Comparison of Three Solution Encodings and Schedule Generation Scheme for the Multi-Site RCPSP

A. Laurent, L Deroussi, Sylvie Norre, Nathalie Grangeon

Comparison of Three Solution Encodings and Schedule Generation Scheme for the Multi-Site RCPSP

A. Laurent [*] **L. Deroussi** [**] **N. Grangeon** [***] **S. Norre** [****]

[*] *Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes, Campus Universitaire des Cézeaux, 1 rue de la Chebarde, 63178 AUBIERE, FRANCE (e-mail: arnaud.laurent@isima.fr).*
[**] *(e-mail: laurent.deroussi@uca.fr)*
[***] *(e-mail: nathalie.grangeon@uca.fr)*
[****] *(e-mail: sylvie.norre@uca.fr)*

Abstract: In this paper we propose to study an extension of the classical RCPSP called the Multi-site RCPSP. This extension allows to take into account transportation times (both for resources and tasks) between several sites and has some medical or industrial applications for instance. We design an approximate resolution method in which three solution encodings are proposed. Each of them is coupled with a dedicated list algorithm which build feasible schedules. The results obtained on a benchmark inspired by literature are presented and discussed. The methods are also compared with a mathematical model.

Keywords: RCPSP, Multi-Site, Metaheuristic, Solution Encoding, List algorithm, Neighborhood System.

1. INTRODUCTION OF THE MULTI-SITE RCPSP

Resource pooling management for multi-site organisations receives an increasing interest from decision workers. Resources are shared between several sites and have the possibility to move from one site to another. To model this situation, we have proposed a new model, the Multi-Site RCPSP.

In the classical Resource Constraint Project Scheduling Problem (RCPSP) a set of N tasks has to be scheduled. The duration of the task j is p_j. Each task has a set P_j of precedence relations. Each task j needs a set of $r_{j,k}$ resources of type k, for each K types. Mathematical models have been proposed by Oğuz and Bala (1994) and Correia et al. (2012). A lot of paper deal with this classical problem and its extensions.

Because of the multi-site context, new characteristics which are not considered in the classical RCPSP have to be used. We first introduce the notion of site. Each task needs a site to be performed. The second concept is the distinction between fixed resources and mobile resources. For example, a fixed resource can be a machine that can not be moved. Thus, its use for a task will determine the site where the task will be assigned. A fixed resource can not be assigned to a task performed on a site where the resource is not located. In contrast, a mobile resource can execute tasks on every site.

If a mobile resource executes two consecutive tasks on two different sites, time constraints must be considered to model the transportation time of the resource from one site to another. Mobile resources are available on the site where they realized their first task. A transportation time results in a minimum delay between the end of the execution of the first task and the beginning of the execution of the second task. This time depends on the pair of sites where both tasks are performed. There is another case in which a transportation time is applied: when two tasks are linked by a precedence relation and are realized on different sites. In this case, it represents the transportation time of a semi-finished product between two sites.

This problem extends the definition of the RCPSP. If only one site is considered for an instance of the Multi-Site RCPSP, the remaining problem is a classical RCPSP. As the RCPSP is NP-hard in the strong sense, this extension is NP-hard too.

The goals of this paper is to propose and compare three different solution encodings to solve the Multi-Site RCPSP. We will determine if there is a more effective encoding, or in which case an encoding will perform better than the others.

In this paper we first present a quick state of the art about RCPSP extensions related to transportation times. Then we present the proposed proposes three different solution encodings to solve the Multi-Site RCPSP. Finally we present some results and discuss about the interests of the encodings.

2. TRANSPORTATION TIME IN RCPSP

The classical RCPSP, written $PS|prec|C_{max}$ by Kan (1976), consists in scheduling each task and assigning resources to it. A version of this problem exists with several modes of execution for each task. A mode define a resource

requirement and completion time to execute a task. This problem is called MRCPSP for Multi-mode RCPSP.

Two extensions of the RCPSP and three extensions of the MRCPSP are related to our problem:

- RCPSP with minimum time lags (RCPSP min) Klein (2000)
- RCPSP with conditional minimum time lags (RCPSP-CTL) Toussaint (2010)
- MRCPCP with generalized precedence relations (MRCPSP-GPR) De Reyck and Herroelen (1999)
- MRCPSP with Mode Dependent Time Lags Sabzehparvar and Seyed-Hosseini (2008)
- MRCPSP with schedule dependent set-up time (MRCPSP-SST) Mika et al. (2004)

Each RCPSP extension can model transportation times. The RCPSP with minimum time lags considers a time between the end of a task and the beginning of another task. This time is known and given for each pair of tasks. The RCPSP-CTL also considers resource transfer with transportation time known for each couple of tasks. The MRCPSP-GPR considers mode-dependant time lags. In a multi-site context, the mode can represent the site where a task is assigned and the time lag will be the transportation time of the semi-finish product to the next site. But in the MRCPSP-GPR, no transportation time is considered for resources. In the MRCPSP-SST each task needs a site to be executed and the time lag for a couple of tasks is dependent of the site assignment.

Some of these extensions can model transportation time of semi-finish products or resources transportation. However, none of these extensions models both aspects of multi-site context: the site assignment for tasks and the resources and tasks transportation. This is why the Multi-site RCPSP has been proposed Laurent et al. (2015).

3. RESOLUTION METHOD

As this problem is NP hard in the strong sense, approximate methods form a good alternative for average-to-large size instances. Among them, metaheuristics are a family of generic methods that are considered to be efficient for solving hard optimisation problems. We propose to implement individual-based metaheuristics. First we describe the three solution encodings used for this problem and the corresponding neighborhood system. Then, we define the corresponding list algorithms that schedule the solution depending on the encoding.

3.1 Solution encodings

We added two fictitious tasks to the project, one will represent the begining of the project and the other one will represent the end. These two tasks have a duration of 0 period. The beginning task precede every tasks and every tasks precede the ending task. We propose three different solution encodings based on the three following components:

- $\sigma = (\sigma_1, \sigma_2, ..., \sigma_N)$ is a permutation of the N tasks describing a topological list in which precedence relations will always be respected. σ_1 and σ_N are fictitious project tasks, representing respectively the start and the end of the project. Resources will execute tasks in the order given by σ.
- $l = (l_2, l_3, ...l_{N-1})$ with $l_j \in \{1, S\}$ the site among the S sites where the task $j \in \{1, N\}$ will be executed. This assignment must respect the amount of resources available on the site, which is equal to the sum of fixed resources on the site and the mobile resources.
- $a = (a_2, a_3, ...a_{N-1})$ with a_j the set of resources assigned to the task $j \in 2, N-1$.

The three solution encodings are:

(σ, l, a) is the most complete encoding, with the resources and site assignment and the topological list for tasks.
(σ, l) which does not define the resources assignment as (σ, l, a)
(σ) which has only the topological list

3.2 Schedule Generation Scheme (SGS)

The principle of SGS introduced by Kelley et al. (1963) is to schedule a solution based on a sequence of tasks.

We define three SGS, one for each encoding.

- For the encoding (σ, l, a): the algorithm schedules tasks as soon as possible. In the σ order, it determines the earliest availability date for each resource assigned to a task. This availability date for a resource is computed depending on the end of the last task executed and the sites assigned to the two tasks. The task will be scheduled at the maximum date of all resources availabilities or ending date of tasks with precedence relation plus transportation time.

- For the encoding (σ, l): the algorithm determines the assignment of resources to tasks. To do that, in the order σ, it selects the earliest available resources needed on the site l_j to execute task j. When the resources have been determined, the task is schedule according to the same rules that for (σ, l, a) list algorithm. The algorithm 1 represent the SGS for this solution encoding.

- For the encoding (σ): the algorithm assigns tasks to sites and assigns resources to tasks. To assign tasks to sites, the algorithm applies the (σ, l) list algorithm for each possible site, for each task in the σ order. The site with the earliest date is chosen.

Algorithm 1 List-scheduling based algorithm for the σ, l encoding

Require: $X = (\sigma, l)$
  for $j = \sigma_1$ to σ_N **do**
    Selection of the earliest available resource
    Assignment of the selected resources to the task j
    Computation of the completion time for task j
  end for

Each SGS associated to a solution encoding forms a search space $\Omega_{\sigma,l,a}$, $\Omega_{\sigma,l}$ and Ω_{σ}. To our knowledge, there is no polynomial SGS for (σ) and (σ, l) that ensures the existence for every instances of an optimal solution in $\Omega_{\sigma,l}$ and Ω_{σ}.

3.3 Neighborhood system

The neighborhood system used for the different encodings is composed of three basic moves. The first one is dedicated to the sequence of tasks, the second one concerns the site assignment l and the last one concerns the resource assignment. At each iteration, each move is randomly chosen. Let us describe these moves.

Insertion of the tasks The first move modifies the tasks by applying an insertion move. A task is moved from a position p to another one p' ($p' \neq p$). A move is feasible if the resulting sequence of tasks satisfies the precedence constraints. This move is implemented such that only the feasible moves are considered. This means in particular that the starting and the ending project tasks stay in the first and last position. See algorithm 2.

Algorithm 2 Algorithm of the insertion move

Require: σ : initial sequence
  while Not Stop **do**
    undo the moves;
    Choose two random positions $p \in [2, N-1]$ and $p' \in [2, N-1] / p' \neq p$;
    if $p < p'$ **then**
      $\sigma' = \sigma_1, ...\sigma_{p-1}, \sigma_{p+1}, ..., \sigma_{p'}, \sigma_p, ..., \sigma_N$
    else
      $\sigma' = \sigma_1, ..., \sigma_{p'}, \sigma_p, ...\sigma_{p-1}, \sigma_{p+1}, ..., \sigma_N$
    end if
    if σ' is feasible **then**
      Stop
    end if
  end while

Site-assignment move The second move modifies the site assigned to a task j. Like for the previous one, this move preserves the feasibility of the solution. For each type of resource, the sum of the fixed resources located in the new site and the mobile resources must be greater than the amount of resources required for executing the task j. The principle of this move is given by algorithm 3.

Algorithm 3 Algorithm of site-assignment move

Require: l : initial site-assignment
  Choose a task $j \in 2, N-1$;
  Choose a compatible site;
  $l' \leftarrow l_2, ..., lj-1, l'_j, lj+1, ..., l_{N-1}$;

Resource reassignment The third move substitutes a resource for a task j by another one. This move selects a task and removes a resource assignment. Another resource of the same type but different from the previous one is assigned to the task. The chosen resource must be compatible, this mean that either the resource is fixed and it is necessary on the site, or the resource is mobile. The principle of this move is given by algorithm 4.

Algorithm 4 Algorithm of resource reassignment move

Require: a : initial assignment;
  Choose randomly a set of resource a_j
  Replace a resource r in the set a_j by a resource $r' \notin a_j$ and compatible

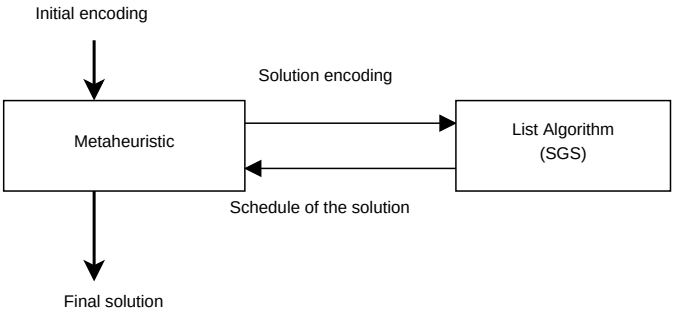

Fig. 1. Principle of the coupling

3.4 Metaheuristic

To solve this problem we use a metaheuristic to modify the solution with the neighborhood system. The principle of the coupling is presented in figure 1. At each iteration, a solution encoding is created by the application of a neighborhood system on the current solution. Then the solution encoding is evaluated by the list algorithm. This solution is now compared to the current one and selected depending on the acceptance criterion.

Algorithm 5 Algorithm of the Iterated Local Search (ILS)

Require: X_0 : initial solution
  $X^* \leftarrow$ local search on X_0;
  while Stopping criterion is not satisfied **do**
    $X' \leftarrow$ Perturbation of X^*;
    $X*' \leftarrow$ Local search on X';
    $X^* \leftarrow$ Acceptance criterion of $X*'$ to X^* taking into consideration the history;
  end while
  return X^*

To test these coupling we use the Iterated Local Search (ILS) proposed by Lourenço et al. (2003). The principle of the ILS (Algorithm 5) is to run many local searches. When a local minimum is obtained from the first local search, a new solution is build by perturbation of the local minimum. A new local search is applied on the new solution. The two local minima are compared and one of them is selected depending on the acceptance criterion (history). The process is iterated until the stopping criterion is reached. There is some parameters to fix to use this metaheuristic:

- The stopping criterion: we will use two : 100K iterations in total for small instances and 10 minutes of computation time for the larger ones.
- The acceptance criterion: we use the Metropolis acceptance criterion Metropolis et al. (1953). This criterion is also known as the simulated annealing acceptance criterion. The probability to accept a lower quality solution decreases with the time.
- The perturbation: the perturbation used is 4 times the neighborhood system.

4. RESULTS

The purpose of this section is to test the effectiveness of the proposed encodings on the new problem.

Encoding	σ	σ, l	σ, l, a	MM	% completed
10 tasks	26,45	26,5	**26,125**	26	100
15 tasks	**36**	36,45	36,22	36	100
20 tasks	57,75	**57,5**	58,9	55,5	100
25 tasks	70,98	**70,38**	72,23	68,25	100
30 tasks	**79,15**	82,125	90,725	X	10
2 sites	**51,98**	52,08	54,99	X	84
3 sites	**56,11**	57,11	58,69	X 80	
10 resources	**55,53**	55,56	56,27	84	
20 resources	**52,56**	53,63	57,41	80	
TCT	2240,45	1110,0	**400,19**	X	

Table 1. Average makespan obtained on the small instances

First of all we test these methods on small instances and compare the results obtained to the ones given by the mathematical model [Laurent et al. (2015)]. We have generated a set of 160 instances. An instance is composed of 10 to 30 tasks, 2 or 3 sites and 10 or 20 resources in total. For a set of instance, there is 8 instances with the same number of resources and the same number of sites. Ten replications are run on each instance. The goal is to compare the performance of the different encodings depending on the number of tasks, the number of resources and the number of sites. For each method, 10 replications are run with a maximum number of iterations (stopping criterion) fixed to 100 000. The initial solution for every replication is build randomly. The mathematical model (MM) are stopped after 30 minutes if an optimal solution is not found. For the 30 tasks instances, no solution is found with the mathematical model. This is why no value are given for the 30 tasks instances, all the 2 or 3 sites instances and all the 10 or 20 resources instances. For each set of instances with the same parameters, the average makespan is given. This average is done one the 8 different instances with 10 replications. The Total Computation Time (TCT) is also given for each method. The total computation time is the sum of all computation times for a given method to solve every instances.

The results given by table 1 show that the performance of the methods depend on the size of the instances. The Second thing is that the result of σ and σ, l encodings are close to the ones given by the mathematical model. Finally, the computation time of the different encodings system is not equivalent, depending on the complexity of the list algorithm. This is why in the next experimentations we will compare the results depending on the computation time.

In this second part, to test the different encodings depending on the computation time, we adapt literature instances of the PSPLIB [Kolisch and Sprecher (1997)]. In this paper we only use the 30 tasks set of the library. We add to these instances 2 or 3 sites. Each resource has a probability of 50%to be fixed to a random site, otherwise the resource is considered as mobile. We set transportation times between each site with a value in the same range as the duration of the tasks. There is 480 instances of 48 different classes. We run each instances 10 times, with a maximum computation time of 10 minutes.

The results on the 30 tasks instances are reported in the table 2. The presented results are the average makespan of the best solution found by each encoding according to the execution time.

Time(s)	0	10	60	150	300	600
(σ, l, a)	208,9	103,6	96,1	92,1	87,7	85,1
(σ, l)	138,2	80,1	73,9	**72,4**	**71,15**	**70,5**
(σ)	**95,9**	**75,5**	**72,9**	**72,4**	71,9	71,6

Table 2. Average makespan obtained on the 30 tasks instances

In the table 2, the best average makespan for a given computation time is highlighted. If we compare the results obtained by the different methods, we can observe the following results:

- After 10 minutes, the (σ, l) encoding gives the best results and (σ, l, a) the worse.
- The (σ, l, a) gives good results on small instances, but for larger instances, the other encoding seems a better alternative. For 30 tasks instances the (σ, l, a) encoding seems to not have converge yet after 10 minutes. This can be explained by the largest search space.
- The (σ, l) encoding gives better results than (σ) encoding after 150 seconds.

According to these results, the σ encoding could be more relevant to use on larger instances (number of tasks, sites and resources). In the other hand, on small instances, σ, l will give better results if the computation time is long enough. The use of these methods will depend on the available time of computation and the size of the instance.

5. CONCLUSION AND PERSPECTIVES

To conclude, we propose in this article a resolution method, using three solution encodings. The Multi-Site RCPSP is a new and complex problem with transportation of resources and tasks allocations to site. With these three different solution encodings, we try to determine which solution encodings has the more advantages. The first results on small instances show that the (σ, l, a) encoding will not be exploitable on this form. The (σ, l) encoding gives the best results when enough CPU time is available, but the (σ) encoding converges faster to the best solution that it can found. These two methods offer a compromise between speed and performance to find a good solution for the Multi-Site RCPSP .

One of our highest priority is to test our method on largest instances to see the relevance of the (σ, l) encoding compared to the (σ) encoding.

REFERENCES

Correia, I., Lourenço, L.L., and Saldanha-da Gama, F. (2012). Project scheduling with flexible resources: formulation and inequalities. *OR spectrum*, 34(3), 635–663.

De Reyck, B. and Herroelen, W. (1999). The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2), 538–556.

Kan, A.R. (1976). *Machine scheduling problem: Classification , complexity and computations*. Martinus Nijhoff.

Kelley, J.L., Namioka, I., Donoghue Jr, W.F., Lucas, K.R., Pettis, B., Poulsen, E.T., Price, G.B., Robertson, W.,

Scott, W., and Smith, K.T. (1963). *Linear topological spaces*. Springer.

Klein, R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research*, 38(16), 3937–3952. doi:10.1080/00207540050176094. URL http://www.tandfonline.com/doi/abs/10.1080/00207540050176094.

Kolisch, R. and Sprecher, A. (1997). {PSPLIB} - a project scheduling problem library: {OR} software - {ORSEP} operations research software exchange program. *European Journal of Operational Research*, 96(1), 205 – 216. doi:http://dx.doi.org/10.1016/S0377-2217(96)00170-1. URL http://www.sciencedirect.com/science/article/pii/S0377221796001701.

Laurent, A., Deroussi, L., Grangeon, N., and Norre, S. (2015). Multi-site rcpsp: Notations, mathematical model and resolution method. In *CIE45, International Conference on Computers & Industrial Engineering*.

Lourenço, H.R., Martin, O.C., and Stützle, T. (2003). *Iterated Local Search*, 320–353. Springer US, Boston, MA. doi:10.1007/0-306-48056-5_11. URL http://dx.doi.org/10.1007/0-306-48056-5_11.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6), 1087–1092.

Mika, M., Waligora, G., and Weglarz, J. (2004). A metaheuristic approach to scheduling workflow jobs on a grid. In *Grid resource management*, 295–318. Springer.

Oğuz, O. and Bala, H. (1994). A comparative study of computational procedures for the resource constrained project scheduling problem. *European Journal of Operational Research*, 72(2), 406 – 416. doi:http://dx.doi.org/10.1016/0377-2217(94)90319-0. URL http://www.sciencedirect.com/science/article/pii/0377221794903190.

Sabzehparvar, M. and Seyed-Hosseini, S.M. (2008). A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *The Journal of Supercomputing*, 44(3), 257–273.

Toussaint, H. (2010). *Algorithmique rapide pour les problèmes de tournées et d'ordonnancement*. Ph.D. thesis, Université Blaise Pascal-Clermont-Ferrand II.