A Scale-Free Based Memetic Algorithm for Resource-Constrained Project Scheduling Problems

Lixia Wang and Jing Liu

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an 710071, China
kjdxwlx@126.com, neouma@mail.xidian.edu.cn

Abstract. The resource-constrained project scheduling problem (RCPSP) is a popular problem that has attracted attentions of many researchers with various backgrounds. In this paper, a new memetic algorithm (MA) based on scale-free networks is proposed for solving RCPSPs, namely SFMA-RCPSPs. In SFMA, the chromosomes are located on a scale-free network. Thus, each chromosome can only communicate with the ones that have connections with it. In the experiments, benchmark problems, namely Patterson, J30 and J60, are used to validate the performance of SFMA. The results show that the SFMA performs well in finding out the best known solutions especially for Patterson and J30 data sets, besides, the average deviations from the best known solutions are small. Therefore, SFMA improves the search speed and effect.

Keywords: scale free networks, memetic algorithms, local search, resource-constrained project scheduling.

1 Introduction

Usually, a project is made up of several tasks and each task needs time and resources to complete. When the gross of resource is limited, and there occurs precedence relationships among these tasks, it comes the resource-constrained project scheduling problem (RCPSP). We can image this problem as a generalization of the well-known job shop scheduling problem, and it belongs to the class of combinatorial problems that is most difficult to solve, i.e., it is NP-hard [1].

In recent years, the research on scale-free networks becomes more and more popular [2]. In a scale-free network, most vertices have few connections with other nodes but there exist a few of vertices that have large amounts of connections [3], then these vertices with relatively large connections play an important part in communication of information in the network [2]. Various of networks in real world have the scale-free properties, such as World Wide Web [4], neural networks [5], metabolic interaction networks [6]. Therefore, more and more researchers propose to combine scale-free networks with existing algorithms to solve optimization problems [2].

Evolutionary algorithms (EAs), a kind of stochastic global optimization methods inspired by the biological mechanism of evolution and heredity, have been successfully used to solve various problems [7-12]. Memetic algorithms (MAs) are proposed by

Pablo Moscato [13], and are also called genetic local search, cultural algorithms or hybrid genetic algorithms [14]. In fact, MAs are the combination of population-based global search and the heuristic local search [15-17]. This kind of combination makes the search efficiency of MAs outperform EAs. Therefore, we propose using MAs to solve RCPSPs. The SFMA is tested on the Patterson, J30 and J60 sets, the results obtained show that it performs well in improving the search efficiency.

Many methods have been proposed for RCPSPs, such as heuristics methods, they can be divided into priority rule-based X-pass methods, non-standard metaheuristics and classical metaheuristics etc [18], Classical metaheuristics consist of genetic algorithm, simulated annealing algorithm, tabu search algorithm, particle swarm optimization algorithm, ant colony optimization etc [19]. Alcaraz and Maroto propose a GA based on the serial SGS and list representation [20]. Valls et al. develop a SA method focuses on forward-backward improvement [21]. Fleszar and Hindi use a variable neighborhood search to solve the RCPSP [1].

The rest of this paper is organized as follows. Section 2 introduces the detail content about the SFMA. Section 3 gives the experiments on the benchmark data sets. Section 4 concludes the work in this paper.

2 Scale-Free Based Memetic Algorithm for RCPSP

2.1 RCPSPs

In a resource-constrained project scheduling problem, a project is made up of n tasks which are marked by 1, 2, …, n. A duration d_j is needed to finish task j [22]. The objective of RCPSPs is to arrange the start time for these tasks. In the same time, two kinds of constraints exist to bound the work; they are called precedence constraints which can be depicted by a graph shown in Figure 1 and resource capacities. The precedence constraints mean that a task can not be arranged before all of its predecessors have been finished [23]. Every process to complete a task needs certain of time and resources. The aim is to find out start and finish time for all of the tasks, at the same time, both the constraints given above must be satisfied and the duration of the project or the resources cost or some other suitable objective is optimized [5]. Makespan is the most popular objective function for RCPSP, because people would like to use the least time to complete a project. In this paper, makespan is used as the objective function.

As shown in figure 1, each vertex represents a task, task 1 and n are dummy tasks with their durations d_1 and d_n both be zeros. Task 1 and n are called the initial task and final task which must be arranged before and after all of the other tasks respectively. If there is an arrow between task i and j of length d_i, it presents that task i is j's predecessor and j can not start before i has finished, d_i is the duration of i. All immediate successors of the task i make up of a set represented by A_i, in the same way, the set of task i's immediate predecessors is denoted by B_i [24]. We use γ_i to presents the amount of predecessors of task i, it equals to the number of arrows terminate at vertex i in figure 1. The graph thus defined must be acyclic; otherwise, someone will be its own predecessor [7].

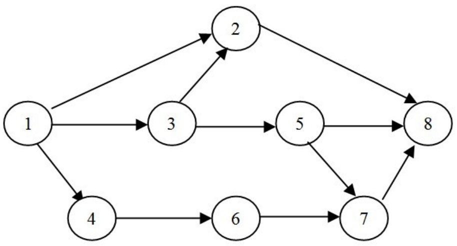

Fig. 1. Precedence graph

2.2 Population Initializations with Scale-Free Structure

For a node k in a network, its degree distribution usually expressed by $P(k)$. In traditional random networks just like ER networks, the degree distribution obeys the Poisson distribution [3]. In fact, the distribution of real networks is far from the Poisson distribution [25], it is similar to a simple form:

$$P(k) \sim k^{-r} \tag{1}$$

Here γ is a constant exponent which varies between 1.8 and 2.5 [3]. Its speed of falling off is much slower than ER networks, so it allows a few nodes to have large connections. Researchers call this distribution a power-law distribution [3].

Before initializing the population, firstly construct the scale free network to deposit the chromosomes. The network is constructed according to BA-modal, which is proposed by Barabási and Albert [25].

Algorithm 1. Construction of the BA modal

Step 1: Produce a network with m_0 nodes attach to each other;

Step 2: When a new node is generated, calculate the degree distribution of these existing nodes, then select which m nodes (node) to attach according to the distribution, here m and m_0 are both integers and $m<m_0$;

Step 3: Execute step 2 repeatedly, until the total number of nodes is enough.

After finishing the construction of scale free network, the chromosomes are generated corresponding to the vertices on the network. Three methods are used to produce the initial population alternately until every vertex on the network has a chromosome and the corresponding. The first two methods are recomposed from the famous algorithms [7], and the third one is proposed by us, here we just give the detail procedure of the third one.

Suppose all the tasks are presented in a nonperiodic graph G, each task corresponds to one node, if there is an arrow from node i to node j ($i, j \in G$), thus task i is called the immediate predecessor of task j, or task j is the immediate successor of task i. γ_i is used to present the number of immediate successors of task i.

Algorithm 2. Method 3 for initialization

Step 1: For each task v of G, calculate its number of immediate predecessors γ_w, s is an empty set use to deposit tasks;

Step 2: Find out the tasks those γ_s are zeros then select only one task w randomly and arrange it into s.

Step 3: For the task arranged just now, find out its immediate successors for example task v, then $\gamma_v = \gamma_v - 1$;

Step 4: Find out the tasks that $\gamma=0$ and belong to the immediate successors of the task arranged just now, then arrange one of them randomly, repeat step 3 and 4 until no task can be arranged;

Step 5: Repeat step 2, 3 and 4 until all of the tasks in G have been arranged into s, then a chromosome has been produced.

2.3 Competition-Crossover Operator

As all of the chromosomes are fixed on a scale-free network, it represents that a chromosome can change information only with the one who has connection relationship with it. For each chromosome s in the population, we firstly find out the units connect with it and compare those fitness values, the unit has the smallest makespan is marked as m, let m execute the mutation operator then obtain a new chromosome labeled p_2, the mutation operator will be introduced in the content below, let s be p_1. At present if a random number produced is larger than the crossover probability Pc, we select the makespan smaller one between p_1 and p_2 as a new unit and mark by s_{new}, then s will be replaced by s_{new}; otherwise, if the random number is smaller than P_c, the crossover operator will be executed, here we use the two-point operator adapted from [7], the detail procedure is introduced below.

Algorithm 3. Crossover Operator

Let p_1 and p_2 be the two parents, c_1 and c_2 be the children. n is the total number of the tasks as well as the length of a chromosome.

Step 1: Generate two numbers r_1 and r_2 randomly, $r_1 < r_2$, r_1, r_2 are both integers and belong to interval $(1, n)$.

Step 2: $c_1(1: r_1) = p_1(1: r_1)$ and $c_1(r_2+1: n) = p_1(r_2+1: n)$.

Step 3: for j=1: n, if p_2 (j) is not belong to c_1, then $r_1 = r_1+1$ and $c_1(r_1) = p_2(j)$.

Similarly, c_2 is obtained in the same way by interchanging p_1 and p_2. Now there are four units include p_1, p_2, c_1, c_2, then choose the one with the minimum makespan to replace s. In addition, if the parents are valid, the children produced are valid too, because the children inherit the prior orders from their parents.

2.4 Mutation Operator and Local Search Operator

The mutation operator is adapted from [7]. This kind of mutation is helpful to the diversity of the population. For any task in a chromosome, our method to mutate is to find the interval that allows the task to move and will not violate the precedence constraints. The detail of the operator is described as follow; image s is a valid chromosome.

Algorithm 4. Mutation operator

Step 1: Choose a task i among all of the tasks in random except 1 and n, here n indicates the length of the chromosome, then find out its position p_i in the chromosome s;

Step 2: For the tasks belong to i's immediate predecessors, calculate their largest position in the chromosome s marked l; similarly, find out the smallest position of the tasks of i's immediate successors and label as e;

Step 3: Randomly generate an integer k between l and e, and k can not equal to p_i;
Step 4: Then task i can move to position k, and a new chromosome is generated.

In order to keep the better units in current generations, we only receive the mutation ones whose makespans are smaller than before. Finally, compare the fitness value of the new chromosome with the old one's, and reserve the individual with the smaller makespan.

Calculate the fitness values of the current population and find out the minimum makespan, as the number of chromosomes whose fitness values are equal to the minimum makespan is not only one, the SFMA find out the whole chromosomes and execute local search upon them to make their fitness values less and less. Here we let the best individuals in the current generations execute mutation operator 20 times. During the experiments we can see that this operator is much helpful to obtain the optimal makespan rapidly.

2.5 Implementation of SFMA

The overall scheme of the SFMA is demonstrated in Algorithm 7 below. As states above, MAs are GAs combine with local search, competition-crossover and mutation are belonging to the GAs, step 10-12 are local search. The algorithm is stopped when either the minimum makespan is equal to the best-known makespan or the total schedules produced exceed the number we set.

Algorithm 5. The implementation of SFMA

```
1. Algorithm begin;
2. Construct the scale free network by BA modal;
3. Initialize original population;
4. Calculate the fitness values of the population;
5. Store the minimum makespan of the current population as fmin;
6. while the stopping conditions are not satisfied;
7. Execute competition-crossover operators;
8. Execute mutation operator;
9. Find out the minimum makespan in the current population $f_c$;
10. for the chromosomes whose fitness values are equal to $f_c$;
11. Execute local search;
12. end
13. Update the minimum makespan $f_{min}$;
14. end
15. Algorithm end
```

3 Experiments

In this section, the benchmark problems from PSPLIB [26], namely Patterson, J30, and J60, are used to validate the performance of SFMA. In the following experiments, for each instance, 10 independent runs are executed and the average values of the following criteria are used to show the performance of the SFMA:

(1) %NBS: It means the percentage ratio of the best known solutions found by the SFMA to the total number of instances [7].

(2) %ERR: Means the average error; we calculate this error according the following expression:

$$\%\mathrm{ERR} = \frac{solution\_makespan - best\_makespan}{best\_makespan} \bullet 100\% \tag{6}$$

In the scale free network we generate, *m* indicates the number of vertices (vertex) connected by the new node, we randomly choose one hundred instances from the J30 sets and test the effect of m on %NBS, there we set *m* to be 1, 2, 3, 4, 5, 8 respectively, and other parameters are the same, the results are shown in Table 1, from the results we can see, the smaller m is set, the better it performs. Besides, through the tests on the same data, we find that the two point crossover operator performs better than the one point crossover operator.

Table 1. Comparison of different *m*

m	**1**	**2**	**3**	**4**	**5**	**8**
%NBS	76.0	74.7	73.7	74.3	73.1	74.2

In the local search process, as the number of individuals whose fitness values are equal to the minimum one is unconcern, we choose several individuals to execute local search operator rather than only one individual. The results shown in Table 2 proves this. Through the results, we can see that no matter the %NBS or the %ERR, the multi units method performs much better than the single unit method. Table 3 shows the average deviations, and the stopping criterias are maximum of 1000, 5000 schedules respectively. The results validate that the deviations are not small enough, so in the future, it is a goal for us to reduce the deviation.

Table 2. Comparison of local search with single unit and multi units

Data sets	**Local search**	**%NBS**	**%ERR**	**AvgEvals**
Patterson	Single unit	93.36	0.08	954
Patterson	Multi units	98.73	0.02	735
J30	Single unit	82.68	0.47	1607
J30	Multi units	88.27	0.28	1867
J60	Single unit	67.42	1.78	4130
J60	Multi units	70.04	1.42	4707

Table 3. Deviation from critical-path lower bound

Data sets	**Max schedules**	
	1000	5000
J30	0.94	0.42
J60	14.58	13.06

4 Conclusions

In this paper, we proposed a SFMA for the single project, single mode, resource-constrained project scheduling problem. The SFMA combines the global search and the local search, so that it can search the solution space much more thoroughly. Besides, it is characterized by introducing a scale-free network, here we construct a scale-free network to let the chromosomes to fix on, when executing the competition and crossover operators, the chromosomes can not operate with random individuals but must with the ones attach to them. Thus the good individuals can influence others more. The SFMA is tested on the Patterson, J30 and J60 sets, the average rates of best known solutions found by SFMA are rather high especially on Patterson data sets and J30 sets, it proves that the SFMA improves the search efficiency.

Acknowledgement. This work is partially supported by the National Natural Science Foundation of China under Grants 61271301 and 61103119, and the Fundamental Research Funds for the Central Universities under Grant K5051202052.

References

1. Krzysztof, F., Khalil, S.H.: Solving the resource-constrained project scheduling problem by a variable neighbourhood search. European Journal of Operational Research 155(1), 402–413 (2004)
2. Cheng, Z., Yi, Z.: Scale-free fully informed particle swarm optimization algorithm. Information Sciences 181(20), 4550–4568 (2011)
3. Oliver, H., Michael, S., Wolfgang, K.: Scale-Free Networks the Impact of Fat Tailed Degree Distribution on Diffusion and Communication Processes. Wirtschaftsinformatic, 267–275 (2006)
4. Barabási, A.L., Albert, R., Jeong, H.: Scale-free characteristics of random networks: the topology of the world wide web. Physica A 281(1-4), 69–77 (2000)
5. Egúluz, V.M., Chialvo, D.R., Cencchi, G.A., Baliki, M., Apkarian, A.V.: Scale-free brain functional networks. Phys. Rev. Lett. 94(1) (2005)
6. Jeong, H., Mason, S., Barabási, A.L., Oltvai, Z.N.: Lethality and centrality in protein networks. Nature 411, 41–42 (2001)
7. Khalil, S.H., Hongbo, Y., Krazysztof, F.: An Evolutionary Algorithm for Resource-Constrained Project Scheduling. IEEE Transactions on Evolutionary Computation 6(5), 512–518 (2002)
8. Zhong, W., Liu, J., Xue, M., Jiao, L.: A multiagent genetic algorithm for global numerical optimization. IEEE Trans. on Syst., Man, and Cybern., Part B 34(2), 1128–1141 (2004)
9. Liu, J., Zhong, W., Jiao, L.: A multiagent evolutionary algorithm for constraint satisfaction problems. IEEE Trans. on Syst., Man, and Cybern., Part B 36(1), 54–73 (2006)
10. Liu, J., Zhong, W., Jiao, L.: Moving block sequence and organizational evolutionary algorithm for general floorplanning with arbitrarily shaped rectilinear blocks. IEEE Trans. on Evolutionary Computation 12(5), 630–646 (2008)
11. Jiao, L., Liu, J., Zhong, W.: An organizational coevolutionary algorithm for classification. IEEE Trans. on Evolutionary Computation 10(1), 67–80 (2006)

12. Liu, J., Zhong, W., Jiao, L.: A multiagent evolutionary algorithm for combinatorial optimization problems. IEEE Trans. on Systems, Man, and Cybernetics Part B 40(1), 229–240 (2010)
13. Pablo, M.: On evolution, search, optimization, genetic algorithms and martial arts towards memetic algorithms Caltech concurrent computation program. C3P Report (1989)
14. Joshua, D. K., David, W.C.: M-PAES: A memetic algorithm for multiobjective optimization. Evolutionary Computation 1(1), 325–332 (2000)
15. Ong, Y.S., Keane, A.J.: Met-lamarckian learning in memetic algorithms. IEEE Transaction on Evolutionary Computation 8(2), 99–110 (2004)
16. Ong, Y.S., Lim, M.H., Zhu, N.: Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36(1), 141–152 (2006)
17. Ong, Y.S., Lim, M., Chen, X.: Memetic computation-Past, present & future (Research Frontier). IEEE Computational Intelligence Magazine 5(2), 24–31 (2010)
18. Rainer, K., Sonke, H.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174(1), 23–37 (2006)
19. Tormos, P., Lova, A.: An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research 41, 1071–1086 (2003)
20. Alcaraz, J., Maroto, C.: A robust genetic algorithm for resource allocation in project scheduling. Annals of Operations Research 102, 83–109 (2001)
21. Valls, V., Ballestin, F., Quintanilla, M.S.: Justification and RCPSP: A technique that pays. European Journal of Operational Research 165, 375–386 (2005)
22. Blazewicz, J., Lenstra, J.K., Rinnooy, A.H.G.: Scheduling subject to resource constraints: Classification and complexity. Discrete Appl. Maths. 5, 11–24 (1983)
23. Valls, V., Ballestin, F., Quintanilla, M.S.: A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 185(2), 495–508 (2008)
24. Tormos, P., Lova, A.: A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling. Annals of Operations Research 102, 65–81 (2001)
25. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
26. Kolisch, R., Sprecher, A., Drexl, A.: Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41(10), 1693–1703 (1995)