MULTI-MODE RESOURCE CONSTRAINED MULTI PROJECT SCHEDULING PROBLEM OPTIMIZATION WITH SYMBIOTIC ORGANISMS SEARCH

Valentinus Alvin Hodianto[1], I-Tung Yang[2]

[1] Graduate Student of Civil and Construction Engineering, National Taiwan University of Science and Technology, [2] Professor at National Taiwan University of Science and Technology, Taiwan

[1] alvin9739@gmail.com, [2] ityang@mail.ntust.edu.tw

ABSTRACT: Multi-mode resource-constrained multi project scheduling problem (MRCMPSP) is the extension of standard resource constrained project scheduling problem which considers multiple activity execution modes and multiple projects, subject to precedence and resource constraints. Multiple execution modes allow the activities to have different duration and resource requirement. Furthermore, companies and project managers normally also handle many projects. This study proposed metaheuristic method symbiotic organisms search (SOS) along with random-key representations, parallel schedule generation scheme (P-SGS), and forward backward scheduling, to find the feasible schedule and minimal project duration of the project portfolio. The evaluation results from standard benchmark instances shows that SOS can get the best solution in most of the tested instances and also achieve better solution in some of them. The validation results from real project case MRCMPSP show that SOS has better performance than other tested metaheuristic methods, namely GA and PSO. Thus, validate the performance of SOS.

Keywords: multi-mode resource constrained multi project scheduling problem (MRCMPSP), metaheuristic, symbiotic organisms search (SOS)

1. INTRODUCTION

The lack of project scheduling is often regarded as one of the main reasons for project delays (Herroelen & Leus, 2005). Project scheduling can be defined as finding a start and finish time for all the activities, under certain constraints such as precedence relations, temporary restrictions and resource constraints, while a predefined scheduling objective is optimized (Félix Villafáñez, Poza, López-Paredes, Pajares, & Olmo, 2019). In the past decades there were two methods that have been mainly used to determine project scheduling: critical path method (CPM) and program evaluation and review technique (PERT). CPM and PERT usually assume that resources are unlimited and always available at the time of the activity execution (Bettemir Önder & Sonmez, 2015). This assumption may be unrealistic because resources

are usually shared between several activities and even several projects. Thus, these resource constraints often turn into a complex scheduling problem that is difficult to solve.

The Resource-Constrained Project Scheduling Problem (RCPSP) involves assigning a resource or set of resources to activities in the project with limited resource capacity, that are more realistic than CPM and PERT, in order to meet some predefined objective, such as minimized project timespan and cost (Yang, Geunes, & O'Brien, 2001). RCPSP could be extent based from the project environment and activity execution modes which are: resource-constrained multi project scheduling problem (RCMPSP), multi-mode resource constrained project scheduling problem (MRCPSP), and multi-mode resource constrained multi project scheduling problem (MRCMPSP).

Compared to other variations, MRCMPSP has the highest complexity and reflect higher practical relevance (Kannimuthu, Raphael, Ekambaram, & Kuppuswamy, 2020). In real-life scheduling situations, companies and project managers do not normally handle a single project but many projects (F. Villafáñez, López-Paredes, & Pajares, 2014). The addition of multiple execution modes is also one of the extension of RCPSP that allows the activities to have different duration and resource requirement (Sonmez & Gürel, 2016).

When dealing specially with the case of multi-project, there are two kinds of approaches that have been used: the first one is a mono-project approach, using dummy activities and precedence relations to combine the projects into a single mega-project, therefore simplify the multi-project into single-project with a single critical path. The second is a multi-project (MP) approach, maintaining the RCMPSP and a separate critical path per project (Kurtulus & Davis, 1982). This study uses the mono-project approach, combining all the project into single mega project with one critical path (centralized method).

Several techniques have been developed to solve the resource constrained scheduling problem. These techniques include exact methods, heuristics, and metaheuristic methods. When the project becomes larger with more than 50 activities and 3 or more execution modes, the exact methods are less efficient that makes heuristics and metaheuristics methods more preferable and suggested (Alcaraz & Maroto, 2001). Although many studies have been done about MRCPSP (multi-mode single project case) and RCMPSP (single-mode multi project case), there are less works that have combine both of the problem. Recently, the field of nature-inspired metaheuristics optimization algorithms (inspired by nature biological evolution) has grown very fast. One of them is symbiotic organisms search (SOS), that have been found by Cheng and Prayogo (2014). SOS is a powerful metaheuristic algorithm inspired by interaction between two biological organisms known as “symbiosis”. This study adopts SOS as the optimization algorithm to to minimize the total project makespan (*Cmax*) from all the project included in MRCMPSP, while satisfy the precedence and resource constraints.

To validate the performance, a real life project instances from India (retrieved from Kannimuthu et al. (2020), and standard benchmark instances from multi-mode and multi-project case are used to evaluate the performance of SOS. All dataset will be tested with mono-project approach, with the addition of random-key (RK) representations, parallel schedule generation scheme (P-SGS), and forward backward scheduling to the framework. The results of the real project instances will be compared with other popular metaheuristics algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO) to evaluate the performance of SOS in solving MRCMPSP.

2. LITERATURE REVIEW

2.1. Resource Constrained Project Scheduling Problem

The basic concept of RCPSP can be described as follows. A project consists of a set activities b (b= 1, 2, …, B). The precedence relations between the activities are Finish-Start (FS) which implies that activity b cannot be start until all its predecessors have finished. Each activity can be processed within a duration d_b without preemption once started. In addition, there are K types of renewable resources available for the project. For each resource k (k= 1, 2, …, K) its availability is constant per day as R_k, and the resource usage required for each activity is denoted as r_{bk}. The activities 1 and B are dummy activities that represents the start and finish of the project, which neither consume time and resources. Thus, $d_1 = d_B = 0$ and $r_{1k} = r_{Bk} = 0$.

Activities in B are related by the following types of constraints: (1) precedence constraints guarantee that each activity ($i \in B$) does not start until all of its predecessor activities ($h \in P_i$) have finished (P_i is a set of the predecessors of activity i); (2) The total amount of resource type k required for all activities being processed cannot exceed R_k in any processing time period. All information is assumed to be deterministic and known from early. The parameters are assumed to be non-negative and integer valued. The general objective is to minimize the makespan of the project by determine the earliest project finish time under the foregoing constraints.

2.2. Resource-Constrained Project Scheduling Problem (RCPSP) Classification

The basic concept of RCPSP is as mentioned above. However, to fulfill practical needs, changes have to be made to the basic concept by the researchers over time. These type of changes divide the element of RCPSP into several type classifications (Hartmann & Briskorn, 2010); (Habibi, Barzinpour, & Sadjadi, 2018), that can be seen on Figure 1.

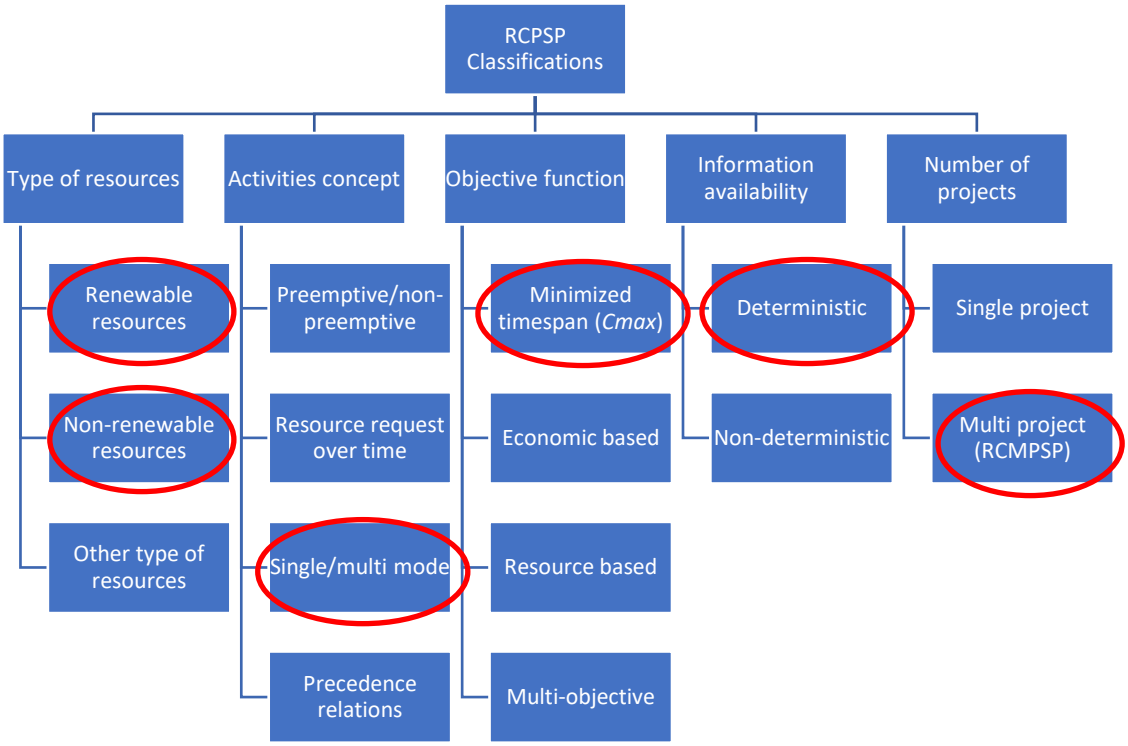

Figure 1. RCPSP classifications

The five types of classifications which are, (1) type of resources: renewable, and non-renewable; (2) concept of the activities: preemptive, multi-mode RCPSP, etc; (3) objective function: single or multi objective; (4) information availability: deterministic or non-deterministic; (5) number of projects: single project or multiple project. In addition, Blazewicz, Lenstra, and Kan (1983) have stated that RCPSP is a strong NP-hard problem. As a result, various methods are used according to the changes in the basic assumptions because it cannot be efficiently determined if the solution obtained is the optimal result for each problem.

This study that can be seen from Figure 1, considers multiple renewable resources and non-renewable resources as the type of resource constraints. At the activity level, this study addresses multi-mode with the basic assumption of RCPSP for non-preemptive scheduling, constant resource requests over time, and basic precedence relations. For the objective function, the basic objective function in this study is to minimize the project makespan (*Cmax*). For the information availability, this study adopts the standard deterministic method. Thus, all the project information was known from the start. For the number of projects, this study model multiple projects simultaneously with mono project approach. The mono-project approach is used because majority of the previous researches were developed for the mono-project approach. The mono-project approach also works well for the portfolio optimization in the set of projects, such as to minimize total makespan.

2.3. Multi-Mode Resource-Constrained Multi Project Scheduling Problem (MRCMPSP)

Multi-mode resource-constrained multi project scheduling problem (MRCMPSP) is the extension of RCPSP which considers multiple activity execution modes and multiple projects instead of the usual single mode and single project on RCPSP. Payne (1995) stated that up to 90% international projects are executed in a multi-project context. In addition, a studies from Maroto, Tormos, and Lova (1999) have shown that managers typically deal with up to four projects at once rather than one project only. Multiple modes allow time/cost, time/resource and resource/resource trade-offs to be considered (Wauters et al., 2014). Managing many projects using limited resources constraints to achieve high production efficiency and certain objective is more complicated than just considering a single project. The concept of MRCMPSP can be described as follows:

1. A company portfolio consists of parallel projects a (a= 1, 2, …, A). Each project consists of several b (b= 1, 2, …, B_a) activities, and each activity also consists of several m (m= 1, 2, …, B_{am}) execution modes.
2. Similar to RCPSP, all the activities in MRCMPSP can start after all of its predecessors have finished, and each activity can be processed within a duration d_{abm} in the respective mode without preemption once started.
3. There are also K types of renewable resources and N types of non-renewable resources available for all the project. For each renewable resource k (k= 1, 2, …, K), its availability is constant per day as R_k, and the resource usage required for each activity in each project is denoted as r_{abmk}. For each non-renewable resource n (n= 1, 2, …, N), its availability cannot be renewed and limited throughout the project as R_n, and the resource usage required for each activity in each project is denoted as r_{abmn}.
4. Both the renewable and non-renewable resources are independent and not interactive, which means that if one of the resources is limited, it will not affect the duration of the activity that has already been known from the start (deterministic).

5. For each time unit *t* in the schedule, the resource usage cannot exceed the availability of both R_k and R_n. If so, the feasible activities will have to be scheduled at a later time or to be scheduled with other execution modes to satisfy the resource constraints (further explain the point 4)
6. The first and last activities of each project is a dummy start and finish which has single mode with zero duration and zero usage of resources.
7. The main objective is to minimize the makespan (*Cmax*) of the project portfolio.

As mentioned briefly in Section 1, there are 2 main approaches that have been used in a multi project environment (Figure 2): mono-project approach (centralized-RCMPSP) and multi-project approach (decentralized-RCMPSP) (F. Villafáñez et al., 2014). In the mono-project approach, all projects are combined into one single mega project, thus reducing the RCMPSP to a RCPSP with a single critical path, a super-dummy start, and super-dummy end node (Browning & Yassine, 2010). In the multi-project approach, all the projects have separate critical paths per project, thus maintaining the RCMPSP.

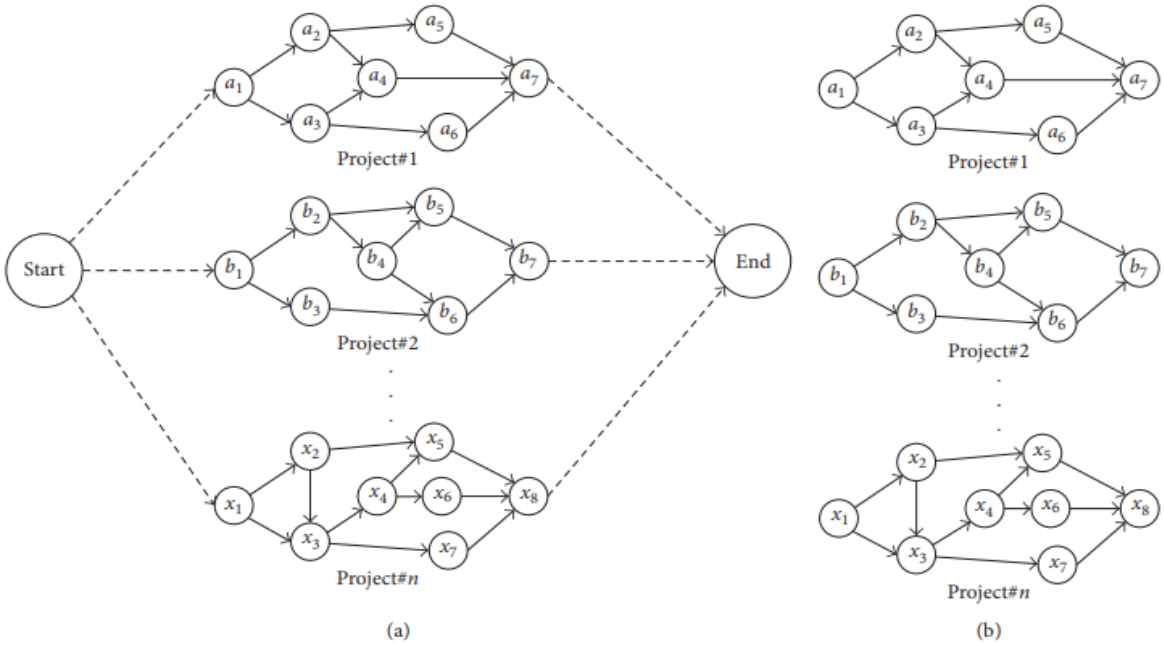

Figure 2. RCMPSP network model with two different approaches: (a) Mono-project approach. (b) Multi-project approach; Source: (Kannimuthu et al., 2020).

In the mono-project approach, it is also assumed that the company has the power to control all the project, or it can be said that only one project manager is responsible for scheduling and allocation of resources that can be shared between all projects for a mega network consisting of all the individual projects (F. Li, Xu, & Li, 2021). This study chooses to focus on the mono-project approach, because majority of the previous researches were developed for the mono-project approach. Thus, it is not easy to find benchmarking case or studies for the multi-project approach. Kannimuthu et al. (2020) also made a comparison between the mono-project and multi-project approaches in multi-objective trade-off between time, cost, and quality MRCMPSP. They found that in a multi-project environment, the mono-project approach generates better schedules than the multi-project approach in the multi-objective optimization. The mono-project approach RCMPSP also works well for the portfolio optimization in the set

of projects, such as the objective on this study to minimize total makespan (Kurtulus & Davis, 1982); (Özdamar & Ulusoy, 1995). When multiple projects are combined into a single mega project, it becomes similar to the single RCPSP, hence the method used for RCPSP can be used to solve the mono-project approach of MRCMPSP.

Previous research in the multi-project environment mainly focused on using heuristic methods. Among them, heuristic relied on priority rules. One of the earliest studies about MRCMPSP has been done by Pritsker, Waiters, and Wolfe (1969) to develop integer programming formulation to solve multi-mode resource usage RCMPSP. Lova and Tormos (2001) studied the effect of schedule generation schemes and priority rules for RCMPSP and found that the parallel schedule generation scheme performed well compared to serial schedule generation scheme on multi-project scheduling problem. This study uses parallel schedule generation scheme (P-SGS) that does time incrementation to generate the schedule that will be further discussed on Section 3.

Many metaheuristic methods have also been developed to solve the multi-project environment efficiently. Example such as, Linyi and Yan (2007) that employed PSO with one-point crossover approach to minimize the makespan of RCMPSP. Gonçalves, Mendes, and Resende (2008) developed GA using random keys representation for RCMPSP. Random key is one type of representations for decoding procedure to obtain feasible schedule which is comprised of real random numbers between 0 and 1. This study applied random key (RK) representation that will also be further discussed on Section 3. Chen and Shahandashti (2009) applied a hybrid metaheuristic, GA and simulated annealing (SA) to solve RCMPSP with multiple resource constraints, to three real project instances which have different types of precedence relations. Sonmez and Uysal (2015) presented a backward-forward hybrid genetic algorithm (BFHGA) for optimal scheduling on multi-project instances. And, Kannimuthu et al. (2020) used a direct search algorithm called probabilistic global search Lausanne to solve MRCMPSP with the multi-objective trade-offs among time, cost, and quality.

3. RESEARCH METHODOLOGY

In this study the proposed SOS will be used with RK representation, P-SGS, and forward-backward scheduling to obtain the feasible schedule. A simple project example with single project case and two execution modes from Zhang (2012) is used to easily described the process. The single project case can be used because the multiple projects on this study are also combined into a single mega project (mono-project approach), thus it becomes similar to the single project.

3.1. Basic Symbiotic Organisms Search (SOS)

SOS developed by Cheng and Prayogo (2014), is a simple and powerful metaheuristic optimization algorithm that can be used in various problem. SOS employs a population-based search strategy to search for the optimal solution to a specific objective function, and it simulates symbiotic interaction strategies that organisms use to live inside the ecosystems. A main advantage of the SOS algorithm over other metaheuristic algorithms is that its operations does not have any tuning parameters. Thus, it does not require tuning at all. SOS begins with an initial population called the ecosystem. In the initial ecosystem, a group of organisms is generated randomly within the search space. Each organism represents one candidate solution to the corresponding problem. Each organism in the ecosystem is associated with a

certain fitness value, which reflects degree of adaptation to the desired objective. In every iteration, a new generation consisted of new organisms is generated based on biological interaction between two organisms in the ecosystem. Three symbiosis phases that resemble the real-world biological interaction model are used in SOS. The three phases include: mutualism, commensalism, and parasitism phase. Each organism interacts with other organism randomly through all phases. The process is then repeated until the stopping criteria is met.

3.2. SOS Proposed Framework for MRCMPSP

The preceding general SOS concept will serve as the basis for developing the framework of MRCMPSP as can be seen on Figure 3.

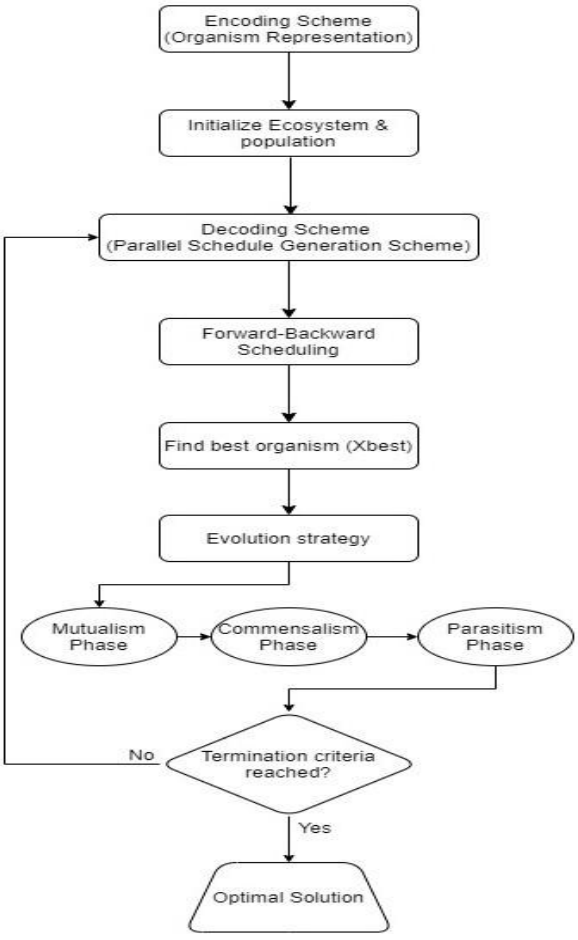

Figure 3. SOS for MRCMPSP flowchart

To start with SOS or any other metaheuristic method, one has to select a suitable representation for solutions. Metaheuristic approaches for RCPSP usually are operate on representations of schedules than on schedules themselves (Kolisch & Hartmann, 1999). After the population is generated, an appropriate decoding procedure must be selected to transform the representation into a feasible schedule. Finally, similar to other population-based metaheuristic methods, an evolutionary strategy of SOS which combines mutualism, commensalism, and parasitism phase are needed to generate new individuals to produce possible better solution.

The following sub-sections will describe the proposed framework in detail, using a case from Zhang (2012) that can be seen on Figure 4 and Table 1. There are 6 activities, 2 modes, and 1 renewable resource with 4/day capacities. The mode predefined all the resources in to one set of a mode, which means that in each mode, all activities will have the same type of resource used on other mode, but with different resource requirement and duration. It is also assumed that the relation between the duration and resource requirement may not be linear.

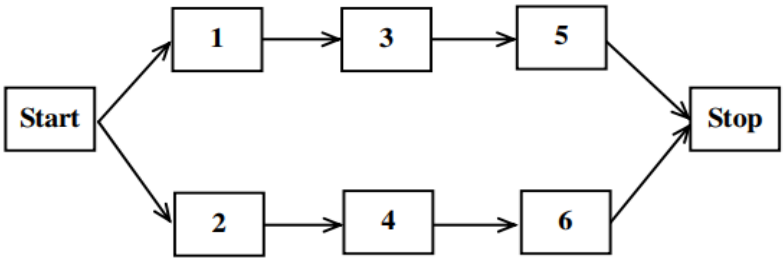

Figure 4. Project example; Source: (Zhang, 2012)

Table 1. Information about the example project case

Activity	Mode 1		Mode 2	
	Resource requirement	Duration	Resource requirement	Duration
1	2	3	1	4
2	3	4	2	6
3	4	2	2	3
4	4	2	3	3
5	3	1	1	3
6	2	4	1	6

3.2.1. Encoding Scheme (Organism Representation)

There are five representations that have been used and reviewed in the previous literature for the encoding scheme (Kolisch & Hartmann, 1999), which are activity list (AL), RK, priority rule (PR), shift vector (SV), and schedule scheme (SS). This study used RK Representations. In the RK scheme, a series of array is provided according to the number of activities. The RK value will represent the activity priority and the RK position showed the activity index. In Figure 5, the RK scheme example from project example is presented. The first column represented the activity number 1 (dummy start) and the value inside represents the organism priority position.

0.52	0.51	0.59	0.69	0.1	0.31	0.46	0.72

Figure 5. RK scheme from project example

In SOS, an organism of candidate solutions represents a potential solution, subject to the objective function. As mentioned above, each organism consists of an array. Each organism of the SOS consists of $\sum_1^2 N$ number of array elements, which are real numbers between 0-1. Where *N* is the total number of activities from all project, and the real numbers between 0-1 showed the organism priority position. The first N array elements will represent the organism position for each activity, and the last N array elements will represent the organism position for the mode used in each activity. In Figure 6, it can be seen that the RK has 16 array elements for 8 activities. Whereas the first 8 array elements are each activity position from all project (in this case it only has 1 project), and the last 8 array elements indicate the mode used for each activity.

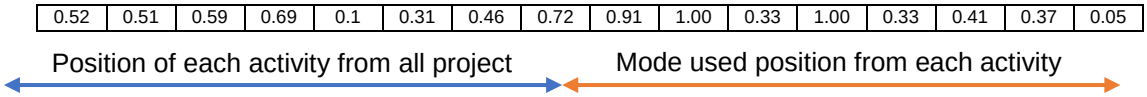

Figure 6. RK Representations used on this thesis

3.2.2. Initialize Population

After the encoding process is completed, SOS begins with an initial population called the ecosystem. A group of organisms is then generated randomly within the search space. Each organism represents one candidate solution and is associated with a certain fitness value which is the project makespan. Control parameters include, the number of solutions, stopping criteria, and the maximum number of iterations.

3.2.3. Decoding Scheme (Schedule Generation Scheme)

The schedule generation scheme (SGS) is the core of most heuristic procedure in RCPSP problem (Kolisch, 1996). SGS is a technique that decode the encoding scheme representation to build a feasible schedule from scratch into a valid complete schedule, subject to precedence and resource constraints. There are only two types of SGS available which are serial schedule generation scheme (S-SGS) and parallel schedule generation scheme (P-SGS). Both schemes choose activities from the priority list and add them to a partial schedule until all project activities are assigned into complete schedule. This study uses the P-SGS for the decoding scheme to construct the feasible schedule.

P-SGS uses the time incrementation approach in their principal. It consists of c =1, …, n turn, each has a schedule time t_c. P-SGS iterates over the time horizon t_c which start with t_c=0 and add activities that are eligible to be scheduled as the time increased. Associated with time t_c there are three separate activity sets which are the scheduled set (S_c), active set (A_c), and feasible set (F_c). S_c consists of all the activities that have been scheduled until t_c. A_c consists of the activities that are active at t_c. And F_c comprises of the activities that are eligible to be scheduled. S_c, A_c, and F_c on the P-SGS are different from the set of all activities B. Table 2 shows the process of P-SGS for the example case.

Table 2. P-SGS example

c(turn)	1	2	2	3	4	4	5	6
t_c	0	0	0	4	6	6	10	11
F_c	[0]	[1,2]	[2,3]	[3,4]	[3,6]	[5,6]	[5]	[7]
s	0	1	2	4	3	6	5	7

3.2.4. Forward-Backward Scheduling

In the previous sub-section, the P-SGS is used as the decoding scheme to construct a feasible schedule. The P-SGS sequentially schedules the activities based on the time incrementation, at their earliest precedence and resource feasible start time (forward-scheduling), according to the organism RK priority positions. The P-SGS could also be executed in the reverse time direction (backward-scheduling). The forward and backward scheduling procedure was first proposed by K. Y. Li and Willis (1992). Backward scheduling applies the P-SGS to the reversed precedence network where the dummy finish become the new dummy start and vice-versa. After that, all the activities are scheduled in a reverse direction from forward scheduling until the schedule is complete.

Because the exact duration of the feasible schedule is not known in backward scheduling, usually an arbitrary project completion time or the early finish time from forward scheduling is selected to start the backward scheduling (Sonmez & Uysal, 2015). To simplify the procedure, on Figure 9 the start time of backward scheduling (dummy finish) is set to be 0. After the schedule is complete until the dummy start, the fitness value of project duration can be known automatically similar to forward scheduling. The proposed algorithm employs P-SGS in order to iteratively schedule both forward and backward scheduling in each iteration. Then, the proposed algorithm will choose the minimum duration from between for each organism, and store the result as the fitness value. Figure 7 illustrate the forward scheduling for the example case, while the normal and proposed backward scheduling can be seen in Figure 8 and 9.

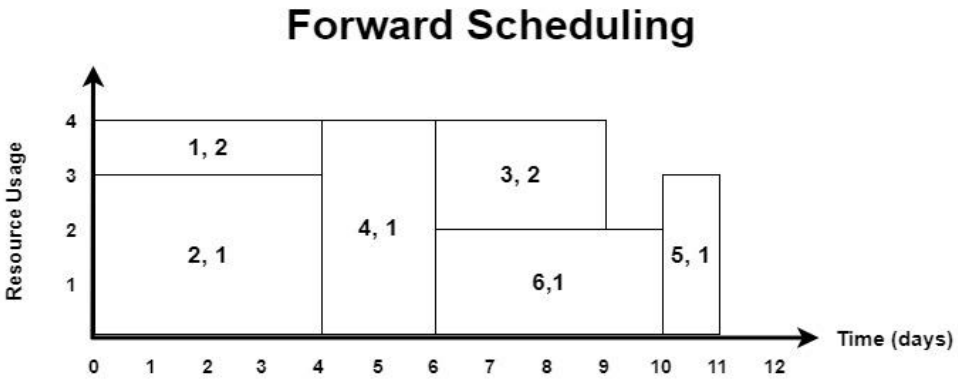

Figure 7. Forward scheduling from project example

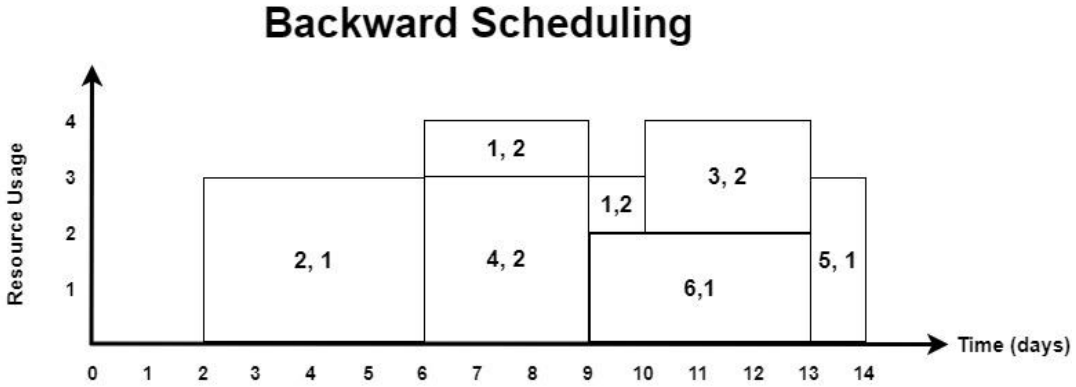

Figure 8. Normal backward scheduling from project example

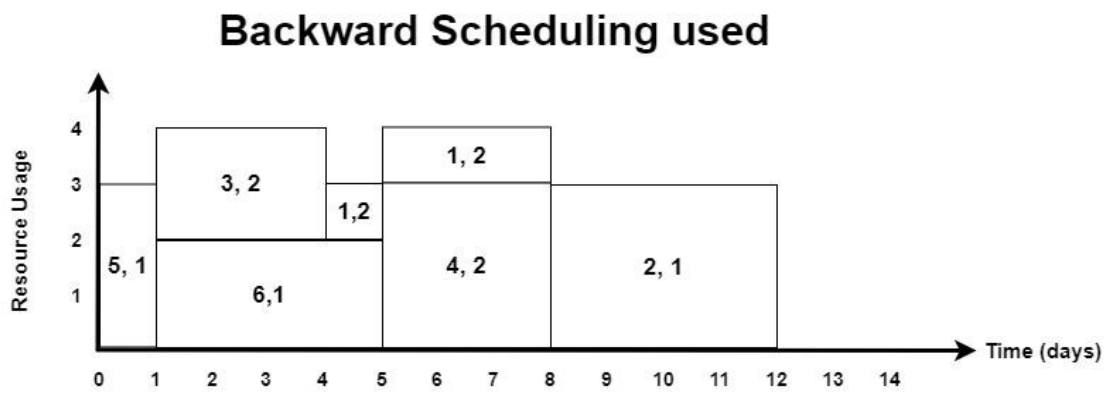

Figure 9. Backward scheduling example from project case (the method used on this study)

3.2.5. Evolution Strategy

In order to generate new solutions for the next iteration, almost all metaheuristic algorithms apply a succession of operations to solutions after each iteration (Cheng & Prayogo, 2014). GA has two operators which are crossover and mutation, while PSO updates the solution through pbest and gbest. In SOS, the evolution comes from the 3 ecosystems phase which are mutualism, commensalism, and parasitism that have been mentioned in the previous section. In each ecosystem phase, the old organism can be replaced by a potential new organism if the fitness value of the latter is better. In this study, the ecosystem searches for better solution by updating the organism positions as the decision variables, which are the random-key values in each of the three phases after every iteration. When the stopping criteria or the number of maximum iterations were met, the algorithm will stop with the shortest project makespan as the optimal solution.

3.3. Software for System Development

This study implements the proposed algorithm in Python language (version 3.8) on Spyder software to solve the MRCMPSP. The machine is a personal computer with a speed of 2.9 GHz Intel Core i5-10400 CPU and 8 GB RAM.

4. CASE STUDY

The performance of SOS in solving MRCMPSP problem will be tested using several case studies, which are standard benchmark instances of MRCPSP and RCMPSP, and real project instances of MRCMPSP. The benchmark instances are used to evaluate the general performance of SOS in theoretical cases, whereas the real project instances are used to validated the performance of SOS in practical MRCMPSP problems.

4.1. Evaluation Testing with Benchmark Instances

First, to evaluate the performance of SOS and the proposed algorithm, comparisons are made in well-known standard benchmark problems of project scheduling. Because there is no open standard benchmark problem for the MRCMPSP problem, the benchmark problem from the MRCPSP (multi-mode & single-project) and RCMPSP (single-mode & multi-project) case is used to compare the results of SOS with the best known solution (BKS). The benchmark problems for the MRCPSP are obtained from Project Scheduling Problem Library by (Kolisch

& Sprecher, 1997) (PSPLIB, http://www.om-db.wi.tum.de/psplib/getdata_mm.html) and (Van Peteghem & Vanhoucke, 2014) (MMLIB, https://www.projectmanagement.ugent.be/research/data). While the benchmark problems for the RCMPSP are obtained from (Homberger, 2007) (MPSPLib, http://www.mpsplib.com/index.php) and (Vázquez, Calvo, & Ordóñez, 2015) (RCMPSPLIB, https://www.eii.uva.es/elena/RCMPSPLIB.htm). The population size is set to be 100, while the maximum number of iterations is set to be 1000. The evaluation results are summarized on Table 3 – Table 6.

The 20 tested instances chosen from PSPLIB are summarized on Table 3. The BKS are the time makespan best known solution computed by various method from the past to the PSPLIB library. The results from table 3 indicated that SOS is able to solve the MRCPSP PSPLIB problem very well. From 20 tested instances, SOS was able to determine the optimal solution in 19 instances, whereas on instance j3021_2 SOS was only able to obtain a near-optimal solution from the BKS with 1 day difference (2.94% deviation). In average, the percent deviation from BKS of the 20 tested instances is 0.15%.

Table 3. Results from PSPLIB instances

No	MRCPSP PSPLIB	BKS	SOS	Dev from BKS (%)
1	j1026_1	14	**14**	**0**
2	j1026_3	16	**16**	**0**
3	j1019_1	13	**13**	**0**
4	j1019_2	15	**15**	**0**
5	j1020_3	21	**21**	**0**
6	j2064_1	21	**21**	**0**
7	j2064_2	23	**23**	**0**
8	j2064_3	23	**23**	**0**
9	j2064_4	19	**19**	**0**
10	j2064_5	26	**26**	**0**
11	j3021_1	38	**38**	**0**
12	j3021_2	34	**35**	**2.94**
13	j3021_3	36	**36**	**0**
14	j3021_4	37	**37**	**0**
15	j3021_5	37	**37**	**0**
16	j3010_1	26	**26**	**0**
17	j3010_2	28	**28**	**0**
18	j3010_3	24	**24**	**0**
19	j3010_4	36	**36**	**0**
20	j3010_5	33	**33**	**0**

The 20 tested instances chosen from MMLIB are summarized on Table 4. The BKS are the time makespan best known solution computed by various method from the past to the MMLIB library. Table 4 shows that the performance of SOS in large scale instances MRCPSP is not that good compared to the smaller problem MRCPSP from PSPLIB library. Given that MRCPSP with 2 or more non-renewable resources is NP-complete problem in the strong sense (Kolisch & Drexl, 1997), consequently, the difficulty to solve the problem will increase as the size of the problem increases. It can be seen from the results with 3 modes, from MMLIB50 and MMLIB100 subset (number 1-10), that SOS is still able to find the optimal or near optimal solution in 6 problems, out of 10 problems given. While on the larger instances

from MMLIB+ (number 11-20), the deviation from BKS solution become bigger. The average results deviation for the tested instances with 50 activities and 9 modes are 16.9%, and for the tested instances with 100 activities and 6 modes are 21.15%. The average results deviation from all tested instances from MMLIB is 10.4%.

Table 4. Results from MMLIB instances

No	MRCPSP MMLIB	BKS	SOS	Dev from BKS (%)
1	J503_1	19	**19**	**0**
2	J503_2	20	**20**	**0**
3	J503_3	18	**19**	**5.56**
4	J503_4	23	**23**	**0**
5	J503_5	18	**19**	**5.56**
6	J10051_1	32	**32**	**0**
7	J10051_2	34	**34**	**0**
8	J10051_3	32	**33**	**3.13**
9	J10051_4	39	**39**	**0**
10	J10051_5	29	**30**	**3.45**
11	Jall217_1	103	**123**	**19.42**
12	Jall217_2	86	**102**	**18.60**
13	Jall217_3	86	**99**	**15.12**
14	Jall217_4	83	**96**	**15.66**
15	Jall217_5	121	**140**	**15.70**
16	Jall500_1	100	**125**	**25**
17	Jall500_2	88	**108**	**22.73**
18	Jall500_3	78	**94**	**20.51**
19	Jall500_4	94	**112**	**19.15**
20	Jall500_5	98	**116**	**18.37**

The 10 tested instances chosen from MPSPLib are summarized on Table 5. The BKS are the time makespan best known solution computed by various method from the past to the MPSPLib library. Table 5 shows that the proposed SOS algorithm can get similar results with the BKS in 4 out of 10 tested instances (mpj90a2agentcopp5, mpj90a5agentcopp1, mpj90a20agentcopp2, mpj120a2agentcopp5). While in the other 6 tested instances the average deviation from the BKS are just 2.24%, with the largest are mpj120a10agentcopp3 instance with 4.77% difference (22 days) from the BKS. The average results deviation from all tested instances is 1.57%.

Table 5. Results from MPSPLib instances

No	RCMPSP MPSPLIB	BKS	SOS	Dev from BKS (%)
1	mpj90a2agentcopp2	330	**336**	**1.82**
2	mpj90a2agentcopp5	72	**72**	**0**
3	mpj90a5agentcopp1	568	**568**	**0**
4	mpj90a10agentcopp10	170	**174**	**2.35**
5	mpj90a20agentcopp2	127	**127**	**0**
6	mpj120a2agentcopp1	212	**218**	**2.83**
7	mpj120a2agentcopp5	95	**95**	**0**

No	RCMPSP MPSPLIB	BKS	SOS	Dev from BKS (%)
8	mpj120a5agentcopp8	527	**533**	**1.14**
9	mpj120a10agentcopp3	461	**483**	**4.77**
10	mpj120a20agentcopp6	863	**887**	**2.78**

The 14 tested instances chosen from RCMPSPLIB are summarized on Table 6. The BKS are the time makespan best known solution available on the RCMPSPLIB library. The results from Table 6 showed that the SOS algorithm able to outperforms the previous best known solutions from Vázquez et al. (2015) in 10 of the 14 tested instances, and performs as good as the previous best known solutions in 2 of the 14 tested instances . Whereas for the other 2 instances, the average deviation from the BKS are 3.34%, with the largest are found on the instance mpj30a2, that have 3.54% difference from the BKS (4 days). The average results deviation from BKS of all RCMPSPLIB tested instances is -1.88%.

Table 6. Results from RCMPSPLIB instances

No	RCMPSPLIB (Vazquez et al., 2013)	BKS	SOS	Dev from BKS (%)
1	mpj30a2	113	**117**	**3.54**
2	mpj30a4	228	**223**	**-2.19**
3	mpj30a6	153	**150**	**-1.96**
4	mpj30a10	313	**300**	**-4.15**
5	mpj60a2	117	**111**	**-5.13**
6	mpj60a3	276	**263**	**-4.71**
7	mpj60a4	356	**341**	**-4.21**
8	mpj60a5	149	**149**	**0**
9	mpj90a2	90	**89**	**-1.11**
10	mpj90a3	390	**374**	**-4.10**
11	mpj90a4	698	**683**	**-2.15**
12	mpj90a5	114	**114**	**0**
13	mpj120a2	181	**175**	**-3.31**
14	mpj120a3	511	**527**	**3.13**

This evaluation results particularly show that the SOS is able to solve complex multi-project scheduling problems relatively well from the 2 benchmark instances that have been used from MPSPLib and RCMPSPLIB. Although, for the instances with a relatively large number of activities (more than 1000 activities) such as the MPSPLib problem from MMLIB library, the results of SOS are particularly not that good.

4.2. Validation Testing with MRCMPSP Real Case

MRCMPSP real project instances, retrieved from Kannimuthu et al. (2020) is used to validate the performance of SOS. The result of SOS will be compared with two other popular metaheuristics algorithms, GA and PSO. The original data also consists of cost and quality value, along with related constraints. The objective function is to find the optimal trade-off relationship among time, cost, and quality. However, this study ignored both the cost and quality data and focus on the single objective to search the minimal total makespan of multiple projects.

4.2.1. Project Information

The data contains of 3 building construction projects from India (Kannimuthu et al., 2020), namely, projects X, Y, and Z. Project X has 32 activities, project Y has 28 activities, and project Z has 18 activities. The estimated total combinations of the data are 14.91^{32} for project X, 18.61^{28} for project Y, and 18.61^{18} for project Z, where the base numbers (14.91 and 18.61) are the average number of execution modes for each activity, and the exponent is the number of activity The activities that are considered in the schedule are for the construction of the first two floors of the above projects, which include: formwork, rebar/reinforcement, concreting, block work, plastering, painting, and flooring. In addition, the data also consists of 22 renewable resources to be shared among all projects.

Because this study models the projects in a mono-project approach, the precedence network of project X, Y, and Z need to be combined into single mega project with one dummy start and finish. The activity number are also changed to be continuous for every project, based on the last activity number of the previous project. The combined precedence network of project X, Y, and Z can be seen in Figure 10. The activity number 1 is the dummy start and activity number 80 is dummy finish.

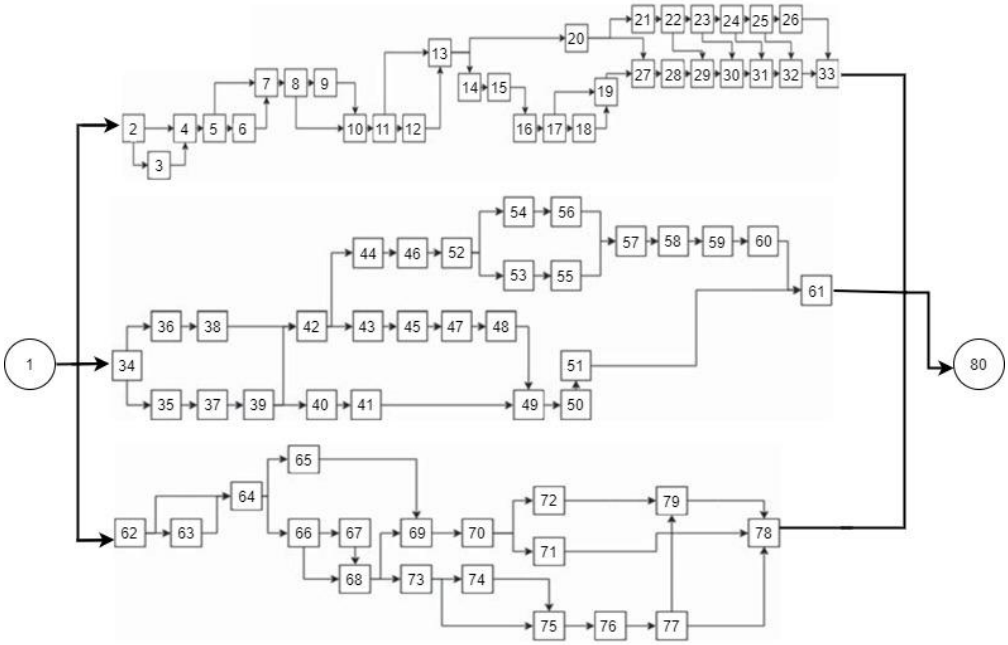

Figure 10. Combined precedence network

4.2.2. Parameter Selection

As mentioned in Section 1, the optimization results of SOS in this case will be compared with the results from GA and PSO. In SOS each iteration generated 4 function evaluations, whereas in GA and PSO each iteration generated 1 function evaluation. To compare it equally, the number of objective function evaluation for all the three algorithms must be the same. Thus, the number of iterations in GA and PSO need to be multiplied by 4 to make it comparable. The population size and the number of iterations of SOS are set to be 100 and 500 respectively. The 500 iterations are involved with 2000 calls for the objective function. Therefore, the population size and number of iterations of GA and PSO are set to be 100 and 2000

respectively. The larger the population size and number of iterations allows for more computations and may lead to better objective function values. The parameters used for SOS, GA, and PSO in this real case study projects are summarized in Table 7.

Table 7. Parameter selection

Metaheuristics	Parameters	Value
SOS	Pop size	100
	Number of max. iterations	500
	Number of objective functions called	2000
GA	Pop size	100
	Number of max. iterations/ obj. functions called	2000
	C_rate	[0,6-1.0]
	M_rate	[0,01-0,25]
PSO	Pop size	100
	Number of max. iterations/ obj. functions called	2000
	inertia weight (w)	w(1)= 0,9; w(T)=0,1
	c1	2
	c2	2
	vmax	0.3

4.2.3. Results and comparison

Metaheuristics are stochastic search. The performance therefore should be evaluated by statistical tests. For this case, 10 runs of experiments are carried out for every of the three metaheuristics methods. The statistics of the performances are completed and summarized on Table 8. The optimal solutions of minimum duration (*Cmax*), average project duration, and standard deviation from the 10 runs of experiments are used for comparison between SOS, GA, and PSO.

Table 8. Results SOS, GA, and PSO from 10 experiments

Metaheuristics	Minimal project duration (*Cmax*)	Average project duration	Standard Deviation (SD)
SOS	**98**	**98.4**	**0.70**
GA [0,6;0,01]	107	107.6	0.97
GA [0,8;0,15]	106	107.1	0.88
GA [1,0;0,25]	106	107.4	0.84
PSO	100	104.6	2.27

The results from Table 8 shows that SOS achieved better results in all minimal project duration, average project duration, and standard deviation compared to GA and PSO. SOS is able to obtain the minimal project duration of 98 days, followed by PSO 100 days, and GA 106 days. For the average project duration, SOS achieved the best average results of 98.4 days from 10 run of experiments, followed by PSO 104.6 days, and GA 107.1 days with crossover rate=0,8 and mutation rate=0,15. While for the standard deviation, SOS able to achieved the lowest standard deviation of 0.70 compared to 0.84 for GA with crossover rate=1,0 and mutation rate=0,25 and 2.27 for PSO.

The convergence from the best results of the 3 metaheuristics method are shown and compared in Figure 11. The *y* axis in the figure is limited in range 95-180 days to make the

comparison easier to see. The figures show how the SOS, GA, and PSO achieved the results through the number of objective function-called and converging to the optimal solution. The figures show that GA and PSO converging faster to their optimal solution compared to SOS. For example, take the 110 days as the criterion. It can be seen that GA only needs a couple of objective function-called to achieve the results of 110 days, followed by PSO who needs around 125 number of objective function-called, while SOS needs around 250 objective-functions called to obtain the results. However, despite having the slowest convergence rate, SOS is able to achieved the most optimal results of 98 days at the end of the iterations. Thus, it also can be said that SOS can escape from the local optima solution compared to GA and PSO, to achieved better global solution.

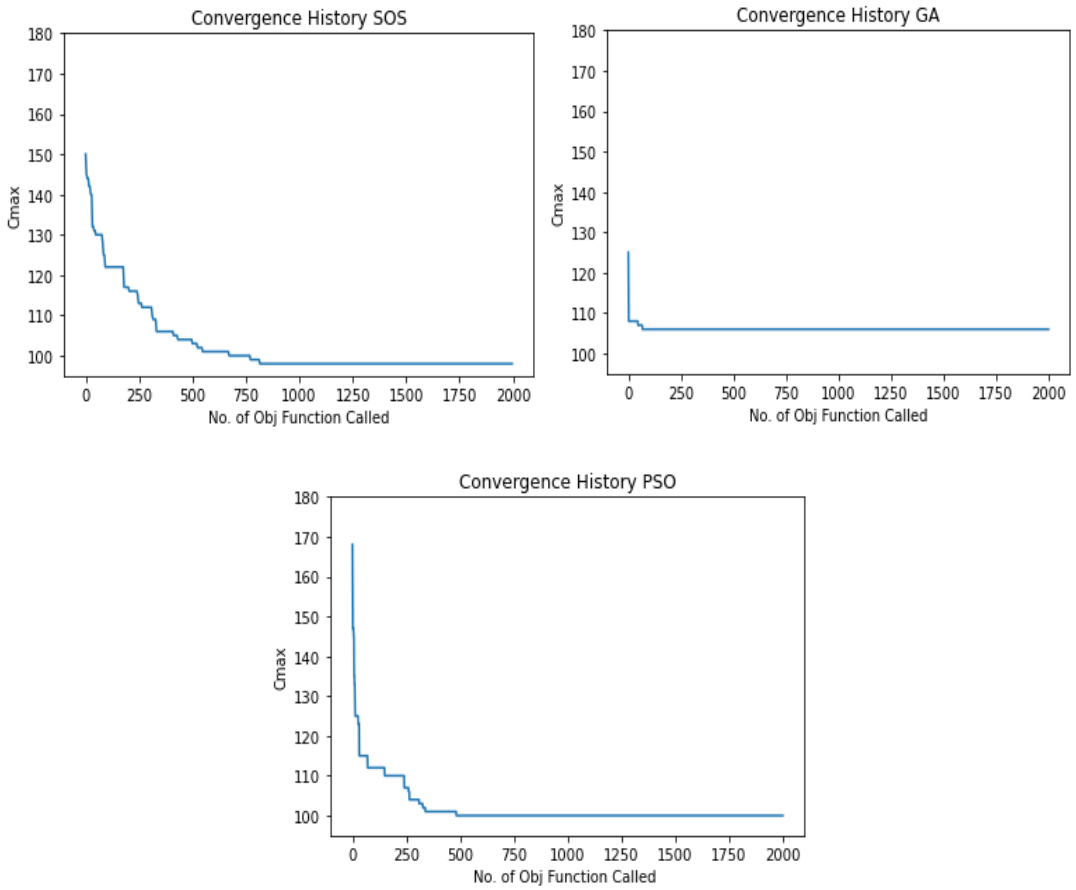

Figure 11. Graphs of convergence comparison

5. CONCLUSIONS

The primary purpose of this study is to minimized the total makespan (*Cmax*) from the MRCMPSP, while must satisfy the precedence and resource constraints. To achieve this purpose, this study proposed metaheuristic method SOS along with RK representations, P-SGS, and forward backward scheduling in one framework, to find feasible schedule and the minimal project duration of the project portfolio. The proposed SOS system along with RK representations, P-SGS, and forward-backward scheduling are combined in one framework to find the feasible schedule and the minimal project duration. To test the performance of the proposed system, several case studies, which are standard benchmark instances of MRCPSP

and RCMPSP, and real project instance of MRCMPSP are used to evaluate and validate the performance of SOS. Furthermore, in the real project instance the performance of SOS is compared with other metaheuristics methods: GA and PSO.

The evaluation results from standard benchmark instances of MRCPSP and RCMPSP (retrieved from: PSPLIB, MMLIB, MPSPLib, and RCMPSPLIB) show that SOS can achieve the best-known solution in some of the tested instances and also outperforms/ achieve better solution in 10 out of 14 RCMPSPLIB instances. However, when the problem is more complex (i.e. many modes and many project activities) the algorithm may not always reach the best-known solution results. Thus, in the future, further work is still needed to try combined and improve different features of encoding and decoding scheme (i.e: AL representations, S-SGS, etc) for the performance of SOS, to potentially achieved better results. The average deviation from BKS in the 4 benchmark libraries is 0.15% in PSPLIB, 10.4% in MMLIB, 1.57% in MPSPLib, and -1.88% in RCMPSPLIB.

The validation results from real project case of MRCMPSP from Kannimuthu et al. (2020) show that SOS is able to achieved better solutions compared to GA and PSO in all minimal project duration (*Cmax*), average project duration, and standard deviation in 10 runs of experiment. SOS found a minimal project duration of 98 days compared to 100 days for PSO, and 106 days for GA. For the average project duration, SOS average 98.4 days from the 10 experiments, compared to 104.6 days and 107.1 days for PSO and GA respectively. While for the standard deviation, SOS achieved the lowest SD with 0.70 compared to 0.84 and 2.27 for GA and PSO respectively. It was also shown from the graphic of convergence that SOS takes more objective function calls to achieve the optimal solutions compared to GA and PSO. However, SOS able to achieve better minimal project duration (*Cmax*) and average project duration, which indicates that SOS will not premature/ be trap in the local optima solution and can escape to achieve better global solutions. Furthermore, SOS also able to achieved the lowest standard deviation compared to GA and PSO, which indicate that it can produce the most consistent results with low variance.

For the future research, several suggestions were proposed in response to this study results. The first one, future research could integrate different features of encoding and decoding schemes with the proposed framework (i.e: AL Representations and SSGS), to seek better performance. Future study can also consider other classifications such as allowed activity preemption, non-deterministic information, or other objective functions, such as cost-based objective (i.e: maximize Net Present Value), resource-based objective (i.e. minimize resource utilization), as well as the tradeoff between time and cost.

6. REFERENCES

Alcaraz, J., & Maroto, C. (2001). A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research, 102*(1), 83-109. doi:10.1023/A:1010949931021

Bettemir Önder, H., & Sonmez, R. (2015). Hybrid Genetic Algorithm with Simulated Annealing for Resource-Constrained Project Scheduling. *Journal of Management in Engineering, 31*(5), 04014082. doi:10.1061/(ASCE)ME.1943-5479.0000323

Blazewicz, J., Lenstra, J. K., & Kan, A. H. G. R. (1983). Scheduling Subject to Resource Constraints: Classification and Complexity. *Discrete Applied Mathematics, 5*(1), 11-24. doi:https://doi.org/10.1016/0166-218X(83)90012-4

Browning, T. R., & Yassine, A. A. (2010). Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited. *International Journal of Production Economics, 126*(2), 212-228. doi:https://doi.org/10.1016/j.ijpe.2010.03.009

Chen, P.-H., & Shahandashti, S. (2009). Hybrid of Genetic Algorithm and Simulated Annealing for Multiple Project Scheduling with Multiple Resource Constraints. *Automation in Construction, 18*, 434-443. doi:10.1016/j.autcon.2008.10.007

Cheng, M.-Y., & Prayogo, D. (2014). Symbiotic Organisms Search: A New Metaheuristic Optimization Algorithm. *Computers & Structures, 139*, 98-112. doi:https://doi.org/10.1016/j.compstruc.2014.03.007

Gonçalves, J. F., Mendes, J. J. M., & Resende, M. G. C. (2008). A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem. *European Journal of Operational Research, 189*(3), 1171-1190. doi:https://doi.org/10.1016/j.ejor.2006.06.074

Habibi, F., Barzinpour, F., & Sadjadi, S. (2018). Resource-Constrained Project Scheduling Problem: Review of Past and Recent Developments. *Journal of Project Management, 3*, 55-88. doi:10.5267/j.jpm.2018.1.005

Hartmann, S., & Briskorn, D. (2010). A Survey of Variants and Extensions of The Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research, 207*(1), 1-14. doi:https://doi.org/10.1016/j.ejor.2009.11.005

Herroelen, W., & Leus, R. (2005). Identification and Illumination of Popular Misconceptions about Project Scheduling and Time Buffering in A Resource-Constrained Environment. *Journal of the Operational Research Society, 56*(1), 102-109. doi:10.1057/palgrave.jors.2601813

Homberger, J. (2007). A Multi-Agent System for the Decentralized Resource-Constrained Multi-Project Scheduling Problem. *International Transactions in Operational Research, 14*(6), 565-589. doi:https://doi.org/10.1111/j.1475-3995.2007.00614.x

Kannimuthu, M., Raphael, B., Ekambaram, P., & Kuppuswamy, A. (2020). Comparing Optimization Modeling Approaches for the Multi-Mode Resource-Constrained Multi-Project Scheduling Problem. *Engineering, Construction and Architectural Management, 27*(4), 893-916. doi:10.1108/ECAM-03-2019-0156

Kolisch, R. (1996). Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation. *European Journal of Operational Research, 90*(2), 320-333. doi:https://doi.org/10.1016/0377-2217(95)00357-6

Kolisch, R., & Drexl, A. (1997). Local Search for Nonpreemptive Multi-Mode Resource-Constrained Project Scheduling. *Lie Transactions, 29*(11), 987-999. doi:10.1023/A:1018552303415

Kolisch, R., & Hartmann, S. (1999). Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms and Applications* (pp. 147-178). Boston, MA: Springer US.

Kolisch, R., & 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:https://doi.org/10.1016/S0377-2217(96)00170-1

Kurtulus, I., & Davis, E. W. (1982). Multi-Project Scheduling: Categorization of Heuristic Rules Performance. *Management Science, 28*(2), 161-172. Retrieved from https://EconPapers.repec.org/RePEc:inm:ormnsc:v:28:y:1982:i:2:p:161-172

Li, F., Xu, Z., & Li, H. (2021). A Multi-Agent Based Cooperative Approach to Decentralized Multi-Project Scheduling and Resource Allocation. *Computers & Industrial Engineering, 151*, 106961. doi:https://doi.org/10.1016/j.cie.2020.106961

Li, K. Y., & Willis, R. J. (1992). An Iterative Scheduling Technique for Resource-Constrained Project Scheduling. *European Journal of Operational Research, 56*(3), 370-379. doi:https://doi.org/10.1016/0377-2217(92)90320-9

Linyi, D., & Yan, L. (2007, 15-19 Dec. 2007). *A Particle Swarm Optimization for Resource-Constrained Multi-Project Scheduling Problem.* Paper presented at the 2007 International Conference on Computational Intelligence and Security (CIS 2007).

Lova, A., & Tormos, P. (2001). Analysis of Scheduling Schemes and Heuristic Rules Performance in Resource-Constrained Multiproject Scheduling. *Annals of Operations Research, 102*(1), 263-286. doi:10.1023/A:1010966401888

Maroto, C., Tormos, P., & Lova, A. (1999). The Evolution of Software Quality in Project Scheduling. In J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms and Applications* (pp. 239-259). Boston, MA: Springer US.

MMLIB. Retrieved from https://www.projectmanagement.ugent.be/research/data

MPSPLib. Retrieved from http://www.mpsplib.com/index.php

Özdamar, L., & Ulusoy, G. (1995). A Survey on the Resource-Constrained Project Scheduling Problem. *Lie Transactions, 27*, 574-586. doi:10.1080/07408179508936773

Payne, J. H. (1995). Management of Multiple Simultaneous Projects: a State-of-the-Art Review. *International Journal of Project Management, 13*(3), 163-168. doi:https://doi.org/10.1016/0263-7863(94)00019-9

Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M. (1969). Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science, 16*(1), 93-108. doi:10.1287/mnsc.16.1.93

PSPLIB. Retrieved from http://www.om-db.wi.tum.de/psplib/getdata_mm.html

RCMPSPLIB. Retrieved from https://www.eii.uva.es/elena/RCMPSPLIB.htm

Sonmez, R., & Gürel, M. (2016). Hybrid Optimization Method for Large-Scale Multimode Resource-Constrained Project Scheduling Problem. *Journal of Management in Engineering, 32*(6), 04016020. doi:10.1061/(ASCE)ME.1943-5479.0000468

Sonmez, R., & Uysal, F. (2015). Backward-Forward Hybrid Genetic Algorithm for Resource-Constrained Multiproject Scheduling Problem. *Journal of Computing in Civil Engineering, 29*(5), 04014072. doi:10.1061/(ASCE)CP.1943-5487.0000382

Van Peteghem, V., & Vanhoucke, M. (2014). An Experimental Investigation of Metaheuristics for the Multi-Mode Resource-Constrained Project Scheduling Problem on New Dataset Instances. *European Journal of Operational Research, 235*(1), 62-72. doi:https://doi.org/10.1016/j.ejor.2013.10.012

Vázquez, E. P., Calvo, M. P., & Ordóñez, P. M. (2015). Learning Process on Priority Rules to Solve the Rcmpsp. *Journal of Intelligent Manufacturing, 26*(1), 123-138. doi:10.1007/s10845-013-0767-5

Villafáñez, F., López-Paredes, A., & Pajares, J. (2014). *From the Rcpsp to the Drcmpsp : Methodological Foundations*.

Villafáñez, F., Poza, D., López-Paredes, A., Pajares, J., & Olmo, R. d. (2019). A Generic Heuristic for Multi-Project Scheduling Problems with Global and Local Resource Constraints (Rcmpsp). *Soft Computing, 23*(10), 3465-3479. doi:10.1007/s00500-017-3003-y

Wauters, T., Kinable, J., Smet, P., Vancroonenburg, W., Vanden Berghe, G., & Verstichel, J. (2014). The Multi-Mode Resource-Constrained Multi-Project Scheduling Problem. *Journal of Scheduling, 19*, 1-13. doi:10.1007/s10951-014-0402-0

Yang, B., Geunes, J., & O'Brien, W. (2001). *Resource-Constrained Project Scheduling: Past Work and New Directions1*.

Zhang, H. (2012). Ant Colony Optimization for Multimode Resource-Constrained Project Scheduling. *Journal of Management in Engineering, 28*(2), 150-159. doi:10.1061/(ASCE)ME.1943-5479.0000089