Individual Scheduling for the Multi-Mode Resource-Constrained Multi-Project Scheduling Problem

Vladislav Korotkov[1], Mikhail Matveev[2]

[1] Voronezh State University, Voronezh, Russia
chasecrunk@gmail.com
[2] Voronezh State University, Voronezh, Russia

1 Introduction

Solving a well-known resource constrained project scheduling problem (RCPSP) is crucial for project management. It consists of assigning a start time to each activity such that the precedence relations and the resource constraints are satisfied [1, 2]. The resulting schedule usually quantitatively distributes resources along the project timeline. However, for practical purposes, individual schedules have to be obtained for employees and other named resources to organize the workflow. Such resource allocation was previously presented assuming the difference in skills and other individual features [3]. This approach dramatically extends the search space. In practice, accounting resource heterogeneity is usually redundant or difficult to implement.

This paper proposes the two-stage approach to individual schedule construction assuming that all units of the same resource are interchangeable. The first stage is a general scheduling based on genetic algorithm. The modified multi-project and multi-mode variation of the problem (also known as MRCMPSP) is considered as it is much closer to the real-world problems [4]. The second stage includes individual scheduling by allocating resource units to activities subject to the required condition (in this case, the uniformity of resource allocation). A simulated annealing algorithm is proposed for this purpose.

2 Problem Description

We consider a set P of n projects. Each project $p \in P$ consists of non-preemptive activities J_p. We need to determine a start time s_{pj} for each activity $j \in J_p$. Each project p also has a start date $Start_p$ so that none of the activities can be started earlier. Some activities require others to complete, so $Pre(j)$ is a set of predecessors of $j \in J_p$.

The set $L_p = L_p^\rho \cup L_p^\nu$ of resources is allocated to each project p. Here L_p^ρ are local renewable resources and L_p^ν are local non-renewable resources. The amount of non-renewable resources is fixed over the entire project duration. Examples of this type of resource are money, consumables, etc. Renewable resources have a fixed capacity per

ICID-2019 Conference

time unit. Workers and machines are examples of this type of resource. Each resource $l \in L_p$ of the project p has a capacity of c_{pl}. There are also global renewable resources G^{ρ} limited by capacities $c_g, g \in G^{\rho}$. They can be used in any project.

Each activity $j \in J_p, p \in P$ is performed in exactly one mode m_{pj} from the set of possible execution modes M_{pj}. The mode determines activity duration and specific resource requirements. Let $d_{pjm_{pj}}$ be the processing time of activity $j \in J_p$ in mode $m_{pj} \in M_{pj}$, and $r_{pjm_{pj}l}$ define its consumption of l.

The objective is to find such activity start times s_{pj} and mode assignments m_{pj} that satisfy precedence and resource constraints while minimizing the total project delay TPD. The total project delay is calculated as follows:

$$TPD = \sum_{p \in P} (MK_p - CPD_p) \tag{1}$$

$$MS_p = Finish_p - Start_p \tag{2}$$

where $Finish_p$ is the last activity completion time for the project p; CPD_p is a critical path duration assuming resource constraint relaxation and the assignment of execution modes with the shortest durations.

Thus, the objective can be formulated as the combinatorial optimization problem:

$$TPD \to \min \tag{3}$$

subject to:

$$\sum_{j \in A_p(t)} r_{pjm_{pj}l} \le c_{pl} \qquad \forall p \in P, l \in L_p^{\rho}, t \in [0,T] \tag{4}$$

$$\sum_{j \in J_p} r_{pjm_{pj}l} \le c_{pl} \qquad \forall p \in P, l \in L_p^{v} \tag{5}$$

$$\sum_{p \in P} \sum_{j \in A_p(t)} r_{pjm_{pj}g} \le c_g \qquad \forall g \in G^{\rho}, t \in [0,T] \tag{6}$$

$$s_{pj} + d_{pjm_{pj}} \le s_{pj'} \qquad \forall p \in P, j' \in J_p, j \in Pre(j') \tag{7}$$

$$s_{pj} \ge Start_p \qquad \forall p \in P, j \in J_p \tag{8}$$

$$m_{pj} \in M_{pj} \qquad \forall p \in P, j \in J_p \tag{9}$$

$$A_p(t) = \left\{ j \in J_p \mid s_{pj} \le t < s_{pj} + d_{pjm_{pj}} \right\} \tag{10}$$

Here T is an upper bound of projects completion time; $A_p(t)$ is the set of activities of the project p in progress in the period $[t,t+1]$; (7) defines the precedence constraints while resource constraints are defined in (4), (5) and (6).

The second stage is aimed to find correspondences between activities and resource units that are in some sense optimal. The schedule obtained in the first stage determines durations and resource requirements for each activity. Let H_l be the set of all available units of resource l and $J_l^\rho \subset \bigcup_{p \in P}$ be the set of all activities that require resource l. Each activity $j \in J_l^\rho$ has duration d_j and requires r_{jl} of resource l. For each resource unit h we define S_h as the desired set of activities from its individual schedule. In this case, the uniformity of the resource allocation is considered as the optimality criterion. We assume that no resource unit can be utilized in multiple parallel activities. Thus, we can formulate the following combinatorial optimization problems for each renewable resource type $l \in G^\rho \cup \bigcup_{p \in P}$:

$$\sum_{h \in H_l} (U(h) - AvgU(l))^2 \to \min \tag{11}$$

$$U(h) = \sum_{j \in S_h} d_j \tag{12}$$

$$AvgU(l) = \frac{\sum_{h \in H_l} U(h)}{|H_l|} \tag{13}$$

subject to:

$$\left|\{S_h \mid j \in S_h\}\right| = r_{jl} \qquad \forall j \in J_l^\rho \tag{14}$$

$$j \notin Over(j') \qquad \forall h \in H_l; j, j' \in S_h; j \neq j' \tag{15}$$

$$S_h \subset J_l^\rho \qquad \forall h \in H_l \tag{16}$$

Here $Over(j)$ is the set of activities which overlap with j. $U(h)$ (12) is the total utilization of h, $AvgU(l)$ (13) is the average resource utilization for l. Constraint (14) verifies that resource needs are met. Constraint (15) prevents allocation to overlapping activities.

3 General Scheduling

RCPSP (and hence MRCMPSP) is computationally intensive combinatorial optimization task. It was proven to be NP-hard [5]. Optimal solution may be obtained only for

rather small instances while practical problems can reach hundreds of activities. Therefore, the majority of studies consider applying different heuristics to find suboptimal solutions without traversing the entire search space [2].

In this study a genetic algorithm approach was used for general schedule construction. Some features of the previous implementations were incorporated [6, 7, 8]. Implementation details are provided below.

3.1 Solution Representation

Each solution is encoded by a chromosome which corresponds to strictly one feasible schedule. Each chromosome consists of two parts: the first one contains an activity list in order of priority and the second one includes chosen activity modes. The corresponding schedule can be obtained by the following sequential procedure. Starting with an empty schedule we construct a new partial schedule by placing the next activity from an activity list to the earliest possible time so that precedence and resource constraints are satisfied. This is usually called a serial schedule generation scheme (SGS). Note that any such solution corresponds to only one schedule but the same schedules can be obtained from different solutions.

3.2 Fitness Function

The modification of fitness function proposed in [8] was used for solution evaluation. It penalizes solutions by adding the number of requested non-renewable resource units that exceed the capacity. Such approach is considered more efficient in terms of faster convergence to suboptimal solutions.

$$f(I)=\begin{cases} 1-\dfrac{MaxTPD-TPD(I)}{MaxTPD}=1-\dfrac{MaxTMS-TMS(I)}{MaxTPD}, & ERR=0 \\ 1+\dfrac{TPD(I)}{MaxTPD}+ERR, & otherwise \end{cases} \tag{17}$$

$$TMS=\sum_{p\in P} MS_p \tag{18}$$

$$MaxTMS=\sum_{p\in P}\sum_{j\in J_p} d_{pjm_{pj}} \tag{19}$$

$$MaxTPD=\sum_{p\in P}\left(\sum_{j\in J_p} d_{pjm_{pj}}-CPD_p\right) \tag{20}$$

$$ERR=\sum_{p\in P}\sum_{l\in L_p^r}\max\left\{0,\frac{\sum_{j\in J_p} r_{pjm_{pj}l}-c_{pl}}{c_{pl}}\right\} \tag{21}$$

Here ERR (21) is the non-renewable resource error, TMS (18) is the total makespan of the projects, equations (19) and (20) define estimations of maximum total makespan and maximum total project delay respectively.

3.3 Initial Population

The initial population is randomly generated to ensure diversity of solutions. The procedure of random generation starts with an empty activity list. At each step of the procedure a random activity whose predecessors are already in the list is appended. Its mode is also taken randomly and added to the second part of the chromosome.

The resulting instances obviously satisfy the precedence constraints. And for the algorithm to work correctly crossover and mutation operations must be implemented in such a way that the resulting solutions are also feasible.

3.4 Crossover Operation

Crossover operation combines a pair of parent solutions to produce two children that inherit their parents' features. The operation is applied independently to activity lists and mode lists of parent chromosomes.

A point q on both active lists is picked randomly splitting them into two parts. All activities from the first part of the first (second) activity list are transferred to the first (second) offspring list and the remaining activities are appended in the order in which they are located in the second (first) parent. Such method is called a single-point crossover.

Activity mode lists of the parents are crossed over according to uniform method. For each *i*-th activity a random value $p_i \in [0,1]$ is picked independently. If $p_i \geq 0.5$ then the first (second) child inherits the mode of this activity from the first (second) parent. Otherwise, the first (second) child inherits the mode from the second (first) parent.

3.5 Mutation Operation

Mutation is defined as a random change in the solution obtained by crossing over. Mutation operation is also applied independently to each part of a chromosome.

For each *i*-th activity of activity list a random value $p_i \in [0,1]$ is picked independently. If $p_i \leq P_{PM}$, where P_{PM} is initially defined mutation probability, then *i*-th and *i+1* activities are swapped unless this violates the precedence constraints.

Each activity mode is replaced by a randomly picked one with initially defined probability P_{MM}.

4 Individual Scheduling

Since we assume interchangeability of individual units of the same resource, it's reasonable to represent solution as a multiset to reduce the search space. Thus, any solution is defined as:

$$Z_l = \left\{ S_l^{(1)} \times N_l^{(1)}, ..., S_l^{(k)} \times N_l^{(k)} \right\} \qquad k \le \left| H_l \right| \tag{22}$$

where $S_l^{(i)} \subset J_l^{\rho}$ is a list of activities, and $N_l^{(i)}$ is the number of such schedules.

The neighbor of any solution relative to the particular activity j can be obtained by removing this activity from some schedule and adding to another one. Note that target schedule should not contain any activities overlapping with j.

The initial solution can be randomly generated. The procedure starts with a set of empty schedules. Then each activity j from J_l^{ρ} is added r_{jl} times to random schedules. Again, the target schedules should contain only activities which don't overlap with j.

Now it becomes possible to apply any local search algorithm to solve the problem. In this study the simulated annealing method was used. The main steps of the algorithm are:

Obtain random initial solution and set x (the current solution) and x_{best} (the best solution) to it. Store corresponding objective function value in f_{best};

1. Initialize the temperature T with T_0;
2. Get random neighbor x' of the current solution;
3. Calculate the objective function $f(x')$ and the difference $\Delta := f(x') - f(x)$;
4. if $\Delta < 0$ then $x := x'$ and
5. if $f(x') < f_{best}$ then $f_{best} := f(x')$ and $x_{best} := x'$;
6. Otherwise, if $e^{-\Delta/T} > p$ ($p \in [0,1]$ is a random value) then $x := x'$;
7. Lower the temperature: $T := \alpha T$ ($\alpha \in (0,1)$);
8. If termination conditions aren't met, then go to step 3.

5 Computational Experiments

The algorithm was tested on a subset of MISTA 2013 Challenge problem instances [4]. The algorithm was implemented in Python and all tests were performed on a computer with Intel Core i5 8250U.

Table 1 shows results for 9 problems of MISTA 2013 Challenge instance set. The problems with different number of projects, activities and renewable resources were considered. Parameters of genetic algorithm were: population size – 150, crossover probability – 0.8, mutation probability – 0.04. Simulated annealing parameters were: iteration count – 1000, initial temperature – 1000, temperature reduction factor – 0.9.

Table 1. The results of applying the algorithm to some problems of MISTA 2013 Challenge. f_{GS} – total project delay of the general schedule, f_{IS} – average square resource utilization error of individual schedules, t_{GS} – total general scheduling time, t_{IS} – total individual scheduling time.

Id	Project count	Activity count	Renewable resource count	f_{GS}	f_{IS}	t_{GS} (s)	t_{IS} (s)
A-1	2	20	2	9	59.88	7.69	0.08
A-2	2	40	2	30	11.01	21.84	0.16
A-3	2	60	2	10	77.04	58.42	0.26
A-4	5	50	2	130	35.95	39.19	0.15
A-5	5	100	2	628	77.39	130.33	0.31
A-6	5	150	2	785	40.64	367.99	0.35
A-7	10	100	2	2441	5.18	92.50	0.14
A-8	10	200	2	1677	16.52	607.41	0.32
A-9	10	300	2	1683	61.88	2281.05	0.79

A series of tests was performed on the problem A-1 with different number of generations to estimate the rate of convergence of the proposed genetic algorithm. The values of other parameters were the same. 20 runs were made for each population size. The results are shown in table 2.

Table 2. The results of applying the proposed genetic algorithm with different number of generations.

Number of generations	Average *TPD*	Average processing time (s)
10	11.20	2.80
30	5.85	7.67
50	4.55	12.40
70	4.49	16.85

A series of tests was performed on the problem A-9 with different number of iterations to estimate the rate of convergence of the proposed simulated annealing method. The values of other parameters were the same. 20 runs were made for each number of iterations. The results are shown in table 3.

Table 3. The results of applying the proposed simulated annealing algorithm with different number of iterations.

Number of iterations	Average error	Average processing time (s)
100	4861.26	0.17
500	239.78	0.45
1000	49.94	0.80

5000	29.82	3.59
10000	28.23	7.06

6 Conclusion

In this study, an extension to the traditional MRCMPSP was introduced to make individual schedules for renewable resources. The resulting two-stage optimization model was implemented using genetic algorithm and simulated annealing method. The algorithm was tested on the set of problems from MISTA 2013 Challenge.

References

1. Kolisch, R.: Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes. Physica-Verlag, Heidelberg (1995)
2. Abdolshah, M.: A Review of Resource-Constrained Project Scheduling Problems (RCPSP) Approaches and Solutions. International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies 5(4), 253–286 (2014)
3. Yannibelli, V., Amandi, A.: A knowledge-based evolutionary assistant to software development project scheduling. Expert Systems with Applications 38(7), 8403-8413 (2011)
4. Wauters, T., Kinable, J., Smet, P., et al.: The Multi-Mode Resource-Constrained Multi-Project Scheduling Problem. Journal of Scheduling 19(3), 271-283 (2016)
5. Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.G.: Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5(1), 11-24 (1983)
6. Hartmann, S.: Project Scheduling with Multiple Modes: A Genetic Algorithm. Annals of Operations Research 102(1-4), 111-135 (2001)
7. Kumanan, S., Jose, G.J., Raja, K.: Multi-project scheduling using a heuristic and a genetic algorithm. The International Journal of Advanced Manufacturing Technology 31(3-4), 360–366 (2006)
8. Sebt, M.H., Afshar, M.R., Alipouri, Y.: An Efficient Genetic Algorithm for Solving the Multi-Mode Resource-Constrained Project Scheduling Problem Based on Random Key Representation. International Journal of Supply and Operations Management 2(3), 905-924 (2015)