Variable Neighborhood Search for the Resource Constrained Project Scheduling Problem

Evgenii N. Goncharov[1,2](✉)

[1] Sobolev Institute of Mathematics, prosp. Akad. Koptyuga, 4, Novosibirsk, Russia
[2] Novosibirsk State University, str. Pirogova, 1, Novosibirsk, Russia
gon@math.nsc.ru
http://www.math.nsc.ru/

Abstract. We consider the resource-constrained project scheduling problem (RCPSP) with respect to the makespan minimization criterion. The problem accounts for technological constraints of activities precedence together with resource constraints. Activities preemptions are not allowed. The problem with renewable resources is NP-hard in the strong sense. We propose a variable neighborhood search algorithm with two neighborhoods. Numerical experiments based on standard RCPSP test dataset j120 from the PCPLIB library demonstrated that the proposed algorithm produces better results than existing algorithms in the literature for large-sized instances. For some instances from the dataset j120 the best known heuristic solutions were improved.

Keywords: Project management · Resource-constrained project scheduling problem · Renewable resources · Variable neighborhood search

1 Introduction

We consider the resource constrained project scheduling problem (RCPSP) with precedence and resource constraints. The RCPSP can be defined as a combinatorial optimization problem. A partial order on the set of activities is defined with a directed acyclic graph. We know duration, the set and amounts of consumed resources, for every activity. We assume that at every unit interval of the planning horizon $\hat{T}$ the same number of resources is allotted, and the resources are assumed to be unbounded outside the project horizon $\hat{T}$. All resources are renewable. Activities preemptions are not allowed. The objective is to schedule the activities of a project so as to minimize the project makespan. According to the classification scheme proposed in [19] this problem is denoted as $m, 1|cpm|C_{\max}$. According to the classification proposed in [3], this problem is

The work was supported by the program of fundamental scientific researches of the SB RAS, project No. 0314-2019-0014.

I. Bykadorov et al. (Eds.): MOTOR 2019, CCIS 1090, pp. 39–50, 2019.
https://doi.org/10.1007/978-3-030-33394-2_4

denoted as $PS \mid prec \mid C_{\max}$. As a generalization of the job-shop scheduling problem the RCPSP is NP-hard in the strong sense [1] and is actually one of the most intractable classical problems in practice. Worth noting that introducing cumulative resources into the same problem makes the problem solvable with polynomial complexity [9].

It may be conceivable to use optimal methods only for projects of small size. An exact-solution approaches have been developed in [2,8,27,31]. For larger problems, one needs heuristics to get the best solution within a convenient response time, and heuristics remain the best way to solve these problems efficiently. The construction methods are based on a scheduling scheme and an activity selection mechanism made by one or more priority rules or sampling techniques. Papers by Brucker [3], Kolisch and Hartmann [23], Herroelen et al. [18], Kolisch and Padman [21], Kolisch and Hartmann [22], Hartmann and Briskorn [15] survey the RCPSP, its numerous variants, and the solution techniques.

Many local search methods have been proposed to solve the RCPSP. These methods provide, in most cases, solutions better than construction methods as they proceed with a starting feasible schedule generated by one or many construction methods. As the local search methods are more effective than construction methods for large problems, this paper introduces a new hybrid approach combining concepts of tabu search [10] and variable neighborhood search [14] algorithm that uses the activity sequence encoding as a basis to search a neighborhood. This allows fast computation strategies in order to provide very good schedules in a real-time environment. The neighborhood search (NS) algorithm is used to improve one feasible solution by fixing the start times of some activities and rescheduling other activities. Palpant et al. [29] proposed five selection methods. Their numerical experiments showed that the "Block" selection method clearly outperformed other methods. Therefore, we use only the Block selection method to form the sub-problem in all NS operators. In this paper we propose a variable neighborhood search [14]. We use two alternative versions of neighborhoods. In fact, one of them is a modification of that proposed in [29], and the other is a modification of that proposed in [20]. The quality of the proposed algorithm has been examined for dataset j120 from the electronic library PSPLIB [25]. We provide results of the numerical experiments. For the dataset j120 (50000 and 500000 iterations) we have obtained one of the best average deviations from the critical path value. For 8 instances from the dataset j120 we have found the best (previously unknown) heuristic solutions.

2 Problem Setting

The RCPSP problem can be defined as follows. A project is taken as a directed acyclic graph $G = (N, A)$. We denote by $N = \{1, ..., n\} \cup \{0, n+1\}$ the set of activities in the project where activities 0 and $n+1$ are dummy. The latter activities define the start and the completion of the project, respectively. The precedence relation on the set N is defined with a set of pairs $A = \{(i, j) \mid i -$

−precedes $j\}$. If $(i,j) \in A$, then activity j cannot start before activity i has been completed. The set A contains all pairs $(0,j)$ and $(j,n+1)$, $j = 1,...,n$.

We have a set of renewable resources K, for each resource type $k \in K$ there is a constant availability $R_k \in Z^+$ throughout the project horizon $\hat{T}$. Activity j has deterministic duration $p_j \in Z^+$. The profile of resource consumption is assumed to be constant for every activity. So, activity j requires $r_{jk} \geq 0$ units of resource of type k, $k \in K$ at every time instant when it is processed. We assume that $r_{jk} \leq R_k,\ j \in N,\ k \in K$.

Now, we introduce the problem variables. We denote by $s_j \geq 0$ the starting time of activity $j \in N$. Since activities are executed without preemptions, the completion time of activity j is equal to $c_j = s_j + p_j$. We define a schedule S as an $(n+2)$-vector $(s_0, ..., s_{n+1})$. The completion time $T(S)$ of the project corresponds to the moment when the last activity $n+1$ is completed, i.e., $T(S) = c_{n+1}$. We denote by $J(t) = \{j \in N \mid s_j < t \leq c_j\}$ the set of activities which are executed in the unit time interval $[t-1, t)$ under schedule S. The problem is to find a feasible schedule $S = \{s_j\}$ respecting the resource and precedence constraints so that the completion time of the project is minimized. It can be formalized as follows: minimize the makespan of the project

$$T(S) = \max_{j \in N}(s_j + p_j) \tag{1}$$

under constraints

$$s_i + p_i \leq s_j, \quad \forall (i,j) \in A; \tag{2}$$

$$\sum_{j \in J(t)} r_{jk} \leq R_k,\ k \in K,\ t = 1, ..., \hat{T}; \tag{3}$$

$$s_j \in \mathbf{Z}^+, \quad j \in N. \tag{4}$$

Inequalities (2) define activities precedence constraints. Relation (3) corresponds to the resource constraints. Finally, (4) defines the variables in question.

3 Variable Neighborhood Search

3.1 Solution Representation

We represent a feasible solution as an activity list [23]. Feasible solution is encoded by the list of activities $L = (j_0, ..., j_{n+1})$. All lists under consideration are assumed to be compatible with the precedence relations. For an arbitrary list L, the serial decoding procedure (S-SGS) calculates the active schedule $S(L)$ [23]. It is known that there is an optimal schedule among the active schedules. A schedule is called active if the starting times of the activities are such that no activity can be started earlier of its starting time without violating either a precedence relation or a resource constraint. The parallel decoder (P-SGS) sequentially considers increasing moments of time, and schedules a subset of the eligible activities to start at this moment.

3.2 Resource Weights

We use the following heuristic rule to operate with a solution being evaluated. In the preliminary stage, before VNS algorithm has started, we find the degree of scarcity for each resource and rank them, assigning them with a weight. We denote w_k the weight of a resource of type $k,\ k = 1, ..., K$. If we have resources weights, we can compare them, giving priority to those where the higher priority (scarce) resources are used rationally, i.e., give less surplus of unused resources. We denote the weight v_j of activity j as

$$v_j = \sum_{k \in K} r_{jk} w_k / R_k.$$

We determine the degree of relative scarcity for the resources by solving a relaxed problem. For this purpose, we weaken the renewability condition for the resources and consider a problem with cumulative resources.

$$T(S) = \max_{j \in N}(s_j + p_j) \tag{5}$$

under constraints

$$s_i + p_i \leq s_j, \quad \forall (i, j) \in A; \tag{6}$$

$$\sum_{t'=1}^{t} \sum_{j \in J(t')} r_{jk} \leq \sum_{t'=1}^{t} R_k, \quad k \in K,\ t = 1, ..., \hat{T}. \tag{7}$$

$$s_j \in \mathbf{Z}^+, \quad j \in N. \tag{8}$$

The fast approximated algorithm to solve problem (5)–(8) is known, it's computational complexity depends on the number n of activities as a function of order n^2. In the case of real-valued activity durations, the algorithm is asymptotically exact with absolute error that tends to zero as the problem dimension grows [7]. In addition, for integer-valued activity durations the exact algorithm is developed [8,9]. In this work we consider a problem with integer-valued activity durations, so we can use any one of these two algorithms to solve the relaxed problem (5)–(8). We choose the first one. By applying this algorithm, in addition to the solution of the relaxed problem, we also get the residue for each cumulative resource that allows us to define the degree of scarcity for all resources: the less is a resource's surplus the scarcer it is. As a final step we apply the resulting resource ranking rule obtained in the relaxed problem to the original problem (1)–(4).

3.3 Block of Activities

For a given feasible schedule $S = (s_0, ..., s_{n+1})$ and a core activity $j = 1, ..., n$, the NS operator reschedules a set of activities, A_j^s, while keeping the start times of all other activities. Let P be a predetermined number of activities that will be rescheduled. The value of P influences the computational time to obtain a neighborhood solution by rescheduling. Smaller value of P usually means fewer activities to be rescheduled and less time to obtain the new schedule. The following Block selection method is used to create A_j^s [29].

$CreateBlock(j, S) \rightarrow A_j^s$.

1. $A_j^s = j$; $b = 0$; create a random order for all activities in $A/\{j\}$. Let i is the first activity in the order.
2. If $s_j - p_i - b \le s_i \le s_j + p_j + b$, $A_j^s = A_j^s \bigcup \{i\}$.
3. If $|A_j^s| = P$, go to Step 6.
4. If i is the last activity among the ones not belonging to A_j^s based on the order defined in Step 1, $b = b + 1$.
5. Let i be the next activity among the ones not belonging to A_j^s based on the order defined in Step 1. Go to Step 2.
6. END.

The Block selection method basically selects a set of P activities that are overlapped or close to activity j in a given feasible schedule.

3.4 The Initial Solution

We can use any available method to generate a good initial solution rapidly. The choice of the algorithm for the initial solution is not critical for the local search methods. Gagnon et al. [6] noted that there is some dilemma concerning the choice of the initial solution used by a NS method adaptation. Starting with a very good solution doesn't let enough space to find a significant improvement. On the other side, it may take a long computation time to improve a bad starting solution. We can use, for example, the S-SGS and P-SGS schemes, stochastic methods with the forward-backward improvement procedure (FBI) [32], greedy algorithms. In this paper we use the stochastic greedy algorithm [13] to build up a starting schedule solution.

3.5 Tabu List Management

The NS method exploits the knowledge gained from the solutions considered previously. This knowledge is maintained in a tabu list used as a memory in order to avoid cyclicality, i.e. repeating of recent transformations applied to obtain the solutions under evaluation. Recency tabu tenure is recorded by keeping at each entry of the tabu list, of attributes on the last visited solutions. We use tabu list of the constant length. As a tabu status of an arbitrary solution L we consider the sum of the starting times:

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

for the schedule $S(L)$.

3.6 Neighborhood A

The first neighborhood $N_A(S)$ is the modification of the scheme proposed in [30]. For a given feasible schedule $S = \{s_0, s_1, ..., s_n, s_{n+1}\}$ and a core activity $j \in A$, we determine the block of activities A_j^s. The NS operator reschedules a set of activities A_j^s, while keeping the start times of other activities. The rescheduling sub-problem is formed by the following steps. We fix the start times of all the activities not belonging to the set A_j^s and release resources used by all the activities from A_j^s in each time period t. The available amount of resource k for activities from A_j^s in period t is R_k minus the resource used by all the activities not belonging to the set A_j^s in period t. Then we derive an earliest start time (EST) and a latest finish time (LFT) for each activity $i \in A_j^s$ as $EST_i = max\{s_l + p_l,\ \forall\ l \notin A_j^s\ and\ (l,i) \in A\}$ and $LFT_i = max\{s_l,\ \forall\ l \notin A_j^s\ and\ (l,i) \in A\}$.

(EST_i, LFT_i) defines a time window for activity i that could be rescheduled in order to guarantee that the new schedule is still feasible for all activities that will not be rescheduled. The rescheduling problem is to reschedule all the activities from A_j^s to minimize their makespan while meeting the resource restriction of each period and time window constraints defined by (EST_i, LFT_i).

Palpant et al. [29] used a commercial integer linear programming solver to obtain the optimal solution for the rescheduling problem. Proon and Jin [30] adopts the forward or backward serial scheduling generation schemes (S-SGS) [23] to solve the rescheduling sub-problem. In this paper we propose a modification of this algorithm [30]. In each iteration, a new random vector is produced for the activities $i \in A_j^s$ as a priority list. We order the activities in A_j^s by decreasing their weights v_j. The vector is created iteratively by randomly picking the next activity from the ordered list among all unselected activities whose precedent activities in A_j^s have been selected. Following the priority list, one activity by one is moved to the earliest (latest) start time that is precedence- and resource-feasible and satisfies the time window (EST_i, LFT_i). Once all the activities $i \in A_j^s$ are rescheduled, the activities that do not belong to A_j^s are added to form a complete feasible solution. A global left shift is then performed on all the activities in A to possibly reduce the makespan. The resulting new schedule is compared with the previous solution before applying the NS operator. If the makespan is improved, the resulting schedule replaces the previous schedule and the NS operator stops. If there is no improvement, as long as the number of iterations has not reached a predefined limit, λ, the S-SGS is applied on the schedule with a new random priority list as the next iteration.

3.7 Neighborhood B

As the neighborhood $N_B(S)$ we use the modification of the scheme proposed in [20]. For a given list of activities L (and a correspondent active schedule $S = \{s_0, ..., s_{n+1}\}$) and a core activity $j \in A$, we determine the block of activity A_j^s. If the block contains at least one predecessor of the activity j then we put

A_j^s to be empty. We represent the list L in the form of three consecutive lists $L = A^1, A_j^s, A^2$.

The element L' of the neighborhood $N_B(S)$ is constructed for each activity $j \in A$ by using the non-empty block A_j^s. The list L' is obtained from the list L by the following steps. We fix the start times of all the activities from the set A^1 and release resources used by all the activities from set A^1 in each time period t. We calculate a partial schedule for the activities belonging to set A^1 via the serial decoding procedure. Then we extend the partial schedule by scheduling activities belonging to set A_j^s via the parallel decoding procedure. According to the procedure for each schedule time t we have the corresponding eligible set E_t, i.e. a set of activities which could be started at t without violation of any constraints. There are exponentially many possibilities to select a subset of activities from the eligible set to include into the schedule. We solve the multi-dimensional knapsack problem with objective function maximizing the weighted resource utilization ratio [35]

$$max \sum_{j \in E_t} x_j \sum_{k \in K} \frac{w_k r_{jk}}{R_k}, \tag{9}$$

$$\sum_{j \in E_t} r_{jk} x_j \leq R_k - \sum_{j \in J(t)} r_{jk} \quad k \in K, \tag{10}$$

$$x_j \in \{0, 1\}, \quad j \in N. \tag{11}$$

The right-hand side of the restriction is a remaining capacity of the resource type k at the time t. We use Greedy Randomized Adaptive Search Procedures (GRASP) to solve the problem. Note that in the process of solving the problem (9)–(11), the advantage may be gained not by the activities that make the best use of resources, but by those that make the best use of more scarced resources.

Finally, we construct the list L' as follows. We put the activities $\in A_j^s$ into the list L' in non-decreasing order of its starting times in the partial schedule. Remaining activities are listed in the list L' at the same order as in the list L. The schedule $S(L')$ is called the neighbor sample for the schedule S. The set that contains all neighbor samples is called a neighborhood of the schedule S and is denoted by $N_B(S)$.

3.8 Algorithm Outline

Step 1. Generate the initial schedule S, and set $T^* := T(S),\ S^* := S$. Tabu list TL is set empty.

Step 2. Until the stopping criterion is satisfied, the following is done.
Step 2.1. Choose a neighborhood equally probable.
Step 2.2. Find the neighbor sample S', not prohibited by the tabu list TL.
Step 2.3. If $T(S') < T^*$, then we assume $T^* := T(S'),\ S^* := S'$.
Step 2.4. Update the tabu list TL and set $S := S'$.

As a stopping criteria we consider reaching the maximum number of sequences evaluated, denoted as λ. The value of T^* is the result of the algorithm. If the value of T^* does not change for a certain (predefined) iterations limit, then we change the block size (parameter P). We make this change of parameter P a predefined number of times. Finally, if the value of T^* does not change a predefined number of times, we generate a new initial schedule.

4 Numerical Experiments

The VNS algorithm was coded in C++ in the Visual Studio system and run on a 3.4 GHz CPU and 8 Gb RAM computer under the operating system Windows 7. In order to evaluate the performance of the proposed VNS algorithm, we use the standard set presented in Kolisch and Sprecher [25] referred as j120. These instances are available in the project scheduling library PSPLIB along with their the best-known values. The dataset j120 contains 60 series of instances, 10 instances in each series, 600 instances in total. Each instance considers four types of resources. Three parameters: network complexity (NC), resource factor (RF) and resource strength (RS) are combined together to define the full factorial experimental design. The NC defines the average number of precedence relations per activity. The RF sets the average percent of various resource type demand by activities. The RS measures scarcity of the resources. Zero value of the RS factor corresponds to the minimum need for each resource type to execute all activities while the RS value of one corresponds to the required amount of each resource type obtained from the early start time schedule. The parameter values used to built up these instances for the set j120 are: $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$ and $RS \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. It is known [37] that values of the parameters $RF = 4$, $RS = 0.2$ match hard enough series. Identifiers j12016, j12036, j12056, j12011, j12031, j12051 with $n = 120$ correspond to the series with the largest gap between the best solutions found and the length of the critical path. Each triplet of such identifications matches values $NC = \{1.5; 1.8; 2.1\}$, respectively.

See Kolisch et al. [24] for the process of how the instances were created. The instances can be found in Kolisch and Sprecher [25] and are downloadable at http://www.om-db.wi.tum.de/psplib/.

The measure of the solution quality is the average percent deviation (APD) from the lower bounds obtained by the critical path algorithm [24] for instances in the dataset j120 which optimal solutions are unknown. It is customary to compare the heuristic efficiency by restricting to the same number of schedules evaluated. In Table 1 one can find comparison the VNS algorithm performance with the previous results of experimental evaluation of competitive heuristics for the dataset j120. Limit of schedules λ we set at 50000 and 500000. The scrutiny of the presented results clearly shows the good performance of the proposed VNS algorithm: for $\lambda = 50000$ it was the third, and for $\lambda = 500000$ it shows the second result.

Furthermore, for 8 instances from the dataset j120 we obtained the best (previously unknown) heuristic solutions, they are currently presented in the

Table 1. Average deviations from the critical path for dataset j120.

Algorithm	Reference	APD, %	
		$\lambda = 50000$	$\lambda = 500000$
GA	Goncharov and Leonov [12]	30,50	**29,74**
VNS	This paper	30,56	29,88
Biased random-key GA	Goncalves [11]	32,76	30,08
GANS	Proon and Jin [30]	**30,45**	30,78
ACOSS	Chen et al. [36]	30,56	–
DBGA	Debels and Vanhoucke [5]	30,69	–
GA	Debels and Vanhoucke [5]	30,82	–
GA - Hybrid, FBI	Valls et al. [34]	31,24	30,95
Enhanced SS	Mobini et al. [28]	31,37	–
Scatter search - FBI	Debels et al. [4]	31,57	30,48
GAPS	Mendes et al. [26]	31,44	31,20
GA, FBI	Valls et al. [33]	31,58	–
GA, TS-Path re-linking	Kochetov and Stolyar [20]	32,06	–
GA-Self adapting	Hartmann [17]	33,21	–
GA-Activity list	Hartmann [16]	34,04	–
Sampling-LFT, FBI	Tormos and Lova [32]	35,01	–
SGE-Priority rule, FBI	Goncharov [13]	35,08	–
GA-Priority rule	Hartmann [16]	36,51	–

PSPLIB library. We provide the list of the mentioned instances in Table 2.As one can see from Table 1, the VNS algorithm conceded to the genetic algorithm on the whole dataset j120. But at the same time, as one can see from Table 2, the VNS algorithm has found the previously unknown best heuristic solutions exclusively on the series with the largest gap between the best solutions found and the length of the critical path. Therefore, we can conclude that on such “hard series” of instances, the VNS algorithm shows better results in comparison with the GA algorithm.

Average processing time is 16 s for $\lambda = 50000$ and 150 s for $\lambda = 500000$.

Table 2. List of instances for which new heuristic solutions are obtained.

Dataset	Series	Instances
j120	11	3
j120	16	8
j120	31	4
j120	36	3
j120	51	3, 4
j120	56	7, 8

5 Conclusion

Authors have proposed a variable neighborhood search algorithm for the resource-constrained project scheduling problem with respect to the makespan minimization criterion. We have developed two versions of the neighborhoods. The algorithm uses a heuristic that takes into account the degree of criticality (scarcity) of the resources, which is derived from the solution of the relaxed problem with a constraint on the cumulative resources. We have conducted numerical experiments on sets of instances from the PSPLIB electronic library. The results of the computational experiments suggest that the proposed VNS algorithm is a very competitive heuristic and yields better results than several heuristics presented in the literature. For some instances from the dataset j120 the best known heuristic solutions were improved.

Further studies will be focused on constructing hybrid algorithms for the RCPSP problem.

References

1. Blażewicz, J., Lenstra, J.K., Rinnoy Kan, A.H.G.: Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**(1), 11–24 (1983)
2. Brucker, P., Knust, S., Schoo, A., Thiele, O.: A branch and bound algorithm for the resource-constrained project scheduling problem. Eur. J. Oper. Res. **107**, 272–288 (1998)
3. Brucker, P., Drexl, A., Möhring, R., et al.: Resource-constrained project scheduling: notation, classification, models, and methods. Eur. J. Oper. Res. **112**(1), 3–41 (1999)
4. Debels, D., De Reyck Leus, B.R., Vanhoucke, M.: A hybrid scatter search electromagnetism meta-heuristic for project scheduling. Eur. J. Oper. Res. **169**, 638–653 (2006)
5. Debels, D., Vanhoucke, M.: Decomposition-based genetic algorithm for the resource-consrtained project scheduling problem. Oper. Res. **55**, 457–469 (2007)
6. Gagnon, M., Boctor, F.F., d'Avignon, G.: A Tabu Search Algorithm for the Resource-constrained Project Scheduling Problem. ASAC (2004)
7. Gimadi, E.Kh.: On some mathematical models and methods for planning large-scale projects. models and optimization methods. In: Proceedings AN USSR Sib. Branch, Math. Inst., Novosibirsk. Nauka, vol. 10, pp. 89–115 (1988)
8. Gimadi, E.Kh., Goncharov, E.N., Mishin, D.V.: On some implementations of solving the resource-constrained project scheduling problem. Yugoslav J. Oper. Res. **29**(1), 31–42 (2019)
9. Gimadi, E.Kh., Zalyubovskii, V.V., Sevast'yanov, S.V.: Polynomial solvability of scheduling problems with storable resources and deadlines. Diskretnyi Analiz i Issledovanie Operazii, Ser. 2 **7**(1), 9–34 (2000)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
11. Goncalves, J., Resende, M.G.C., Mendes, J.: A biased random key genetic algorithm with forward-backward improvement for resource-constrained project scheduling problem. J. Heuristics **17**, 467–486 (2011)
12. Goncharov, E.N., Leonov, V.V.: Genetic algorithm for the resource-constrained project scheduling problem. Autom. Remote Control **78**(6), 1101–1114 (2017)

13. Goncharov, E.N.: Stochastic greedy algorithm for the resource-constrained project scheduling problem. Diskret. Anal. Issled. Oper. **21**(3), 10–23 (2014)
14. Hansen, P., Mladenovic, N.: Developments of variable neighborhood search. In: Ribeiro, C., Hansen, P. (eds.) Essays and Surveys of Metaheuristics, pp. 415–440. Kluwer Academic Publishers, Boston (2002)
15. Hartmann, S., Briskorn, D.: A survey of variants and extentions of the resource-constrained project scheduling problem. Eur. J. Oper. Res. **207**, 1–14 (2010)
16. Hartmann, S.: A competitive genetic algorithm for the resource-constrained project scheduling. Naval Res. Logistics. **45**, 733–750 (1998)
17. Hartmann, S.: A self-adaptive genetic algorithm for project scheduling under resource constraints. Naval Res. Logistics. **49**, 433–448 (2002)
18. Herroelen, W., De Reyck, B., Demeulemeester, E.: Resource-constrained project scheduling: a survey of recent developments. Comput. Oper. Res. **25**(4), 279–302 (1998)
19. Herroelen, W., Demeulemeester, E., De Reyck, B.: A classification scheme for project scheduling. In: Weglarz, J. (Ed.) Project Scheduling-Recent Models, Algorithms and Applications, International Series in Operations Research and Management Science, vol. 14(1), pp. 77–106. Kluwer Academic Publishers, Dordrecht (1998)
20. Kochetov, Yu., Stolyar, A.: Evolutionary local search with variable neighborhood for the resource-constrained project scheduling problem. In: Proceedings of 3rd International Workshop of Computer Science and Information Technologies. Russia, pp. 96–99 (2003)
21. Kolisch, R., Padman, R.: An integrated survey of deterministic project scheduling. Omega **49**(3), 249–272 (2001)
22. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: an update. Eur. J. Oper. Res. **174**, 23–37 (2006)
23. Kolisch, R., Hartmann, S.: Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In: Weglarz, J. (ed.) Project Scheduling: Recent Models, Algorithms and Applications, pp. 147–178. Kluwer Academic Publishers (1999)
24. Kolisch, R., Sprecher, A., Drexl, A.: Characterization and generation of a general class of resource-constrained project scheduling problems. Manage. Sci. **41**, 1693–1703 (1995)
25. Kolisch, R., Sprecher, A.: PSPLIB - a project scheduling problem library. Eur. J. Oper. Res. **96**, 205–216 (1996). http://www.om-db.wi.tum.de/psplib/
26. Mendes, J.J.M., Goncalves, J.F., Resende, M.G.C.: A random key based genetic algorithm for the resource constrained project scheduling problem. Comput. Oper. Res. **36**, 92–109 (2009)
27. Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L.: An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Manage. Sci. **44**, 715–729 (1998)
28. Mobini, M.D.M., Rabbani, M., Amalnik, M.S., et al.: Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. Soft Comput. **13**, 597–610 (2009)
29. Palpant, M., Artigues, C., Michelon, P.: LSSPER: solving the resource-constrained project scheduling problem with large neighborhood search. Ann. Oper. Res. **131**, 237–257 (2004)
30. Proon, S., Jin, M.: A genetic algorithm with neighborhood search for the resource-consrtained project scheduling problem. Naval Res. Logist. **58**, 73–82 (2011)

31. Sprecher, A.: Scheduling resource-constrained projects competitively at modest resource requirements. Manage. Sci. **46**, 710–723 (2000)
32. Tormos, P., Lova, A.: A competitive heuristic solution techniques for resource-consrtained project scheduling. Ann. Oper. Res. **102**, 65–81 (2001)
33. Valls, V., Ballestin, F., Quintanilla, M.S.: Justification and RCPSP: a technique that pays. Eur. J. Oper. Res. **165**, 375–386 (2005)
34. Valls, V., Ballestin, F., Quintanilla, S.: A hybrid genetic algorithm for the resource-consrtained project scheduling problem. Eur. J. Oper. Res. **185**(2), 495–508 (2008)
35. Valls, V., Ballestin, F., Quintanilla, S.: A population-based approach to the resource-constrained project scheduling problem. Ann. Oper. Res. **131**, 305–324 (2004)
36. Chen, W., Shi, Y.J., Teng, H.F., et al.: An efficient hybrid algorithm for resource-constrained project scheduling. Inf. Sci. **180**(6), 1031–1039 (2010)
37. Weglarz, J.: Project Scheduling. Recent Models, Algorithms and Applications. Kluwer Academic Publishers, Boston (1999)