Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 164 (2019) 478–484

Procedia Computer Science

www.elsevier.com/locate/procedia

CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies

A new approach for the multi-site resource-constrained project scheduling problem

Cyrine Stiti[a,*], Olfa Belkahla Driss[b]

[a]*Institut Supérieur de Gestion de Tunis, Université de Tunis, Tunisia*
[b]*Ecole Supérieure de Commerce de Tunis, Université de la Manouba, Tunisia*

Abstract

In recent years, the study of Resource Constrained Project Scheduling Problem RCPSP which as a well-known NP-hard problem in project management and operational research is taking substantial interest. The present work investigates a new RCPSP extension in multi-site environments with resource pooling between different sites. This extension considers two types of resources: fixed and mobile resources. New constraints are taken into account like transfer times of mobile resources and the decision on which site a task is carried out. We propose a new Particle Swarm Optimization-based solution to deal with this RCPSP variant aiming at minimizing the total project Makespan. The proposed algorithm uses a Valid Particle Generator to produce feasible schedules. As well, inertia weight is adaptively tuned to realize a trade-off between diversification and intensification. Our solution has been tested on a set of newly created benchmarks to reveal very promising results.

Keywords: Multi-site, RCPSP, Scheduling, Transfer-time, PSO, Swarm Intelligence, metaheuristic, Factory 4.0.

* Corresponding author. Tel.: +216-52742793.
E-mail address: cyrine.stiti@gmail.com

1. Introduction

RCPSP has become the most studied standard in project scheduling since it concerns activity planning and scarce resource allocation while optimizing a decision criterion [1]. Studies and analyses resulted in numerous variants based on the classic one. As well, as decentralization of business grows, scheduling research efforts are focusing on solving problems in distributed environments.
In this manuscript, we take the challenge to solve a new RCPSP variant called Multi-Site RCPSP (MSRCPSP) newly proposed by Laurent et al., 2017 [2] which is a result of decentralization. The MSRCPSP deals with a distributed RCPSP i.e. each activity has an execution location where it has to be realized. In fact, in a real multi-location environment, new constraints arise such as transportation of resources. Scheduling phases will be delayed because of moving required resources from one location to another. In this respect, resources taken into consideration may be fixed or mobile. This variant is very close to real world applications such as factory 4.0.

2. Multi-Site RCPSP

The MSRCPSP [2] adds for each activity a location where to be executed. It introduces also the concept of mobile and fixed resources. In fact, a fixed resource is located on a reference location and cannot be transferred to another one which means that activities requiring this type of resource must have the possibility to be executed on this site. While a mobile resource can move from location A to location B in order to ensure the execution of an activity consuming a time frame called Transportation time (TT), this time lag is applied as a finish-to-start precedence relationship between activities.
Formally, a site is a location where an activity could be executed such as a hospital, a warehouse, a construction site (plant) or even a computer in the case of grid computing applications [2]. The distance between sites represents the TT, it is a deterministic distance (measured in time units) between 2 sites. Actually, Transportation times are schedule-dependent, i.e. they depend on both locations where the resource is initially available and where it is moved to [2].
Technically, the inherent transportation concept is the result of two main cases:
- The transportation of a shared mobile resource: If two consecutive activities share the same mobile resource but they are assigned to different sites.
- The transportation of a semi-finished product: If there is a precedence dependency between two activities with different execution sites. In this case, resource transfers express transportation of intermediate goods, i.e. products under fabrication process or partly finished ones.

As the RCPSP has been confirmed to be NP-hard in the strong sense in [3], MSRCPSP is also NP-hard. It is noteworthy that adding the concept of activity location and resource pooling may require additional time to move a mobile resource from a distant location to the needed one and hence adds more complexity to solve the MSRCPSP compared to RCPSP.

2.1. Problem description

Authors in [2] presented a linear program for the introduced problem that takes into consideration new concepts embedded in the Multi-Site RCPSP. The mathematical model includes the application of Transportation times as well as respecting problem constraints such as non-preemption, precedence and resource consumption limit.
Formally, a MSRCPSP instance is described by a set of tasks N where two of them are dummies used to mark the beginning and the end of the project and a set of renewable resources of types k; $k \in [1..K]$ each of which is available in a limited quantity R_k. Each copy of resource k may be mobile or fixed; $M_{k,r} \in \{0, 1\}$, if $M_{k,r} = 0$ then $loc_{k,r}$ is the reference site of resource r of type k else $loc_{k,r}$ is the location site where the resource is available. An activity i; $i \in [1..N]$ has an execution duration p_i, a set of predecessors P_i consumes a given amount r_{ik} from each resource of type k If $Y_{j,k,r} = 1$ resource r of type k is assigned to activity j and have an execution location s ($loc_{k,r}$); $s \in [1..S]$ implying that $Z_{j,s} = 1$; site s is assigned to activity j. If a pair of activities (j, h) are subject to a transportation time constraint, $w_{j,h} = 1$ ($w_{j,h} \in \{0, 1\}$) and the transportation time is $\delta_{sj,sh}$.
The newly proposed model entails new assumptions, the most important are [2]:
- Each activity needs a site to be executed.
- During transfer, mobile resources are assumed to be unavailable.

- Transportation times are given for a pair of sites.
- Number of resources present on site in a time period is = (fixed resources + mobile resources that joined this site) - mobile resources that have left.
- A mobile resource is initially accessible on the location where it performed its first activity.

2.2. Literature review

It should be noted that the MSRCPSP is a newly introduced model on which only a few studies exist. In fact, the study on RCPSP in distant places was initiated by Gourgand et al. in [4] and [9] where authors characterized the studied model as a: Multi-skills, multi-mode, multi-places RCPSP. Authors planned to realize 100 medical examinations in a fixed time horizon (a week), results reveal that without resource pooling only 88 exams out of 100 were achieved while all the exams were accomplished with resource pooling between different hospitals. Analyses reveal that pooling resources enhances productivity and reduces resource consumption in a medical context.
Recently as a continuation of works in [4], Laurent et al. introduced the studied multi-site RCPSP in [2]. In fact, sharing resources between several locations of distributed factories is a challenging issue for decision makers [5]. In this extension, the merits of sharing resources are exploited through the concept of mobile resources. The latter move between several sites to satisfy activities requirements. Managing transportation of these resources is one of the studied model challenges. The authors produced an accurate set of instances for the newly introduced model that incorporates the multi-location context.

3. Proposed Solution

We propose in the present work to solve MSRCPSP with a population-based metaheuristic called Particle Swarm Optimization (PSO).
Initially elaborated by James Kennedy and Russell Eberhart in 1995, PSO algorithm is a recent evolutionary population-based metaheuristic that gained a significant importance in different research areas in the late decades thanks to its efficient performance in a wide range of applications. PSO process imitates the emerging behaviors of flocks of birds, schools of fish or even mankind social behavior in decision making process [6].
PSO is widely adopted to solve combinatorial optimization problem and RCPSP in particular as in [7][8][9][10][11] [12] [13] [15]. A major advantage of PSO is the fact of being problem independent which increases its flexibility and robustness [14].
Initially, PSO generates a random population of feasible solutions referred to as particles. The generated population evolves in a multidimensional search space. Each particle represents a potential solution that has a position and a velocity as well as its ability to remember its own best position and its neighbor's best positions.
A particle in PSO updates its position by updating its velocity and position means of equations 1 and 2.

$$V_i(t) = w(t)V_i(t-1) + c_1 r_1 \left(X_i^L - X_i(t-1)\right) + c_2 r_2 \left(X^G - X_i(t-1)\right) \quad (1)$$
$$X_i(t) = V_i(t) + X_i(t-1) \quad (2)$$

Velocity, i.e. rate of change [15] is a combination of factors that are inertia weight w(t) and learning factors c_1 and c_2. Inertia weight w(t) informs about the impact of the former velocity on the current one and influences the trade-off between exploration and exploitation abilities while learning factors are employed to maintain a harmony between personal and global experience. As well, r_1 and $r_2$are randomly generated real values in the range [0,1] to supervise the compromise between global and local processes.
Formally, in PSO M particles (size of the swarm) are moving in an N dimensional space (N is the number of activities in our work). A particle position is a vector consisting of N elements (a schedule) $X_i = [x_{i1}.. x_{iN}]$ as well as an N dimensional vector of velocity $V_i = [v_{i1}.. v_{iN}]$. The local best of a particle is denoted $L_i = [l_{i1}..l_{iN}]$ and the global best of the whole swarm is $G_i = [G_{i1}..G_{iN}]$. Local best and global best vectors represent best individual experience encountered by each particle and best position encountered by the whole population respectively.

3.1. Solution representation

The used PSO framework encodes a solution vector X by a position vector corresponding to the feasible activity sequence A_i denoted as $A_i = [a_{i1}..a_{iN}]$ paired with an adequate site allocation S_i denoted as $S_i = [s_{i1}..s_{iN}]$. The particle corresponding to activity list combination is represented through the priority-based form for position and it has its own velocity vector VAi(t) while for site assignment we calculate for each activity its own **possible site list** from which we select the site that generates the minimal completion time. That is the global solution X (a PSO particle and a site list) is:

$$X = A_i \cup S_i \tag{3}$$
$$X = \{(a_{i1}, s_{i1})), \ldots, \{(a_{iN}, s_{iN}); \quad i \in [1..M]\text{: Number of particles (generated solutions)} \tag{4}$$

The aim of the activities priorities representation is essentially to avoid conflicts of many schedulable activities competing on a set of scarce resources [15].
A particle position is a priority-based list depicted as $A_i = \{a_{i1}...a_{iN}\}$; ($i \in \{1..M\}$) along with its corresponding velocity $VA_i = \{va_{i1}\ldots va_{iN}\}$. While a site assignment, is a list $Si = \{s_{i1}\ldots s_{iN}\}$. The local best solution encountered by a particle i is $A_i^L = \{a_{i1}^L .. a_{iN}^L\} \cup S_i$ and the global best solution encountered by the swarm is $A^G = \{a_1^G .. a_N^G\} \cup S_i$. As for classic PSO, the velocity and position updating mechanism for our studied activity sequence can be realized by equation (5) and (6) depicted as follows:

$$VA_i(t) = w(t)\, VA_i(t-1) + c_1 r_1 (A_i^L - A_i(t-1)) + c_2 r_2 \left(A^G - A_i(t-1)\right) \tag{5}$$
$$A_i(t) = VA_i(t) + A_i(t-1) \tag{6}$$

range [0..1]. Activities priorities a_{ij} must be in the range [0,1] ([A_{Min}, A_{Max}]) and their associated velocities va_{ij} are limited in [-1,1] ([$-V_{Max}$, V_{Max}]) as used in [8] [9] [15]. These values must be adjusted if after updating process they exceed the interval limits for $i \in [1..M]$ and $j \in [1..N]$.

3.2. Proposed PSO process

3.2.1. Initial generation

Our algorithm starts with a randomly generated population of solutions i.e. a concatenation of a PSO-based particle and a right site list assignment.

3.2.2. Inertia Tuning

Introduced by Shi and Eberhart, inertia weight is a key factor in particles movement, it influences the algorithm's searching tendency and convergence process [5]. A high inertia value favors good diversification ability while a small value contributes to intensifying the search in the current space. For this reason, tuning inertia is regarded as an effective technique to reach a balance (trade-off) between exploitation and exploration abilities of PSO algorithm [17]. Mainly, inertia tuning strategy consists in decreasing gradually inertia value from large to small ones in the aim of realizing a wise trade-off between local and global exploration [14]. In fact, this mechanism permits a good global space exploration in the early stages that turns into an intensification throughout the search time Inertia tuning is used in many PSO-based solutions in literature [15] [16]... Many important inertia-tuning approaches were pointed out [14]from which we used the linear decreasing mechanism. Inertia fine-tuning mechanism depends on the maximum number of iteration and the index of current iteration. Initially, inertia value is set large enough (usually 0.9) then this value is decreased over time to reach a defined value (usually 0.3 or 0.4). According to this approach inertia for iteration t is obtained by equation (7):

$$w(t) = \left(w(0) - w(max_{it})\right) \frac{max_{it} - t}{max_{it}} + w(max_{it}) \tag{7}$$

Where max_{it} expresses the maximum iteration number, w(0) is the initial inertia value, $w(max_{it})$ is the final inertia value, and t is the current iteration.

3.2.3. Valid Particle Generator

PSO operates easily when project tasks are independent, but a problem arises when tasks are interdependent. This is the case of RCPSP (precedence dependencies) where constrained activities may lead to unfeasible sequences after PSO's updating mechanism. In our method we exploited the Valid Particle Generator (VPG) used in [16] to deal with this problem.

3.2.4. list scheduling algorithm for Makespan calculation

Starting from a feasible solution i.e. produced by VPG, a list scheduling algorithm is applied to assign adequate sites and compute completion times of activities as in [2]. As we are dealing with mobile and fixed resources, an activity must be scheduled at the earliest precedence-, resource-, and transportation times- feasible completion time. Firstly, we compute for each required resource type r_k for activity j the availability $av_{k,r}$ of corresponding copies using formulas from [2] where 2 major situations are possible:

- If resource copy is fixed
  - If $loc_{k,r} \neq s_j$; assignment is invalid: $av_{k,r} = \infty$;
  - If $loc_{k,r} = s_j$; $av_{k,r} = C_i$: where C_i is the completion time of the last activity i recently assigned to r_k
- if r_k is mobile; $av_{k,r} = C_i + \delta_{si,sj}$

Note that if j is the first activity assigned to r_k its availability is immediate and $av_{,k,r} = 0$.
After that, the earliest precedence-feasible time and resource-feasible time are calculated which. Technically, three equations are applied; the starting time of an activity j is the maximum between the earliest precedence feasible date EPFD (i.e. latest predecessor completion time including transportation times) and the earliest resource feasible date ERFD (i.e. maximum availability for its required resources including transportation times).

$$EPFD = max_{h \in P_j}(C_h + \delta_{si,sj}) \tag{8}$$

$$ERFD = max_{k \in \{1 \dots K\}; r \in \{1 \dots R_k\} / Y_{j,k,r}=1}(av_{k,r}) \tag{9}$$

$$C_j = max(EPFD, ERFD) + p_j \tag{10}$$

3.2.5. Stopping criterion

In our work, two stopping criteria are considered (1) Attaining the defined maximum iteration number or (2) Attaining the defined max iteration number after last solution updating.

4. Experimental results

In order to assess the performance of our approach, we used the PSPLIB inspired library created by researchers in [2], a new library that adapts the well-known literature instances library PSPLIB created by Kolish and Sprecher in 1997 for the basic Resource Constrained Project Scheduling Problem [25].
Our proposed algorithm is encoded in Java on a personal computer with an Intel(R) core (TM) i3-5005U CPU @ 2.00GHz processor. The used library of instances comprises data sets of 30, 60, 90 and 120 activities.
In our work, Taguchi method [26] will be used to identify the optimal combination of parameters used by our PSO procedure which are the learning factors c_1 and c_2, population size and the maximum number of iterations.
Results are selected over 20 experimental runs to avoid randomness effect and they report the effect of different parameter combinations on average deviation.
Performance of our model is evaluated in terms of two measures; Average Deviation (AvgDev) from classic RCPSP Best Known Solutions (optimal or Lower Bounds) and Computational Time. The first measure reports the relative gap between the given solution by our method and the Best Known Solution for RCPSP. Computational Time is the amount of time taken to run an algorithm i.e. its computational complexity. To the best of our knowledge, Laurent et al's work [2] is the first and only existing on the Multi-Site RCPSP. In this work, researchers calculated the average deviation from the classic RCPSP as it is considered as a lower bound for MSRCPSP (eq.11). It is worth mentioning

that our Best Known Solutions are recorded after 20 run per instance and the minimal makespan is saved.

$$AvgDev = (Obtained\ Solution - \ Best\ Known\ Solution)/Obtained\ Solution \quad (11)$$

In tables 1 and 2 we summarized our recorded average deviation from basic RCPSP using equation 11 and compared them with Laurent el al.'s obtained values.

Table 1. Average Deviation for two-sites-instances.

AvgDev(%)	DataSet			
	J30	J60	J90	J120
Laurent et al.(2017)	11.01	14.17	17.92	37.37
PSO	13.47	15.74	17.28	32.09

Table 2. Average Deviation for three-sites-instances

AvgDev(%)	DataSet			
	J30	J60	J90	J120
Laurent et al.(2017)	17.12	17.69	34.80	63.67
PSO	18.95	20.06	21.24	36.97

Our computational analysis in the present investigation shows a strong competitiveness with Laurent et al. methods when dealing with small instances (J30 and J60 for two and three sites benchmark instances). In addition, PSO bestows an evident advantage over the other approach in solving large-sized instances.

5. Conclusion

In the present work, we investigated a recently introduced RCPSP variant; the so-called MSRCPSP under total makespan minimization. Our proposed solution is based on PSO and it considers a set of particles corresponding to activity sequences paired with lists of assigned sites. We adopted the VPG operator from literature to repair invalid particles and generate feasible schedules. Moreover, a wise inertia-tuning principle is also embedded in our proposed PSO to promote a better exploration ability

PSO is competitive to Laurent et al. methods in addressing small-sized instances and outperforms the latter in addressing large-sized instances. Computational complexity confirms PSO fast convergence in fact PSO was able to find its optimal solution within a lower computational complexity and this is thanks to reducing the search space by only considering accessible sites for each activity.

Our future directions may include taking into account additional constraints or load balancing between different execution sites.

References

[1] Pinedo, M. L. (2005). Planning and scheduling in manufacturing and services. Springer.

[2] Laurent, A., Deroussi, L., Grangeon, N., & Norre, S. (2017). A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics. Computers & Industrial Engineering, 112, 634-644.

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

[4] Gourgand, M., Grangeon, N., & Klement, N. (2015). Activities planning and resources assignment on distinct places: A mathematical model. RAIRO-Operations Research, 49(1), 79-98.

[5] Blum, C., & Merkle, D. (2008). Swarm intelligence. Swarm Intelligence in Optimization; Blum, C., Merkle, D., Eds, 43-85.

[6] Kennedy J., Eberhart R. C. (1995), Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, 19421948.

[7] Khalilzadeh, M., Kianfar, F., Shirzadeh Chaleshtari, A., Shadrokh, S., & Ranjbar, M. (2012). A modified PSO algorithm for minimizing the total costs of resources in MRCPSP. Mathematical Problems in Engineering, 2012.

[8] Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization- based schemes for resource-constrained project scheduling. Automation in Construction, 14(3), 393-404.

[9] Zhang, H., Li, H., & Tam, C. M. (2006). Particle swarm optimization for resource-constrained project scheduling. International Journal of Project Management, 24(1), 83-92.

[10] Jia, Q., & Seo, Y. (2013). An improved particle swarm optimization for the resource-constrained project scheduling problem. The International Journal of Advanced Manufacturing Technology, 67(9-12), 2627-2638.

[11] Nasiri, M. M. (2013). A pseudo particle swarm optimization for the RCPSP. The International Journal of Advanced Manufacturing Technology, 65(5-8), 909-918.

[12] Chen, R. M., & Sandnes, F. E. (2014). An efficient particle swarm optimizer with application to man-day project scheduling problems. Mathematical Problems in Engineering, 2014.

[13] Liu, C., & Yang, S. (2011). A Serial Insertion Schedule Generation Scheme for Resource-constrained Project Scheduling. Journal of Computers, 6(11), 2365-2375.

[14] Engelbrecht, A. P. (2006). Fundamentals of computational swarm intelligence. Hoboken: John Wiley & Sons, Ltd.

[15] Blum, C., & Merkle, D. (2008). Swarm intelligence. Swarm Intelligence in Optimization; Blum, C., Merkle, D., Eds, 43-85.

[16] Kumar, N., & Vidyarthi, D. P. (2016). A model for resource-constrained project scheduling using adaptive PSO. Soft Computing, 20(4), 1565-1580.

[17] Lazinica, A. (Ed.). (2009). Particle swarm optimization. Kirchengasse: InTech.