Multiple Trajectory Search for the Resource-Constrained Project Scheduling Problem

Lin-Yu Tseng
Department of Computer Science and Communication Engineering
Providence University
Taichung, Taiwan
lytseng@pu.edu.tw

Kuan-Cheng Lin
Department of Computer Science and Engineering
National Chung Hsing University
Taichung, Taiwan
kclin@cs.nchu.edu.tw

***Abstract*—The resource-constrained project scheduling problem is one of the most important scheduling problems and has attracted much attention of researchers. In this study, we proposed a new metaheuristic called the multiple trajectory search for solving this problem. The multiple trajectory search algorithm was previously proposed by us to solve the real-parameter optimization problems, both single-objective and multi-objective. And its performance was good as revealed by the ranking in the competitions held in 2008 and 2009 IEEE Congress on Evolutionary Computation. In this study, we arranged the multiple trajectory search algorithm to solve a combinatorial problem – the resource-constrained project scheduling problem. The experimental results show that the proposed method is competitive with other state-of-the-art methods, especially for the problem sets with 30 and 60 activities.**

Keywords- resource-constrained project scheduling problem; multiple trajectory search; peak crossover; forward-backward improvement

I. INTRODUCTION

The resource-constrained project scheduling problem (RCPSP) is an important scheduling problem and many researchers have devoted much effort solving it. Being an NP-hard problem, Alcaraz and Maroto [1] mentioned that the optimal solution could only be achieved by exact solution procedure in small projects, usually with number of activities less than 60 and with the project not highly resource-constrained. Therefore, heuristic and metaheuristic methods were designed to solve large and highly resource-constrained projects.

Some algorithms are exact and are based on the branch-and-bound strategy. Demeulemeester and Herroelen [2] developed a depth-first branching scheme with dominance criteria and the bounding rules. Brucker, Knust, Schoo and Thiele [3] presented a branch-and-bound algorithm whose branching scheme applied a set of conjunctions and disjunctions to pairs of activities.

Some algorithms are heuristic. They are briefly described in the following. Möhring, Schulz, Stork and Uetz [4] proposed a heuristic based on the Lagrangian relaxation and minimum cut computations. The heuristic methods based on priority rules can be divided into two classes: single-pass methods and multi-pass methods. The following studies presented single-pass methods: Boctor [5], Kolisch [6], Valls, Perez and Quintanilla [7], Ulusoy and Özdamar [8], and Özdamar and Ulusoy [9]. Multi-pass methods were presented in Ulusoy and Özdamar [8] and Boctor [5]. The forward-backward methods were proposed in Li and Willis [10] and Özdamar and Ulusoy [9].

Some metaheuristic algorithms were also proposed for RCPSP. They include the genetic algorithm, the simulated annealing, the tabu search, the ant colony optimization, the path relinking, and hybrid algorithms. The following studies incorporated genetic algorithms: Lee and Kim [11], Özdamar [12], Mori and Tseng [13], Alcaraz and Maroto [1], and Hartmann [14][15]. Some simulated annealing algorithms were proposed by Boctor [16], Lee and Kim [11], Cho and Kim [17], and Bouleimen and Lecocq [18]. Methods based on tabu search were presented in Valls, Quintanilla and Ballestin [19] and Artigues, Michelon and Reusser [20]. Merkle, Middendorf and Schmeck [21] presented an ant colony optimization by using the summation of the values in the pheromone set for this problem. Valls, Ballestin and Quintanilla [22] presented a simple technique named justification that can be applied in many methods to improve the quality of solution without generally requiring more computing time. They also designed a peak crossover operator within a hybrid genetic algorithm with justification [23].

The investigation of Hartmann and Kolisch [24] and its updated version conducted an elaborate study on state-of-the-art heuristic and metaheuristic methods. They presented performance comparisons among heuristic and metaheuristic methods in their study by applying these methods to different standard instance sets, namely J30, J60 and J120, generated by ProGen in the PSPLIB [25]. As shown by the latest experimental evaluation, metaheuristic methods outperformed heuristic methods.

In this study, we arranged the multiple trajectory search (MTS) algorithm that was previously proposed for continuous optimization to solve the RCPSP. The results obtained are comparable with the results obtained by other state-of-the-art methods.

The remainder of the paper is organized as follows. In Section II, the definition of the RCPSP is described. In Section III, the proposed MTS for the RCPSP is elaborated.

 ISBN: 978-1-61208-217-2

Section IV gives experimental results and Section V concludes the paper.

II. DEFINITION OF THE PROBLEM

The single-mode RCPSP considers each activity in the project having an operation mode only. All resources available in the project are renewable. The objective is to find a schedule of operation start times for activities, subject to the precedence constraints and resource constraints, such that the makespan of the project is minimized. The notations are defined as follows. The set of activities in the project is denoted by $\{0, 1, \ldots, n, n+1\}$, where 0 and $n+1$ are dummy activities and all other activities are non-dummy. The operation start time of activity j is denoted by S_j. The duration of activity j is denoted by d_j. The set of resources in the project is denoted by $\{1, 2, \ldots, K\}$. The capacity of resource k available in each time period during the process of the project is denoted by R_k. The quantity of resource demand of activity j to resource k is denoted by r_{jk}. $i \rightarrow j$ denotes that activity i is a predecessor of activity j. P_j denotes the set of all predecessors of activity j.

The project can be represented as an activity-on-node network by the precedence relations. Fig. 1 shows an example of a project of this problem. There is only one resource with capacity 5 in this example. d_j / r_{jk} above each node denotes the duration of activity and the demand of activity j to resource k. The dummy activities which have zero duration and no any resource demand are the single source and sink in the network.

In this study, solutions are represented in the form of a precedence-feasible activity list. When an activity list is given, the serial schedule generation scheme is applied to produce a schedule. We consider both of *forward scheduling* and *backward scheduling*. A *forward schedule* is a mapping, by this mapping, activity 0 to activity $n+1$ are mapped to a set of operation start times which are set to be as early as possible while satisfying the resource constraints. A *backward schedule* is also a mapping, by this mapping, activity $n+1$ to activity 0 are mapped to a set of operation start times which are set to be as late as possible while satisfying the resource constraints.

III. MULTIPLE TRAJECTORY SEARCH FOR RCPSP

The MTS was previously proposed by us to solve the single-objective real-value optimization problem [26] and the multi-objective real-value optimization problem [27]. In this section, the proposed MTS for the RCPSP is elaborated. The MTS consists of two phases. In the first phase, a set of seeds are found, and in the second phase, a region search method is applied to do the search beginning with each seed. A genetic local search algorithm is used both in the first phase and in the region search. We first introduce the MTS.

A. Multiple Trajectory Search

The MTS algorithm is described in the following.

Step1. Randomly generate the initial population and apply the genetic local search algorithm to find N good solutions.

Step2. Using these N good solutions as the initial population, apply the genetic local search algorithm again to find m seeds.

Step3. For each of these m seeds, apply the region search algorithm to it.

Step4. When the maximum number of evaluations is reached, output the best found solution.

Because most studies on the RCPSP used the maximum number of evaluations as the termination condition, this study also adopted it as the termination condition in order to compare the results with those of others methods. The first phase of the MTS consists of Step1 and Step2. One-tenth of the maximum number of evaluations is used in Step1 to globally find good solutions. Another one-tenth of the maximum number of evaluations is used in Step2 to find m seeds. Step3 and Step4 are the second phase. In the second phase, we apply the region search algorithm to each seed until the maximum number of evaluations is reached. In the following subsections, we introduce the region search algorithm and the genetic local search algorithm.

B. Region Search Algorithm

The region search begins with a seed. It first generates k solutions from the seed by randomly picking p activities and re-inserting them into the activity list. These k solutions form the initial population of the genetic local search algorithm. Then, it applies the genetic local search algorithm to search for better solutions. If the best solution found by the genetic local search algorithm is better than the original seed, the best solution found replaces the original seed, and the region search begins again with the new seed. If the best solution found is not better than the original seed, the value of p is set to $0.8 * p$. The value of 0.8 is determined based on empirical experiences. The region search terminates when the value of p reaches one-third of the original value of p.

C. Genetic Local Search Algorithm

In the genetic local search algorithm, we use a modified version of the peak crossover operator proposed by Valls, Ballestin and Quintanilla [23]. Also, based on the FBI (forward-backward improvement) [10], we proposed the FBI-WP (forward-backward improvement with perturbation) as the local search scheme. We now describe the genetic local search algorithm in the following.

Step1. The following Step2 to Step4 are executed #_of_generation times.

Step2. Randomly pair all chromosomes in the parent population, then apply modified peak crossover operator to each pair of parents. All child chromosomes are put in the child population.

 ISBN: 978-1-61208-217-2

Step3. Apply FBI-WP to each chromosome in the child population.

Step4. Apply 2-tournament selection to the parent population and the child population to produce the population for the next generation.

Next, let us explain the modified peak crossover operator. Let the parent chromosomes be *F* and *M*. Two resources r_1 and r_2 are randomly selected as the basis for calculating the peak. The average usage avg_r and the maximum usage max_r of the resource *r* are calculated, and a threshold t_r is defined as $t_r = avg_r + (max_r - avg_r)*0.5$. For *F* (or *M*), the time units where the resource usage is greater than this threshold are taken as the peak. Suppose that the crossover of *F* and *M* is to produce *S* and *D*, then *S* preserves the peak of *F* and the other activities come from *M*, and in a similar way, *D* preserves the peak of *M* and the other activities come from *F*.

Forward-backward improvement (FBI) was proposed by Li and Willis [10]. Basically, this scheme iteratively applies serial forward and backward scheduling until no further improvement in the makespan can be obtained. The activity finish times of a forward schedule determine the activity priorities when we produce the next backward schedule. And the activity start times of a backward schedule determine the activity priorities when we produce the next forward schedule. Valls, Ballestin and Quintanilla [22] showed that the FBI is rather effective. In this study, we add a perturbation mechanism to the FBI. If two consecutive applications of the FBI does not improve the solution, we perturb (mutate) the solution before continue to apply the FBI. In the perturbation, we re-insert randomly chosen activities *num_of_mutate* times. The application of the FBI will be terminated only when the number of no improvement reaches a pre-specified value (*Max_No_Improve*).

IV. Experimental Results

The proposed MTS for the RCPSP was implemented in C++ and applied to solve the standard instance sets J30, J60, and J120 generated by the problem generator ProGen devised by Kolisch and Sprecher [25], which were available in PSPLIB [28]. Both J30 and J60 contain 480 instances, and J120 contains 600 instances. Only in J30 the optimal makespans for all instances are known. In J60 and J120 the optimal makespans for some instances are not known and only upper bounds (best known solutions) and lower bounds are provided. The lower bounds are determined by the critical path method. The performance comparison between different methods is conducted by evaluating the same number of schedules, namely, 1000, 5000 and 50000 schedules in order to make it a machine-independent comparison.

The values of parameters were set by some preliminary experiments. These values are shown in Table I. The average deviation from the optimal makespans or the lower bounds and the CPU time used by the MTS are shown in Table II. Tables III to V list the performance comparison of the MTS with other state-of-the-art methods on J30, J60 and J120 respectively. From the last three Tables, the performances of the MTS are ranked the third on J30, the second on J60, and the fifth on J120 when compared with other state-of-the-art methods.

V. Conclusions and Future Research

The MTS was previously proposed for solving continuous optimization, both single-objective and multi-objective. The performance of the MTS was good for the real-parameter optimization. In this study, we arranged the MTS to solve the RCPSP, which is a combinatorial optimization problem. The result obtained is promising, though not excellent.

The combinatorial optimization is quite different from the continuous optimization. How to convert the concepts used in the continuous version of the MTS to fit the characteristics of the combinatorial optimization is the main issue in the future research. The other future research topics are trying to solve other combinatorial optimization problems by applying the MTS.

Acknowledgment

The authors thank National Science Council of the Republic of China, Taiwan (Contract No. NSC98-2221-E126-014-MY3) for partially supporting this research work.

References

[1] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," Annals of Operations Research, 2001. **102**: pp. 83-109.

[2] E Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," Management Science, 1992. **38**(12): pp. 1803-1818.

[3] P. Brucker, S. Knust, A. Schoo, and O. Thiele, "A branch and bound algorithm for the resource-constrained project scheduling problem," European Journal of Operational Research, 1998. **107**(2): pp. 272-288.

[4] R. Möhring, A. Schulz, F. Stork, and M. Uetz, "Solving project scheduling problems by minimum cut computations," Management Science, 2003. **49**(3): pp. 330-350.

[5] F.F. Boctor, "Some efficient multi-heuristic procedures for resource-constrained project scheduling," European Journal of Operational Research, 1990. **49**(1): p. 3-13.

[6] R. Kolisch, "Efficient priority rules for the resource-constrained project scheduling problem", J. Oper. Manag., vol. 14, pp. 179-192, 1996.

[7] V. Valls, M.A. Perez, and M.S. Quintanilla, "Heuristic performance in large resource-constrained projects", Technical Report, Department of Statistics and Operations Research, Universitat de Valencia, Valencia, Spain, 1992.

[8] G. Ulusoy and L. Ozdamar, "A constraint-based perspective in resource constrained project scheduling," International Journal of Production Research, 1994. **32**(3): pp. 693-705.

[9] L. Ozdamar and G. Ulusoy, "A note on an iterative forward backward scheduling technique with reference to a procedure by Li and Willis," European Journal of Operational Research, 1996. **89**(2): pp. 400-407.

 ISBN: 978-1-61208-217-2

[10] K.Y. Li and R.J. Willis, “An iterative scheduling technique for resource-constrained project scheduling,” European Journal of Operational Research, 1992. **56**(3): pp. 370-379.

[11] J.K. Lee and Y.D. Kim, “Search heuristics for resource constrained project scheduling,” Journal of the Operational Research Society, 1996. **47**(5): pp. 678-689.

[12] L. Ozdamar , “A genetic algorithm approach to a general category project scheduling problem,” Ieee Transactions on Systems Man and Cybernetics Part C - Applications and Reviews, 1999. **29**(1): pp. 44-59.

[13] M. Mori and C.C. Tseng, “A genetic algorithm for multi-mode resource constrained project scheduling problem,” European Journal of Operational Research, 1997. **100**(1): pp. 134-141.

[14] S. Hartmann, “A competitive genetic algorithm for resource-constrained project scheduling,” Naval Research Logistics, 1998. **45**(7): pp. 733-750.

[15] S. Hartmann, “A self-adapting genetic algorithm for project scheduling under resource constraints,” Naval Research Logistics, 2002. **49**(5): pp. 433-448.

[16] F.F. Boctor, “Heuristics for scheduling projects with resource restrictions and several resource-duration modes,” International Journal of Production Research, 1993. **31**(11): pp. 2547-2558.

[17] J.H. Cho and Y.D. Kim, “A simulated annealing algorithm for resource constrained project scheduling problems,” Journal of the Operational Research Society, 1997. **48**(7): pp. 736-744.

[18] K. Bouleimen. and H. Lecocq, “A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version,” European Journal of Operational Research, 2003. **149**(2): pp. 268-281.

[19] V. Valls , S. Quintanilla, and F. Ballestin, “Resource-constrained project scheduling: A critical activity reordering heuristic,” European Journal of Operational Research, 2003. **149**(2): pp. 282-301.

[20] C. Artigues , P. Michelon, and S. Reusser, “Insertion techniques for static and dynamic resource-constrained project scheduling,” European Journal of Operational Research, 2003. **149**(2): pp. 249-267.

[21] D. Merkle , M. Middendorf, and H. Schmeck, “Ant colony optimization for resource-constrained project scheduling,” IEEE Transactions on Evolutionary Computation, 2002. **6**(4): pp. 333-346.

[22] V. Valls, F. Ballestin, and S. Quintanilla, “Justification and RCPSP: A technique that pays,” European Journal of Operational Research, 2005. **165**(2): pp. 375-386.

[23] V. Valls , F. Ballestin, and S. Quintanilla, “A hybrid genetic algorithm for the resource-constrained project scheduling problem.,” European Journal of Operational Research, 2008. **185**(2): pp. 495-508.

[24] S. Hartmann and R. Kolisch, “Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem,” European Journal of Operational Research, 2000. **127**(2): pp. 394-407.

[25] R. Kolisch and A. Sprecher, “PSPLIB - A project scheduling problem library,” European Journal of Operational Research, 1997. **96**(1): pp. 205-216.

[26] L. Y. Tseng and C. Chen, “Multiple Trajectory Search for Large Scale Global Optimization,” Proceedings of the 2008 IEEE Congress on Evolutionary Computation, June 2-6, 2008, Hong Kong.

[27] Lin-Yu Tseng and Chun Chen, “Multiple Trajectory Search for Unconstrained/Constrained Multi-Objective Optimization,” Proceedings of the 2009 IEEE Congress on Evolutionary Computation, May 18-21, Trondheim, Norway.

[28] PROJECT SCHEDULING PROBLEM LIBRARY - PSPLIB, http://129.187.106.231/psplib/

[29] M. Ranjbar and F. Kianfar, “A Hybrid Scatter Search for the RCPSP,” Scientia Iranica Transaction E-Industrial Engineering, 2009. **16**(1): pp. 11-18.

[30] Y. Kochetov and A. Stolyar, “Evolutionary local search with variable neighborhood search for the resource-constrained project scheduling problem”, Proc. of the 3rd International Workshop of Computer Science and Information Technologies, Russia, 2003.

[31] V. Valls, F. Ballestin, and S. Quintanilla, “A population-based approach to the resource-constrained project scheduling problem”, Ann. Oper. Res., vol. 165, pp. 375-386, 2005.

[32] P. Tormos and A. Lova, “A competitive heuristic solution technique for Resource-Constrained Project Scheduling,” Annals of Operations Research, 2001. **102**: pp. 65-81.

[33] K. Nonobe and T. Ibaraki, “Formulation and tabu search algorithm for the resource constrained project scheduling problem”, in *Essays and surveys in metaheuristics*, Ribeiro, CC. and Hansen, P. (Ed.), Kluwer Academic Publishers, pp. 557-588, 2001.

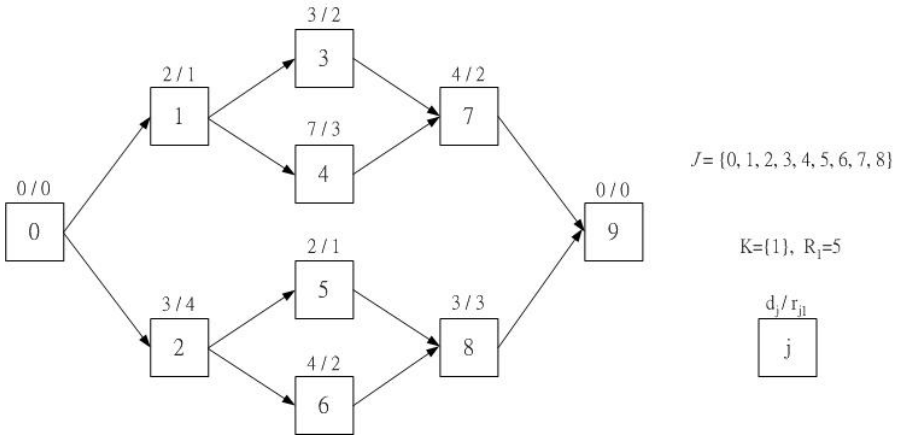

Figure 1. A project example for the RCPSP.

Table I The setting of parameter values

Parameter	Max_#_sched	*J*30	*J*60	*J*120
N (size of initial population)	All	20		
m (# of seeds)	All	3		
k (popul. size of region search)	All	12		
Max_No_Improve	All	3		
Number of GLS generations	1000	10		
	5000	15		
	50000	45		
num_of_mutate	All	15	30	60
p (# of activities to be re-inserted)	All	24	30	50

Table II The average deviation achieved and the CPU time spent by the MTS

	max-sch	*J*30	*J*60	*J*120
Avg. Dev.(%)	1000	0.12	11.72	35.81
	5000	0.04	11.05	33.67
	50000	0.01	10.67	32.11
Avg. CPU (seconds)	1000	0.005	0.019	0.089
	5000	0.013	0.080	0.396
	50000	0.054	0.783	3.781

Table III Performance comparison of MTS and other state-of-the-art methods on J30

Author(s)	Algorithm type	Average deviation(%)		
		1000	5000	50000
M. Ranjbar and F. Kianfar [29]	HSS, path reli.	0.10	0.03	0.00
Kochetov and Stolyar [30]	GA, TS, path reli.	0.10	0.04	0.00
This study	MTS, GLS, FBI	0.12	0.04	0.01*
Valls et al. [23]	Hybrid GA	0.27	0.06	0.02
Alcaraz and Maroto [1]	GA	0.33	0.12	–
Valls et al. [31]	DJGA	0.34	0.20	0.02
Tormos and Lova [32]	Sampling + BF/FB	0.25	0.13	0.05
Nonobe and Ibaraki [33]	Tabu Search	0.46	0.16	0.05
Tormos and Lova [32]	Sampling + BF	0.30	0.16	0.07
Hartmann [15]	Self-adapting GA	0.38	0.22	0.08

 ISBN: 978-1-61208-217-2

Table IV Performance comparison of MTS and other state-of-the-art methods on J60

Author(s)	Algorithm type	Average deviation(%)		
		1000	5000	50000
M. Ranjbar and F. Kianfar [29]	HSS, path reli.	11.59	11.07	10.64
This study	MTS, GLS, FBI	11.72	11.05	10.67*
Valls et al. [23]	Hybrid GA	11.56	11.10	10.73
Kochetov and Stolyar [30]	GA, TS, path reli.	11.71	11.17	10.74
Valls et al. [31]	DJGA	12.21	11.27	10.74
Hartmann [15]	Self-adapting GA	12.21	11.7	11.21
Hartmann [14]	Activity list GA	12.68	11.89	11.23
Tormos and Lova [32]	Sampling + BF/FB	11.88	11.62	11.36
Tormos and Lova [32]	Sampling + BF	12.14	11.82	11.47
Alcaraz and Maroto [1]	GA	12.57	11.86	–

Table V Performance comparison of MTS and other state-of-the-art methods on J120

Author(s)	Algorithm type	Average deviation(%)		
		1000	5000	50000
Valls et al. [23]	Hybrid GA	34.07	32.54	31.24
M. Ranjbar and F. Kianfar [29]	HSS, path reli.	35.08	33.24	31.49
Valls et al. [31]	DJGA	35.39	33.24	31.58
Kochetov and Stolyar [30]	GA, TS, path reli.	34.74	33.36	32.06
This study	MTS, GLS, FBI	35.81	33.67	32.11*
Valls et al. [31]	DJ, population based, changes operator	35.18	34.02	32.81
Hartmann [15]	Self-adapting GA	37.19	35.39	33.21
Tormos and Lova [32]	Sampling + BF/FB	35.01	34.41	33.71
Merkle et al. [21]	Ant Co. Opt.	–	35.43	–
Hartmann [14]	Activity list GA	39.37	36.74	34.03

 ISBN: 978-1-61208-217-2