax) were mapped to RCPSP. The result of the scheduler was compared with the result of the Johnson-algorithm [18].



Problem set

Example data sets

☐ Example problem - 1

☐ Example problem - 2

☐ Example problem - Gyula

Random problem generation

☐ Generate random problems

Number of iteration	10
Number of generated jobs	10
Task maximum processing time	100

Benchmark Set

☐ Use Benchmark

Benchmark Problem Size	30
Benchmark Instance	1

Solver Settings

Used generation scheme

● Serial Generation Scheme

○ Parallel Generation Scheme

Heuristic Search Parameter

☐ Neightbourhood search

☐ Genetic search

Number of iteration	10
Size of neighborhood/populat.	10

Figure 5. Selection screen of solver

RCPSP benchmark problems were mapped to our model and the results were compared with the known and published benchmark results [4], [19], [20].

6. Conclusion and future work

In this paper, we summarized the results of our research focused on modelling and solving of multi-project scheduling problem taking into account extended constraints and multiple objective functions. The detailed characteristics of the problem were presented. A new model was proposed that includes the set of projects to be scheduled, the resource-constraints, the advanced features of the activities and the characteristics of the execution environment. The proposed model supports the flexible usage of many objective functions based on project-dependent arguments.

We introduced a new combined scheduling approach to solve the problems flexibly while all the constraints were met. The proposed solving method uses a predictive searching algorithm and an advanced simulation algorithm. The simulation uses different constructive generation schemes based on priority rules. The most efficient and flexible version of the solving algorithm variants was the combined method in which the population-based search algorithm defines the control priorities of the generation scheme based simulation.

The framework of the solver engine was developed with object-oriented methodology and implemented by using ABAP language. The implementation supports the usage of different concrete algorithms variants and testing the algorithms on different type of benchmark instances. Based on the running results of the benchmark tests, we concluded that the proposed solution concept is sufficiently efficient and flexible to apply even to medium and large size problems. The best results are generated an advanced population based genetic algorithm in which random mutation operator was used and the crossover operator was not applied. This algorithm calibrates the control priority of each activity of the projects and the modified serial generation scheme created the complete solution based on the control priorities of the activities.

The proposed model and methods can support the consideration of individual requirements of activities and projects in practice. Our approach is independent from the optimization goals. Consequently, many objective functions can be taken into account simultaneously. The results of the research allow a wide range of project scheduling problems to be solved in practice. Next step is to implement additional data model transformation modules to enable the solver to work with different project management systems. The implemented modularized architecture enables the solution to be used as training environment for students learning programming and project or manufacturing scheduling.

References

[1] Pritsker, A. A. B., Waiters, L. J., and Wolfe, P. M.: Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, **16**(1), (1969), 93–108.

[2] Blazewicz, J., Lenstra, J. K., and Kan, A. R.: Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, **5**(1), (1983), 11–24.

[3] Hartmann, S. and Kolisch, R.: Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European journal of operational research*, **127**(2), (2000), 394–407.

[4] Kolisch, R., Schwindt, C., and Sprecher, A.: Benchmark instances for project scheduling problems. *Project scheduling: recent models, algorithms and applications*, pp. 197–212, URL https://doi.org/10.1007/978-1-4615-5533-9_9.

[5] Kolisch, R. and Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, **174**(1), (2006), 23–37.

[6] Pellerin, R., Perrier, N., and Berthaut, F.: A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, **280**(2), (2020), 395–416.

[7] Brucker, P.: Scheduling and constraint propagation. *Discrete applied mathematics*, **123**(1-3), (2002), 227–256.

[8] Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, **112**(1), (1999), 3–41.

[9] Tavares, L. V.: A review of the contribution of operational research to project management. *European Journal of Operational Research*, **136**(1), (2002), 1–18.

[10] Schwindt, C., Zimmermann, J., et al.: *Handbook on project management and scheduling vol. 1*. Springer, 2015.

[11] Hartman, S. and Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem cc: 000. *Operations Research Management Science*, **51**(1), (2011), 67.

[12] Hartmann, S. and Briskorn, D.: An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, **297**(1), (2022), 1–14.

[13] Iso 21500:2021(en): Project, programme and portfolio management — context and concepts. URL https://www.iso.org/standard/75704.html. Accessed: 2022-06-06.

[14] Mohanty, R. u. and Siddiq, M.: Multiple projects-multiple resources-constrained scheduling: some studies. *The International Journal of Production Research*, **27**(2), (1989), 261–280, URL https://doi.org/10.1080/00207548908942546.

[15] Kumanan, S., Jegan Jose, G., and Raja, K.: Multi-project scheduling using an heuristic and a genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, **31**, (2006), 360–366.

[16] Garey, M. R. and Johnson, D. S.: *Computers and intractability a guide to the theory of NP completeness*. Springer, 1979, URL https://doi.org/10.1137/1024022.

[17] McDonough, J. E. and McDonough, J. E.: *Automated Unit Testing with ABAP*. Springer, 2021, URL https://doi.org/10.1007/978-1-4842-6951-0_5.

[18] Johnson, S. M.: Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, **1**(1), (1954), 61–68, URL https://doi.org/10.1002/nav.3800010110.

[19] Sprecher, A. and Kolisch, R.: Psplib—a project scheduling problem library. *Eur. J. Oper. Res*, **96**, (1996), 205–216.

[20] Sprecher, A.: Project scheduling problem library. URL http://www.om-db.wi.tum.de/psplib/library.html. Accessed: 2022-06-06.