Evolutionary Local Search with Variable Neighborhood for the Resource Constrained Project Scheduling Problem

Yu.A. Kochetov
Sobolev Institute of Mathematics
Novosibirsk, Russia
e-mail: jkochet@math.nsc.ru

A.A. Stolyar
Sobolev Institute of Mathematics
Novosibirsk, Russia
e-mail: asto@math.nsc.ru

Abstract

In this paper we consider the resource constrained project scheduling problem. We propose an evolutionary algorithm based on path relinking strategy and tabu search with variable neighborhood. Computational experiments are made for the PSPLib data set. For most benchmarks the algorithm found best known solutions. For several test instances the best solutions was improved.

1. Introduction

In the resource constrained project scheduling problem we consider the single project which consists of a set of activities. Each activity has duration and must be executed without preemptions. During the processing period of an activity, constant amounts of renewable resource are needed. The available capacity of each resource type is limited. A partial order representing precedence relations between activities is given. The objective is to determine the starting time for each activity such that to minimize the total project length.

The resource constrained project scheduling problem, as generalization of well-known job-shop problem, is *NP* - hard in the strong sense [1]. There are exact and heuristic algorithms for solving the problem. In this paper we propose a hybrid approach combining concepts of tabu search [3], variable neighborhood search [4] and genetic algorithms [3]. The main idea of the approach is to accumulate useful information which is contained in the best found local optima.

2. Problem statement

By $J=\{1,\ldots,n\}\cup\{0, n+1\}$ denote the set of activities. The dummy activities *0* and $n+1$ has zero duration and zero resource requirements. They represent the beginning and the end of the project. For each activity $j \in J$ we denote its processing time by $p_j \geq 0$, $p_0 = p_{n+1} = 0$. The precedence relations on J are represented by the set of pairs $C=\{(i,j)\}$, where activity i is a predecessor of the activity j. By P_j denote the set of all predecessors of the activity j. The set C contains all pairs $(0,j)$ and $(j,n+1)$, $1 \leq j \leq n$.

By $K=\{1,\ldots,m\}$ denote the set of renewable resources. We assume that amount of each resource R_k, $k \in K$ is a constant. By r_{jk} denote the activity j per-period usage of the k^{th} type resource.

Now we introduce variables. The starting time of the activity j is denoted by s_j. The set $A(t) = \{j \in J \mid s_j \leq t < s_j+p_j\}$ is a set of activities which are in progress at the time t. The formal problem statement can be written as follows:

$$\min s_{n+1}$$

$$s.t. s_i + p_i \leq s_j, (i,j) \in C,$$

$$\sum_{j \in A(t)} r_{jk} \leq R_k, k \in K, t \geq 0,$$

$$s_j \geq 0, j \in J.$$

The objective function minimizes theproject makespan. The first constraint corresponds to the precedence relations. The second one corresponds to the resource constraints.

3. Algorithm outline

The algorithm is evolutionary. At the each stage of evolution, we have a population of local optima. The crossover operator is based on path relinking strategy. We choose a pair of solutions from the population and construct a path between them in the neighborhood graph. For this purpose we use the greedy randomized adaptive search procedure (GRASP) [2]. The path constructed is a collection of feasible solutions. We select one of them and improve it by the tabu search algorithm with variable neighborhood. The improved solution is added to the

Workshop on Computer Science and Information Technologies CSIT'2003, Ufa, Russia, 2003

population. The worst one is removed from it. The diversification strategy is applied at the end of evolution.

3.1 Solution Representation

We use an activity list representation [5]. Feasible solution is coded by the list of activities $L=(j_1,\ldots,j_n)$. All lists considered are assumed to be compatible with the precedence relations. In other words, activity i is listed before activity j if the precedence relations require to finish i before start j. For arbitrary list L, the serial decoding procedure [5] calculates an active schedule $S(L)$. It is known that there is an optimal schedule among active schedules.

3.2 Neighborhoods

Let L be a list of activities and S be a correspondent active schedule.

Definition 1. *Block* of activity j is a set of activities which are scheduled in parallel with activity j:

$$Block_j = \{i \in J \mid s_i + p_i \in [s_j, s_j + p_j]\} \cup$$
$$\cup \{i \in J \mid s_i \in [s_j, s_j + p_j]\}.$$

If the block contains at least one predecessor of the activity j then we put $Block_j := \varnothing$.

For a given schedule S we define the directed graph $G_S(V,E)$, where $V=J$ and $E=\{(i, j) \mid s_i+p_i=s_j, i \in P_j\}$.

Definition 2. *Outcoming Network* of the activity j is a maximal due to inclusions subgraph of the graph G_S which is a connected network with a source corresponding to the activity j.

The element L' of the neighborhood $N_1(L)$ is constructed for each activity j with non-empty block. The list L' is obtained from the list L in four stages.

On the first stage we define two positions *First* and *Last* in the list L. The *First* is a minimal position in the list L among all activities in the block of the activity j. The *Last* is a maximum of two values $Last_1$ and $Last_2$ which are maximal positions in the list L among all activities in the block and in the outcoming network of the activity j respectively.

On the second stage we calculate a partial schedule for the *First-1* activities from the beginning of the list L via serial decoding procedure.

On the third stage we extend the partial schedule by scheduling of next *Last-First+1* activities of the list L via parallel decoding procedure. According to the procedure at each stage of scheduling we have a schedule time t and the correspondent eligible set E_t, i.e. a set of activities which could be started at t without violation of any constraints. There are exponentially number of possibilities to select a subset of activities from the eligible set to include into the schedule. In order to choose the subset we solve the multi-dimensional knapsack problem with objective function maximizing the resource

utilization ratio [7]:

$$\max \sum_{j \in E_t} x_j \frac{1}{K} \sum_{k=1}^{K} \frac{r_{jk}}{R_k},$$
$$s.t. \sum_{j \in E_t} x_j r_{jk} \le R_k - \sum_{j \in A_t} r_{jk}, k \in K,$$
$$x_j \in \{0,1\}.$$

Right side of the restriction $R_k - \sum_{j \in A_t} r_{jk}$ is a remaining capacity the k^{th} type resource at the time t. We use GRASP algorithm [2] to solve the problem. Solution obtained is a subset of eligible set E_t. Each activity of the subset is included into the partial schedule at the time t.

On the fourth stage we construct a list L' as follows. First, we set $L' = L$ in first *First-1* positions. Second, we put the next *Last-First+1* activities into list L' in non-decreasing order of its starting times in the partial schedule. Finally, remaining activities are listed in the list L' at the same order as in the list L.

3.3 Reverse Neighborhood

The reverse neighborhood $N_2(L)$ is constructed at the symmetrical manner. For this purpose it is sufficiently to transform current active schedule into right active schedule and replace outcoming network by correspondent *incoming network*.

3.4 Randomization

We use randomization of neighborhoods in order to reduce efforts per iteration. Each element of the neighborhoods is considered with some probability $p \le 1$.

3.5 Tabu List

We use a tabu list of constant length. As tabu status of a solution L we consider the sum of starting times:

$$TS(L) = \sum_{j=1}^{n} s_j$$

for the schedule $S(L)$.

3.6 Crossover operator

The crossover operator is based on the path relinking strategy [3]. We introduce a shift operator $Shift_{ij}(L)$ which

moves activity *i* immediately after activity *j* together with all successors t_k placed before than activity *j* in the list *L*:

$$Shift_{ij}(L):$$

$$L = (\dots i \dots t_1 \dots t_m \dots j \dots) \to L' = (\dots jit_1 \dots t_m \dots),$$

$$(i,t_1),\dots,(i,t_m) \in C$$

Let *L* and *L′* be two solutions. We construct a path $L=L_0, L_1,\dots,L_k = L', L_i = Shift_{pq}(L_{i-1})$. By $B(j)$ $(B'(j))$ denote the set of activities listed before than activity *j* in the list *L* (*L′*).

Definition 3. We say that the *order condition* is fulfilled for the activity *j*, if $B'(j) \subseteq B(j)$.

By *Order* denote the set of activities for which the order condition holds. The path construction procedure transforms the list *L* into the list *L′* successively decreasing the cardinality of the set $J \setminus Order$ monotonically.

3.7 Diversification

At the end of evolution, we generate a new population to explore new regions of the feasible domain. The new population should be selected as far as possible from the current one. For this purpose we consider the traveling salesman problem with the precedence relations. Let m_{ij} be a number of solutions (lists) of the current population in which the activity *j* is listed immediately after activity *i*. The objective function minimizes the closeness between new solution and solutions of the current population:

$$\min \sum m_{j_k j_{k+1}},$$

$$s.t. P_{j_{k+1}} \subseteq \{j_1 \dots j_k\}.$$

In order to solve the problem we use the GRASP algorithm [2].

4. Computational experiments

The algorithm was tested on the benchmarks taken from the PSPLib data set [6].

4.1 Allocation of local optima

We make the following experiment to study the allocation of local optima in the feasible domain. We randomly generate 1000 feasible solutions and apply local descent to everyone. For each local optima obtained we calculate size and weight. The size is a number of local optima from the sample which are located inside of the fixed range around the current local optimum. The weight is an average makespan of these local optima. Figure 1 shows the dependence between local optimum makespan and its weight. We see that *good* local optima are located near the *good* ones. The *bad* local optima are located near the *bad* ones. The experiment confirms that the genetic selection is quite promising search strategy for this problem.

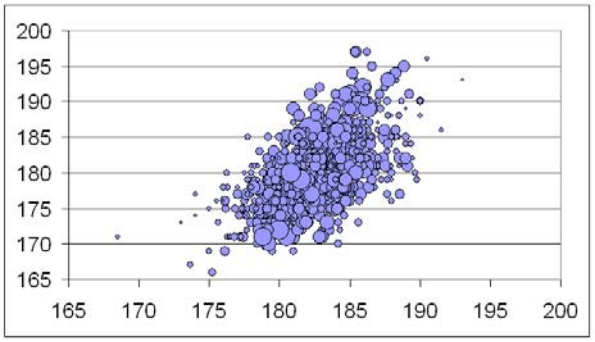

Figure 1: Allocation of local optima

4.2 Algorithm behavior

We test the algorithm in three modes: with single neighborhood without evolution, with variable neighborhood without evolution and with variable neighborhood and evolution. Table 1 shows average percent deviations from the best known upper bound for the most hard benchmark classes with 30 and 60 activities.

Test Classes	Single Neighb.	Variable Neighb.	Evolutionary Algorithm
J3013	2,49	0,00	0,00
J3029	1,96	0,00	0,00
J3045	1,93	0,00	0,00
J6013	3,47	0,96	0,61
J6029	4,01	0,93	0,44
J6045	2,90	0,17	0,18

Table 1: Average percent deviation from the best known upper bound

One can see that the algorithm with variable neighborhood provides better results than algorithm with single neighborhood. For most benchmark classes the evolutionary algorithm overcomes the algorithm without evolution. Sum total 60 benchmarks were considered. In 49 cases best known solutions were obtained. For two instances with 60 activities and for one instance with 120 activities the algorithm found new best solutions.

4. Acknowledgement

This research was supported by the Russian Foundation for Basic Research, grant No. 03-01-00455.

References

1. Blazewich J., Lenstra J. K., Rinooy Kan A. H. G. "Scheduling subject to resource constraints: Classification and complexity". *Discr. Appl. Math.* 1983; 5:11--24
2. Feo T. A., Resende M. G. C. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization* 1995; 6:109--133
3. Glover F., Laguna M. "Tabu Search". Kluwer Academic Publishers, Boston/Dordrecht/London, 1997
4. Hansen P., Mladenovic N. "Developments of Variable Neighborhood Search". In: Ribeiro C., Hansen P. (eds.) *Essays and Surveys of Metaheuristics*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002, pp 415--440
5. Kolisch R., Hartmann S. "Heuristic algorithms for the resource-constrained project scheduling problem: classification and computational analysis". In: Weglarz J. (ed.) *Project Scheduling: Recent Models, Algorithms and Applications*. Kluwer Academic Publishers, 1999, pp 147--178
6. Kolisch R., Schwindt C., Sprecher A. "Benchmark instances for project scheduling problems". In: Weglarz J. (ed.) *Handbook of recent advances in project scheduling.* Kluwer, Dordrecht, 1998, Chapter 9
7. Valls V., Ballestin F., Quintanilla S. "A Population-based Approach to the Resource-Constrained Project Scheduling Problem". *Technical report* 10-2001, University of Valencia, 2001