AUTHOR(S): AYODELE, M., MCCALL, J. and REGNIER-COUDERT, O.

TITLE: Estimation of distribution algorithms for the multi-mode resource constrained project scheduling problem.

YEAR: 2017

Publisher citation: AYODELE, M., MCCALL, J. and REGNIER-COUDERT, O. 2017. Estimation of distribution algorithms for the multi-mode resource constrained project scheduling problem. In Proceedings of the IEEE congress on evolutionary computation (CEC 2017), 5-8 June 2017, San Sebastian, Spain. New York: IEEE [online], pages 1579-1586. Available from: https://doi.org/10.1109/CEC.2017.7969491

Estimation of Distribution Algorithms for the Multi-Mode Resource Constrained Project Scheduling Problem

Mayowa Ayodele, John McCall, Olivier Regnier-Coudert
Robert Gordon University
Aberdeen, Scotland
Email: {m.m.ayodele,j.mccall,o.regnier-coudert}@rgu.ac.uk

***Abstract*—Multi-Mode Resource Constrained Project Problem (MRCPSP) is a multi-component problem which combines two interacting sub-problems; activity scheduling and mode assignment. Multi-component problems have been of research interest to the evolutionary computation community as they are more complex to solve.**

Estimation of Distribution Algorithms (EDAs) generate solutions by sampling a probabilistic model that captures key features of good solutions. Often they can significantly improve search efficiency and solution quality.

Previous research has shown that the mode assignment sub-problem can be more effectively solved with an EDA. Also, a competitive Random Key based EDA (RK-EDA) for permutation problems has recently been proposed. In this paper, activity and mode solutions are respectively generated using the RK-EDA and an integer based EDA. This approach is competitive with leading approaches of solving the MRCPSP.

I. Introduction

Multi-component problems such as the Multi-Mode Resource Constrained Project Problem (MRCPSP) have been of research interest to the computational intelligence community in recent years [1], [2]. This class of problems consists of two or more sub-problems that cannot be solved in isolation. MRCPSP consists of the *Activity Scheduling* and *Mode Assignment* sub-problems. *Activity Scheduling* is a permutation based problem that requires assigning start and finish times to activities of a project such that precedence constraints are respected. The *Mode Assignment* problem defines the mode in which each activity of the project will be performed. A solution to this problem is the selection of a mode of execution for each activity such that resource constraints are respected.

Best practice regarding how to handle multi-component problems remains a question in the community. The MRCPSP has previously been solved with hybrid approaches. In [3], different algorithms were used for each sub-problem of the MRCPSP. The Genetic Algorithm (GA) was applied to the activity scheduling sub-problem while EDA was applied to the mode assignment sub-problem. Most existing approaches however use the same algorithm for both components but also apply some local search improvement methods. Existing approaches of solving the MRCPSP include EDAs [4], [5], Differential Evolution (DE) [6], GAs [7], [8], [9], [10], Particle Swarm Optimisation (PSO) [11] and Scatter Search (SS) [12]. Although there is considerably more research on GAs applied to MRCPSP than EDAs, the use of EDA for the *Mode Assignment* problem has been reported to improve on the GA [3]. Previous attempts to apply EDA to both sub-problems have resulted in algorithms with relatively poor performance [13]. This is due to the difficulty and complexity of using EDAs on permutation problems. In this paper, we apply RK-EDA [14], an EDA designed for permutation problems to the activity scheduling sub problem. This is combined with a previously published EDA approach to mode assignment [3] yielding a pure EDA approach that performs competitively across the board and is particularly strong on larger problems.

The rest of the paper is structured as follows. In Section II, a background to this study is presented, the MRCPSP is formulated and some existing approaches reviewed. Section III describes the proposed algorithm. Section IV presents the experimental configurations and parameter settings. Section V describes results and analysis. Section VI presents the conclusions and suggests directions for future research.

II. Background

A. Problem Formulation

The MRCPSP is formally defined as follows. A project consists of a set of activities labelled from 1,...,n where 1 and n are dummy start and finish activities. Activity $i, i \in [2, n]$ has predecessor(s) $Pred_i$ which suggests that i cannot be performed until every activity $h, h \in Pred_i$ has been completed. Each activity i must be performed in a mode $k \in [1, m_i]$, where m_i is the number of possible modes of i. Given that there are A renewable resources, each renewable resource r, $r \in [1, |A|]$ is available per period of time. The maximum per period availability of r is denoted by αmax_r. There are also B non-renewable resources that cannot be replenished once used up. The overall availability of the non-renewable resource l, $l \in [1, |B|]$, denoted by βmax_l, must therefore not be exceeded while executing the project. Each mode of execution k of an activity i is composed of an integer vector of renewable resources $(\alpha_{i,k,1}, ..., \alpha_{i,k,|A|})$, an integer vector of non-renewable resources $(\beta_{i,k,1}, ..., \beta_{i,k,|B|})$ and the associated duration/execution time $t_{i,k}$.

The aim of the MRCPSP is to select exactly one mode of execution for each activity subject to resource and precedence

978-1-5090-4601-0/17/$31.00

constraints. This is such that makespan is minimised. We formulate the MRCPSP as follow.

Minimise ft_n subject to:

$$\forall\ i \in [1, n],\ st_i \geq ft_h\ \forall\ h \in Pred_i \tag{1}$$

Let C_p be the set of activities being executed during time period [p-1,p], then

$$\sum_{i \in C_p} \alpha_{i,k_i,r} \leq \alpha max_r\ \forall\ r, r \in [1, |A|], \forall\ p \tag{2}$$

$$\sum_{i=1}^{n} \beta_{i,k_i,l} \leq \beta max_l\ \forall\ l, l \in [1, |B|] \tag{3}$$

We denote the start and finish times of activity i by st_i and ft_i respectively. The precedence constraint is presented in (1) while the renewable and non-renewable resource constraints are respectively presented in (2) and (3). In (2) and (3), k_i is the allocated mode of i and can only be one of the predefined modes of i. Also, $\alpha_{i,k_i,r}$ and $\beta_{i,k_i,l}$ are respectively the amount of renewable resource r and non-renewable resource l required by activity i performed in mode k_i.

B. Genotype to Phenotype Translation

In previous research on MRCPSP, a permutation based representation referred to as Activity List (AL) is often used for the *Activity Scheduling* sub-problem while an integer based representation called Mode List (ML) is used for the *Mode Assignment* sub-problem [13]. For *Activity Scheduling*, there are some researches based on the Random Key (RK) representation [12], [10]. With RK, floating point values are used to represent each activity of a project. These values are then sorted in ascending order to yield an execution order for the activities. AL has previously been considered superior to RK until the research in [15] that shows that RK can also perform well. RK has since been used in some of the leading approaches of solving the MRCPSP [13]. In this paper, we respectively use the RK and ML representations for the *Activity Scheduling* and *Mode Assignment* problems.

The Schedule Generation Scheme (SGS) is used to translate genotype to phenotype by creating a schedule from a solution representation. SGS can either be serial or parallel [16]. In serial SGS, each job is scheduled at the earliest precedence and resource feasible time. However, at a selected time, the parallel SGS attempts to perform as many activities as can be performed subject to resource and precedence constraints. The difference between both methods is that the serial SGS performs *activity-incrementation* while the parallel SGS performs *time-incrementation* [16]. Serial SGS is more frequently used as the parallel SGS is sometimes unable to reach the optimal makespan [17]. We therefore use the serial SGS, which schedules each activity of a project at the earliest precedence and resource feasible time. A formal description of the SGS is presented in [16].

It is however common practice to embed some makespan improvement procedures into the standard SGS. The SGS has been improved with techniques such as Multi Mode Forward Backward Improvement (MM-FBI) [8]. It has also been improved with procedures to improve infeasible solutions as well feasible ones and referred to as Multi-mode Serial Schedule Generation Scheme (MSSGS) [5]. Another improved version of the SGS is presented in [10], the authors embedded a mode improvement method (described in Alg. 3) into the SGS. The resulting procedure is referred to as the extended SGS.

C. Literature Review

Many meta-heuristics applied to MRCPSP rely on a lot of local search and other improvement procedures to optimise their performance. A review of these algorithms [13] has shown that algorithms that use more of these additional methods present better results than those that do not. It is particularly common to embed a local search procedure in the SGS. We briefly outline some approaches of solving the MRCPSP.

SS [12] presents the best results on common MRCPSP problem sets. It relies on a lot of additional methods, it uses three improvement and some additional two local search methods. The authors proposed the feasibility improvement, critical path improvement and work content improvement methods. The feasibility improvement method tries to improve solutions that are infeasible with regards to non-renewable resources and therefore requires the continuous evaluation of the Excess Resource Requirement (ERR) (described in eq. (6) of mode solutions. The critical path and work content improvement respectively improve the critical path and the amount of resources required to perform a job. Although SS uses the highest number of improvement procedures, its strength lies in the ability to apply improvement methods based on the property of a problem instance. The SS also uses the mode improvement procedure and the multi-mode left shift in [7].

Furthermore, the most common stopping criteria for measuring the performance of algorithms is based on the number of schedules generated. For algorithms such as the Differential Algorithm (DE) in [6] that do not use any form of local search, this is set as the number of fitness evaluations. Lova et al [8] however identified that the computational effort of one improvement pass is greater than what is needed to generate two SGSs. For this reason, the number of schedules is calculated by dividing the sum of the number of times each activity of the project has obtained a feasible start/finish time by the total number of activities. This allows for a fairer comparison amongst algorithms. Most existing approaches adopt this method. In the SS [12] however, some other methods of evaluating a solution which correlates strongly with the makespan are used. For instance, to determine whether to execute the critical path and work content improvement methods, the critical sequence lower bound (CSLB) of solutions are calculated. CSLB together with Critical Path Lower Bound (CPLB) and Resource Based Lower Bound (RBLB) are three evaluation methods of an MRCPSP solution [18]. The computational cost of executing each of these methods are

however comparable to executing the SGS. In addition to the CSLB, other evaluation methods used by the SS includes the evaluation of the critical path of solutions. These evaluation functions are repeated for every mode change made to a solution needing critical path improvement until it satisfies the set criteria. With this approach, only good quality solutions are scheduled with the SGS and only these count as part of the number of schedules generated. Comparing with other algorithms based on the same number of schedules therefore may not give a true account of the competitiveness of the SS. This is because it uses far more computation cost to arrive at its results. It is therefore difficult to directly compare SS because of the additional effort. So we do not do statistical comparison but present the reported quality results.

The GA has been the most popular of meta-heuristics applied to the MRCPSP. Some of the recent and competitive research on the application of GAs to the MRCPSP are as follows.

The GA in [19] is one of the most recent applications of GA on the MRCPSP. It uses a SAT solver approach in a similar way as the GA in [20]. The former is an improvement of the latter. They use the SAT solver to optimise activity solutions in a feasible manner. They also use a greedy search approach to improve feasibility in solutions. The GA in [19] incorporates some additional procedures so that the algorithm is more efficient and scales better to larger problems than [20].

The GA in [9] uses the conventional improvement of initial population procedure to reduce infeasibility with regards to non-renewable resources. It solves the MRCPSP as a bi-objective problem. It also uses a ranked-based fitness assignment method and a clustering approach to compute densities. This algorithm however does not scale well to the largest problems.

Furthermore, a Hybrid GA called (MM-HGA) is proposed in [8]. It uses an improvement method called MM-FBI, which is embedded in the SGS. This method is applied to every solution that is feasible with respect to non-renewable resources. The aim is to reduce the project completion time by changing the mode of execution of an activity creating an improvement in its start/finish time. MM-HGA also proposed the massive mutation which attempts to improve the feasibility of solutions that are non-renewable resource infeasible. This is done by changing the modes of execution of randomly selected activities until it is feasible or a maximum number of tries is reached. These methods make a significant improvement in the performance of the algorithm.

The Bi Population Genetic Algorithm (BPGA) in [10] is one of the most competitive algorithms in literature. It presents the best results amongst GAs applied to the MRCPSP. The major improvement method used in [10] is the mode improvement method described in Alg. 3. Although the mode improvement procedure does not guarantee an improvement in the makespan, its approach makes it easier for activities to run in parallel thereby encouraging a reduction in makespan. It is also more efficient than other schedule improvement procedures as it focuses on improving the finish time of one activity rather than the entire schedule. This method has also been used in [12] and [3] and has led to significant improvements in the quality of solutions produced. The mode improvement method is embedded in the SGS and referred to as the extended SGS.

Furthermore, a combination of GA and EDA has also been applied to the MRCPSP. The Bi Population Genetic Algorithm with Estimation of Distribution Algorithm (BPGA-EDA) [3] an algorithm created by hybridising the GA in [10] with an EDA is also one of the most competitive algorithms. Most of the methods used in [10] are used by BPGA-EDA. It was shown that the BPGA-EDA can outperform BPGA suggesting that the EDA generates better mode solutions than the GA. The performance of BPGA-EDA was also improved by using the extended SGS.

To the best of our knowledge, there are only two applications of EDAs to both sub-problems of the MRCPSP which are presented in [5] and [4]. The latter is an improvement of the former.

These EDAs use two probabilistic models to generate an MRCPSP solution. The probabilistic model for generating activity solutions is of size $n \times n$ while that of mode solutions is of size $n \times m$. To satisfy the mutual exclusivity constraint such that each activity appears in an ordering only once, they use the permutation-based Probability Generation Mechanism (PGM) [5]. This method entails re-calculating probabilities based on eligible activities. This is a necessary procedure for many permutation based EDAs in order to avoid duplicates in orderings.

Furthermore, existing applications of EDAs to the MRCPSP use the improved serial SGS, MSSGS. The MSSGS includes a procedure for tackling infeasibility as well as improving the finish times of activities in a solution. This method randomly selects the mode of an activity and changes it to that which improves its feasibility. The second aspect of the MSSGS attempts to improve the finish time of an activity in a feasible solution. It does this by changing its mode to that which improves its finish time without delaying the finish time of other activities. Checking each mode of execution of an activity in a bid to improve the solution imposes a significant addition to the computational cost of evaluating a solution. In addition, these EDAs also use the local search method called Multi-mode version Permutation-Based Local Search (MPBLS). This method attempts to make local improvements to the best solutions found at each generation. This is done by changing the mode of randomly selected activities as well as its priority.

In addition to all these methods, the EDA in [4] introduced a random walk local search. This is the principal difference between the two EDAs. For a selected activity, the local search selects a mode with the minimum ERR in resource infeasible solutions or mode with minimum duration in resource feasible solutions. Adding this extra local search method improves the quality of solutions produced by the former.

In this paper, the proposed approach generates both activity and mode solutions with EDAs. It uses RK-EDA proposed in [14], which is a competitive EDA for solving permutation

based problems, for generating new activity solutions. The probabilistic model for mode generation is similar to that proposed in [3]. Although we use the same size of probabilistic model for mode solutions as existing EDAs, we use a lighter weight model of size $n + 1$ for generating new activity solutions. Also, the proposed approach does not require the use of PGM or any similar procedure since RKs always produce permutation feasible solutions.

III. Proposed Approach

In this paper, we use the bi-population approach in [21] which is based on a population of left-justified schedules (POP_L) and another of right-justified schedules (POP_R). As shown in Alg. 1, we start by executing the conventional preprocessing procedure of Spreecher et al [22]. After this, POP_L is randomly generated. The conventional feasibility improvement of initial mode solutions [10] is executed to improve solutions in the initial POP_L. This procedure reduces the number of resource infeasible solutions in the initial population. This is done by changing the modes of randomly selected activities to those that reduce the infeasibility with respect to non-renewable resources. The solutions in this population are then evaluated using the SGS.

A selected population S which contains the best t_s (truncation size) solutions in POP_L is generated. Probabilistic models of activity solutions and mode solutions, PM_{act} and PM_{mod} are generated based on POP_L as shown in Sections III-A and III-B .

Probabilistic models LPM_{mod} and RPM_{mod} which are respectively used for generating mode solutions in POP_L and POP_R are both initialised with PM_{mod}. Probabilistic model RPM_{act} for generating activity solutions in POP_R is initialised with PM_{act}. Probabilistic model LPM_{act} for generating activity solutions in POP_L is however generated based on t_s most promising solutions in POP_R. In a similar way, POP_R is updated based on t_s most promising solutions in POP_L.

At each generation, the solution that generates the best schedule in POP_L, $Lbest$ is rescheduled as a right justified solution and set as an offspring of POP_R. In a similar way, the best solution in POP_R, $Rbest$ is rescheduled as a left justified solution and set as an offspring of POP_L. To produce the remaining population of right/left justified schedules, RPM_{act} and RPM_{mod} / LPM_{act} and LPM_{mod} are sampled to produce $RChild$ / $LChild$. $RChild$ / $LChild$ is evaluated using backward SGS/ forward SGS where activities are scheduled as late as possible/as early as possible within resource and precedence feasibility. Also, while the forward SGS or schedules activities in increasing order of their RKs, backward SGS schedules in decreasing order of RKs.

Note that when mode improvement is used, the extended SGS is used in place of the standard SGS.

Also, once each solution has been scheduled, the RKs of that solution are updated to respect the order in which the activities were performed. This is because the order depicted by the RKs would not be the same as the order of execution because of resource and precedence constraints. We therefore respectively rank the activities of solutions in POP_L and POP_R by start and finish times. To fulfil the normalisation step of the RK-EDA [14], the RK of each activity in the jth rank is set to $\frac{j-1}{n}$ where n is the number of activities in the project

Algorithm 1 Proposed EDA for the MRCPSP

The algorithmic details of the proposed approach is presented as follows.

1: execute preprocessing procedure
2: generate initial population POP_L
3: evaluate POP_L with SGS
4: select best $t_s < |POP_L|$ solutions to form S.
5: build probabilistic models PM_{act} and PM_{mod} from S
6: initialise LPM_{mod} = RPM_{mod} = PM_{mod}
7: initialise RPM_{act} = PM_{act}
8: **repeat**
9: set $Lbest$ as best solution in POP_L
10: **for** i = 1 to $|POP_L|$ **do**
11: **if** i = 1 **then**
12: set $RChild$ as the genome of $Lbest$
13: **else**
14: sample RPM_{act} and RPM_{mod} to produce $RChild$
15: apply backward SGS to $RChild$
16: **end if**
17: update POP_R with $RChild$
18: **end for**
19: build probabilistic model LPM_{act} from POP_R
20: Update RPM_{mod} with POP_R
21: set $Rbest$ as best solution in POP_R
22: **for** i = 1 to $|POP_R|$ **do**
23: **if** i = 1 **then**
24: set $LChild$ as the genome of $Rbest$
25: **else**
26: sample LPM_{act} and LPM_{mod} to produce $LChild$
27: apply SGS to $LChild$
28: **end if**
29: update POP_L with $LChild$
30: **end for**
31: build probabilistic model RPM_{act} from POP_L
32: update LPM_{mod} with POP_L
33: **until** stopping criteria satisfied
34: **return** overall best solution

A. Probabilistic Model of Activity Solutions

PM_{act} is based on the procedures of RK-EDA [14]. It contains $n+1$ values where each value $\mu_i \in [1, n]$ represents the mean of all RKs relating to activity i in S. The $n + 1^{th}$ value of PM_{act} is the variance. PM_{act} is described as follows.

Cooling rate c, as shown in ln. 1, is calculated such that its value reduces as the number of generations g increases. Note that maximum number of generations $MaxGen$ is calculated by dividing the maximum number of schedules allowed by the population size. To estimate the generational variance σ_g

Algorithm 2 Probabilistic Model for Activity Solutions

1: $c = 1 - \frac{g}{MaxGen}$
2: $\sigma_g = \sigma * c$
3: **for** i = 1 to n **do**
4: calculate μ_{act_i}
5: $PM_{act_i} = N(\mu_{act_i}, \sigma_g)$
6: **end for**

in ln. 2, c is multiplied by the initial variance σ. This approach increases exploration at earlier generations of the run and exploits more closer to the end. Note that σ is a parameter of the RK-EDA that needs to be set.

To generate a new activity solution, the RK of each activity i is generated by sampling a normal distribution based on the mean of RKs in S relating to activity i and variance σ_g. Note that the same value of σ_g is used to sample every RK in every solution at a particular generation.

B. Probabilistic Model for Mode generation

PM_{mod} is created based on the fraction of activities performed in certain modes of execution in population S. The probability that activity i will be performed in mode k; $PM_{mod_{i,k}}$ is calculated as $\frac{count(i,k)}{b}$ where $count(i,k)$ denotes the number of solutions in S where activity i is performed in mode k.

$$PM_{mod} = \begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nm} \end{pmatrix} \quad (4)$$

Note that the probabilistic model for mode generation uses a Population-Based Incremental Learning (PBIL) style where the model at each generation learns from the previous generation.

If we respectively represent p_{ik} values of PM_{mod} at generation g and $g-1$ by $p_{ik}(g)$ and $p_{ik}(g-1)$, each $p_{ik}(g)$ value will be estimated as shown in eq. (5).

$$p_{ik}(g) = (lr * p_{ik}(g)) + ((1 - lr) * p_{ik}(g-1)) \quad (5)$$

Probability values $p_{ik}(g)$ is updated by $p_{ik}(g-1)$ using a learning rate lr as shown in eq. (5). The probabilistic model is updated at the end each generation until the stopping criteria is met.

C. Mode Improvement

The mode improvement method [10] attempts to improve feasibility as well as the finish time of an activity. To achieve the first aim of improving feasibility with regards to non-renewable resources, there is a need to calculate the ERR of a mode solution μ. $ERR(\mu)$ is described in Eq. (6).

$$ERR(\mu) = \sum_{x=1}^{|B|}(max(0, \sum_{i=1}^{n} \beta_{i,k_i,l} - \beta max_l)) \quad (6)$$

$ERR(\mu)$ is set to zero if μ is feasible. Otherwise, it is set to the difference between the sum of non-renewable resource requirements $\beta_{i,k_i,l}$ $\forall\ l \in [1, |B|]$, for each activity i in their modes of execution k_i, and availability βmax_l.

The mode improvement procedure is described as follows.

Algorithm 3 Mode Improvement Method

1: **for** j = 1 to $|m_i|$ **do**
2: set new mode solution μ' to existing mode solution μ
3: compute $ERR(\mu)$
4: **if** $k_i \neq j$ **then**
5: set new mode $k_i{}' = j$
6: update mode solution μ' with $k_i{}'$
7: compute $ERR(\mu')$ (see Eq. (6))
8: **if** $ERR(\mu') \leq ERR(\mu)$ **then**
9: compute $f_i{}'$
10: **if** $f_i{}' < f_i$ **then**
11: $\mu = \mu'$
12: $ERR(\mu) = ERR(\mu')$
13: $f_i = f_i{}'$
14: **end if**
15: **end if**
16: **end if**
17: **end for**

For an activity i, a mode solution μ' is generated by replacing k_i in μ with another mode $k_i{}'$. The $ERRs$ of the existing mode solution μ and the new mode solution μ' are calculated. If $ERR(\mu)$ is not higher than $ERR(\mu')$, the procedure executes the next stage. This stage compares the finish times of i based on mode solutions μ' and μ and are respectively denoted by $f_i{}'$ and f_i. If $f_i{}'$ is less than f_i, mode solution μ, $ERR(\mu)$ and f_i are respectively set to μ', $ERR(\mu')$ and $f_i{}'$.

IV. Experimental Settings

In this section, we present the problem sets and parameter settings used in this paper.

A. Problem sets

The most common MRCPSP problem sets in literature are J10, J20 and J30 from the PSPLIB [23]. They respectively require the scheduling of 10, 20 and 30 activities. They also respectively consist of 536, 554 and 552 feasible instances. The proposed algorithm is compared with the BPGA and BPGA-EDA based on J10, J20 and J30 as done in [3]. In addition to these problem sets, PSPLIB also contains J12, J14, J16 and J18. They are respectively made up of 547, 551, 550 and 552 problem instances as well as 12, 14, 16 and 18 activities to be scheduled. With the exception of J30, existing EDAs have been applied to the entire PSPLIB instances. The EDA in [5] does not present results for J30 while the EDA in [4] was not able to achieve 100% feasibility on J30. Comparison with existing EDAs is therefore done based on J10-J20.

More recently, certain disadvantages have been identified with the PSPLIB problem sets. One is the fact that modes of execution can be eliminated by executing the preprocessing technique, simplifying the problems significantly. Also, some of the instances do not have any feasible solution. To avoid these disadvantages, MMLIB problem sets which also consist of larger problem instances were created [13]. In this paper, we use the MMLIB50 and MMLIB100 that respectively require the scheduling of 50 and 100 activities. They consist of 540 instances each. Comparison with a broader range of algorithms was based on the J10, J20, J30, MMLIB50 and MMLIB100. We are able to get results of running several algorithms on these problem sets from the review in [13].

B. Parameter settings

RK-EDA which is used for generating activity solutions and the integer based EDA used for generating mode solutions require two parameters in common which are population size p_s and truncation size t_s . The integer based EDA in addition requires a learning rate lr while RK-EDA requires a variance parameter σ. The parameters used in this paper are presented in Table I.

TABLE I
PARAMETER SETTINGS

Parameter	**PSPLIB**	**MMLIB**
Population Size (p_s)	$\frac{3000}{n}$	100
Truncation Size (t_s)	$0.1 \times p_s$	$0.1 \times p_s$
Variance (σ)	Minimum of 0.2 and $\frac{3}{n}$	0.05
Mode improvement rate	$0.0, 0.2$	$0.0, 0.2$
Learning rate	0.8	0.8
Number of evaluations	5000	5000
Number of Runs	10	10

Values presented in Table I are derived based on preliminary tests. These test reveal that different p_s and σ values are required for the MMLIB and PSPLIB problem sets. This may be attributed to the difference in formulation of both libraries.

We have set the limit on the number of evaluations to $5,000$ as this is the most frequently used stopping criteria [13].

The most common performance measure using number of schedules as stopping criteria is the average percentage deviation from optimal (APD_O) and is calculated as follows.

$$APD_O = \frac{\sum_{i=0}^{n}(((bestFit - optimal)/optimal) * 100)}{n} \quad (7)$$

In eq. (7), $bestFit$ is the fitness of the best solution generated by the algorithm. The value of $optimal$ is either the reported optimal values for J10 and J20 or the critical path based lower bound (CPBLB) for J30, MMLIB50 and MMLIB100. This is the usual practice in previous research especially the review in [13]. The CPBLB is estimated using the critical path based on the modes with the least durations. Average percentage deviation from CPBLB (APD_C) is therefore calculated by replacing optimal with the CPBLB in Eq. (7)

Since several problem instances in the datasets of PSPLIB have similar characteristics, it is common practice to run algorithms only once across all problems. Irrespective of the similarity in problem instances, it was shown in [3] that several runs are still needed to capture the true performance of non-deteministic algorithms applied to these datasets. In this paper, results are averaged over ten runs. We use student t-test to test for statistical significance

Furthermore, local search methods have been reported to significantly improve the performance of meta-heuristics [13]. The proposed approach also considers the mode improvement method which is one of the most efficient improvement method shown to improve the performance of most of the leading algorithms. These includes the SS, BPGA and BPGA-EDA. Since the proposed approach uses a similar approach as the BPGA and BPGA-EDA, we will be comparing directly with these algorithms with and without the use of the mode improvement local search method.

Mode improvement rate relates to the number of activities that improved when scheduling a project. Based on preliminary results, 0.2 mode improvement rate gave the best the results. Also, results on larger problems showed that the effect of mode improvement on the largest problems is not as significant as others. For this reason, rather than use the extended SGS at each generation, we also tried to use it at every other generation. Moreover, the mode improvement rate still incurs additional computation such as estimating ERR.

V. RESULTS AND DISCUSSION

In this section, the proposed approach is compared with other leading and recently published results for the MRCPSP. Results for J10-J20 are based on APD_O while J30, MMLIB50 and MMLIB100 are based on APD_C. All are averaged across ten runs.

A. Comparing the proposed approach with BPGA-EDA and BPGA

To be able to compare based on several runs for results based on the standard SGS and extended SGS, results for the BPGA are obtained from [3].

In Table II, we present the APD_O averaged over ten runs (alongside the standard deviation) for the BPGA, BPGA-EDA and the proposed EDA. These results are based on the standard SGS (i.e without the use of *mode improvement*).

TABLE II
RESULTS BASED ON SGS - AVERAGE APD_O (STANDARD DEVIATION)

Problem sets	BPGA	BPGA-EDA	Proposed EDA
J10	0.61 (0.08)	**0.20 (0.04)**	**0.19 (0.04)**
J20	2.34 (0.05)	1.60 (0.07)	**1.05 (0.08)**
J30	17.89 (0.18)	15.06 (0.07)	**14.52 (0.07)**

In Table II, the best results as well as results that are not significantly different from the best are presented in bold. The proposed EDA is significantly better than the BPGA on J10, J20 and J30 problem sets. The performance of the EDA is

also better than the BPGA-EDA with no statistical difference on J10 but significantly better result on J20 and J30.

The performance of the EDA in comparison with BPGA-EDA show that RK-EDA can produce better quality of activity solutions than GA. Since the Proposed EDA uses similar procedures as the BPGA and BPGA-EDA, Its relative performance also show that EDA can outperform the GA on both components of the MRCPSP.

Since previous research has shown that the mode improvement method can significantly improve the performance of BPGA and BPGA-EDA, we also compare performance based on the use of this improvement method. Table III presents the APD_O (and Standard deviation) averaged across ten runs of the algorithms. The results presented are based on the use of *mode improvement* local search (i.e. solutions are scheduled with the extended SGS [10]).

TABLE III
RESULTS BASED ON EXTENDED SGS - AVERAGE APD_O/APD_C (STANDARD DEVIATION)

Problem sets	BPGA	BPGA-EDA	Proposed EDA
J10	0.05 (0.02)	**0.03 (0.02)**	0.06 (0.02)
J20	0.88 (0.04)	0.69 (0.04)	**0.64 (0.04)**
J30	14.41 (0.05)	13.87 (0.07)	**13.66 (0.07)**

Based on Table III, there is no difference between the performance of the EDA and BPGA on J10 but the EDA performs significantly better than the BPGA on J20 and J30. The EDA is however not statistically worse than the BPGA-EDA on J10 but better on J20 and J30.

Results for BPGA and BPGA-EDA in Tables III and III are retrieved from results presented in [3].

B. Comparing the proposed EDA with existing EDAs

Since existing EDAs use an improved SGS (MSSGS), we compare them with the proposed EDA based on extended SGS. Table IV presents the result of the EDAs in [5] and [4] as well as the proposed EDA.

TABLE IV
RESULTS COMPARING THE PROPOSED EDA WITH OTHER EDAS: AVERAGE APD_O

Problem Sets	Proposed EDA	EDA [5]	EDA [4]
J10	**0.06**	0.12	0.09
J12	0.17	0.14	**0.12**
J14	**0.28**	0.43	0.36
J16	**0.40**	0.59	0.42
J18	**0.48**	0.90	0.85
J20	**0.64**	1.28	1.09
J30	**13.95**	15.55	-

Although the results of the EDA in [4] are based on several runs, information about variance are however not presented. Similarly, there are no information on variance provided in [5]. We are therefore unable to test for statistical significance. Apart from J12, the proposed EDA shows better performance than the EDA in [5] or [4].

This is a competitive performance by the proposed EDA considering the fact that it uses less improvement methods and also a lighter weight model. The probabilistic model used by this EDA (of size $n + 1$) for activity solutions is much smaller than that of existing EDAs (of size $n \times n$)

C. Comparing the proposed EDA with leading or recent algorithms

As previously noted, results presented based on applications of meta-heuristics to the MRCPSP are often based on a single run. This includes the review in [13]. However, to be able to compare with other algorithms on the MMLIB datasets as well as PSPLIB, most results are retrieved from [13]. Some more recent algorithms not captured in the review such as algorithms presented in [4], [24] and [19] are however retrieved from the authors' papers. Apart from results presented in [4] which are averaged across many runs, the results for the BPGA based on a single run as presented by its authors are used. Some more literature on MRCPSP exist but we have not been able to compare with them because they have presented results based on different criteria such as CPU time.

In Table V, results based on improved SGS are appended a "✓" while those that are based on standard SGS are appended a "×". Also, missing results or problem set for which an algorithm has not been able to attain feasibility for its instances are represented by "-". Although, we show SS in the table, we do not directly compare the proposed method with it. This is because we have shown that it contains many other evaluations of complete solutions in addition to the SGS, which inhibits fair comparison.

We present results for the proposed EDA based on the standard SGS (esgs = 0.0), standard SGS/extended SGS and at every generation (esgs = 0.5) and extended SGS (esgs = 1.0) only. Of the three configurations, esgs = 0.5 presents the best results especially on larger problems. The esgs = 1.0 configuration however presents the best results on smaller problems. We show that the performance of the proposed EDA approach when esgs = 0.5 is the most competitive on the largest problem, MMLIB100. This gives an indication that larger problem require less of the improvement procedure. This may be because the mode improvement only encourages more activities to run in parallel [10] but does not guarantee an improvement in makespan. The mode improvement method helps to explore the search space better in smaller problems but do not scale well to larger problems. Moreover, as shown in [12], certain problem instances benefits more from some improvement techniques than others.

Moreover, compared to the algorithms that do not use any schedule improvement procedure (GA in [9] and DE in [6]), the proposed EDA without schedule improvement presents much better results. Overall, BPGA presents the best results on J10 and J20. However, the proposed EDA when esgs = 0.5 presents the best performance on J30, MMLIB50 and MMLIB100. This is consistent with the behaviour of RK-EDA [14] scaling better to larger problems.

Summarily, we show that the proposed EDA which uses two univariate EDAs is one of the most competitive algorithms for solving the MRCPSP. A multi-variate EDA capturing linkage

TABLE V
RESULTS BASED ON EXTENDED SGS - AVERAGE APD_O / APD_C

Algorithms	Improved SGS	J10	J20	J30	MMLIB50	MMLIB100
MMHGA [8]	✓	0.04	0.89	14.58	28.59	31.01
DE [6]	×	0.74	1.62	15.43	32.46	36.87
BPGA [10]	✓	**0.01**	**0.57**	13.75	27.12	29.55
GA [9]	×	0.12	1.51	16.16	32.47	40.22
GA [20]	✓	0.07	0.80	14.44	-	-
EDA [5]	✓	0.09	1.28	15.55	31.95	38.55
EDA [4]	✓	0.09	1.09	-	-	-
LS [24]	✓	-	-	-	33.02	44.11
GA [19]	✓	0.07	0.94	14.62	29.42	34.60
BPGA-EDA [3]	✓	0.03	0.69	13.87	27.96	31.38
Proposed EDA (esgs = 0.0)	×	0.19	1.05	14.52	28.27	28.94
Proposed EDA (esgs = 0.5)	✓	0.09	0.71	**13.65**	**26.20**	**27.47**
Proposed EDA (esgs = 1.0)	✓	0.06	0.64	13.66	26.52	28.46
SS [12]	✓	0.00	0.32	13.66	25.45	26.51

information between the sub-problems of the MRCPSP may however produce even better results.

VI. CONCLUSIONS

In this paper, we proposed an EDA approach motivated by BPGA-EDA which combines GA and EDA to solve the MRCPSP. The proposed EDA uses RK-EDA, one of the leading EDAs for permutation problems to solve the *Activity Scheduling* problem rather than the GA. Using EDA to generate mode solutions have previously showed improved performance. In this paper, we show that RK-EDA applied for generating activity solutions can present even further improvements.

The proposed EDA showed better performance than existing EDAs on most of the problems sets used in this paper. It also uses more efficient methods and a lighter weight model.

The proposed EDA is also compared with state-of-the-art approaches of solving the MRCPSP and shown to be competitive with these approaches. Its performance is most competitive on the larger problem sets (MMLIB50 and MMLIB100).

Application of multi-variate EDA to the MRCPSP is recommended for further studies.

REFERENCES

[1] M. R. Bonyadi, L. Barone, and Z. Michalewicz, "The travelling thief problem: the first step in the transition from theoretical problems to realistic problems," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico*, 2013.

[2] M. R. Bonyadi and Z. Michalewicz, "Evolutionary computation for real-world problems," in *Challenges in Computational Statistics and Data Mining*. Springer, 2016, pp. 1–24.

[3] M. Ayodele, J. McCall, and O. Regnier-Coudert, "BPGA-EDA for the multi-mode resource constrained project scheduling problem," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 3417–3424.

[4] O. S. Soliman and E. A. Elgendi, "A hybrid estimation of distribution algorithm with random walk local search for multi-mode resource-constrained project scheduling problems," *arXiv preprint arXiv:1402.5645*, 2014.

[5] L. Wang and C. Fang, "An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem," *Computers & Operations Research*, vol. 39, no. 2, pp. 449–460, 2012.

[6] N. Damak, B. Jarboui, P. Siarry, and T. Loukil, "Differential evolution for solving multi-mode resource-constrained project scheduling problems," *Computers & Operations Research*, vol. 36, no. 9, pp. 2653–2659, 2009.

[7] S. Hartmann, "Project scheduling with multiple modes: a genetic algorithm," *Annals of Operations Research*, vol. 102, no. 1-4, pp. 111–135, 2001.

[8] A. Lova, P. Tormos, M. Cervantes, and F. Barber, "An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes," *International Journal of Production Economics*, vol. 117, no. 2, pp. 302–316, 2009.

[9] S. Elloumi and P. Fortemps, "A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 31–41, 2010.

[10] V. V. Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, 2010.

[11] H. Zhang, C. Tam, and H. Li, "Multimode project scheduling based on particle swarm optimization," *Computer-Aided Civil and Infrastructure Engineering*, vol. 21, no. 2, pp. 93–103, 2006.

[12] V. Van Peteghem and M. Vanhoucke, "Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem," *Journal of Heuristics*, vol. 17, no. 6, pp. 705–728, 2011.

[13] ——, "An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances," *European Journal of Operational Research*, vol. 235, no. 1, pp. 62–72, 2014.

[14] M. Ayodele, J. McCall, and O. Regnier-Coudert, "RK-EDA: A novel random key based estimation of distribution algorithm," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 849–858.

[15] V. Valls, M. Laguna, P. Lino, A. Pérez, and S. Quintanilla, "Project scheduling with stochastic activity interruptions," in *Project scheduling*. Springer, 1999, pp. 333–353.

[16] R. Kolisch and S. Hartmann, *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*. Springer, 1999.

[17] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation," *European Journal of Operational Research*, vol. 90, no. 2, pp. 320–333, 1996.

[18] M. Vanhoucke, "Integrated project management sourcebook."

[19] M. Vanhoucke and J. Coelho, "An approach using SAT solvers for the RCPSP with logical constraints," *European Journal of Operational Research*, vol. 249, no. 2, pp. 577–591, 2016.

[20] J. Coelho and M. Vanhoucke, "Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers," *European Journal of Operational Research*, vol. 213, no. 1, pp. 73–82, 2011.

[21] D. Debels and M. Vanhoucke, "A bi-population based genetic algorithm for the resource-constrained project scheduling problem," in *International Conference on Computational Science and Its Applications*. Springer, 2005, pp. 378–387.

[22] A. Sprecher, S. Hartmann, and A. Drexl, "An exact algorithm for project scheduling with multiple modes," *Operations-Research-Spektrum*, vol. 19, no. 3, pp. 195–203, 1997.

[23] R. Kolisch and A. Sprecher, "PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[24] M. J. Geiger, "A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem," *European Journal of Operational Research*, 2016.