Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 659297, 7 pages
http://dx.doi.org/10.1155/2013/659297

Research Article

An Iterated Local Search Algorithm for a Place Scheduling Problem

Shicheng Hu,[1] Zhaoze Zhang,[1] Qingsong He,[1] and Xuedong Sun[2]

[1] *School of Economics and Management, Harbin Institute of Technology, Weihai 264209, China*
[2] *School of Software, Sun Yat-sen University, Guangzhou 510275, China*

Correspondence should be addressed to Xuedong Sun; sunxdys@163.com

Received 15 August 2013; Accepted 9 September 2013

Academic Editor: Yunqiang Yin

We study the place scheduling problem which has many application backgrounds in realities. For the block manufacturing project with special manufacturing platform requirements, we propose a place resource schedule problem. First, the mathematical model for the place resource schedule problem is given. On the basis of resource-constrained project scheduling problem and packing problem, we develop a hybrid heuristic method which combines priority rules and three-dimensional best fit algorithm, in which the priority rules determine the scheduling order and the three-dimensional best fit algorithm solves the placement. After this method is used to get an initial solution, the iterated local search is employed to get an improvement. Finally, we use a set of simulation data to demonstrate the steps of the proposed method and verify its feasibility.

1. Introduction

One of the main works in ship manufacturing is hull section processing. This process is always implemented on the block manufacturing platform, which usually takes a long time to finish. Because of the limited area, it cannot meet the requirement of the shipbuilding enterprises. Effective use of limited resources has become a concern for the shipbuilding enterprises.

For utilization of the platform in block manufacturing, we propose a place scheduling problem. It includes two related problems. One is resource constrained project scheduling problem (RCPSP), and the other is bin packing problem.

The RCPSP is described as an individual project that includes n activities; these activities cannot stop during its processing time. There are two kinds of restrictions during the process. One is precedence constraints. It means that activity j cannot start before its predecessor activity in P_j. The other one is resource constraints. To make the activity work smoothly, the activity j needs k_{jr} unit resources during its process. We decide the finish time of every activity under the priority and resource constraints to get the shortest project makespan. Hartmann proposes algorithm for static job scheduling problem which belongs to NP-hard problem [1]. Kelley, Alvar-Valdes, and Tamarit give approximate and exact methods [2, 3]. Alcaraz and Maroto develop a heuristic method based on priority rules [4]. It consists of two parts, a scheduling generation method and a priority rule. Kolisch summarizes the scheduling generation methods into two categories, serial and parallel methods [5]. Serial schedule is implemented by the addition of activities, while parallel schedule is produced with the increasing of the time. Baker considers that every activity should only be scheduled once for a single path [6]. Mingozzi et al. propose a RCPSP problem based on a new mathematical model and use an exact method [7].

Packing problem is about how to put more boxes into a bin in which the bottom area is limited. We can divide the packing problem into two categories for discussion: two-dimensional and three-dimensional packing problems. For two-dimensional packing problem, Burke et al. propose a new placement heuristic for the rectangular cutting issues [8]. Egeblad and Pisinger provide a new method for the two-dimensional packing problem and extend to three-dimensional packing problem [9]. In three-dimensional packing, the method of 3BF is based on searching the list in

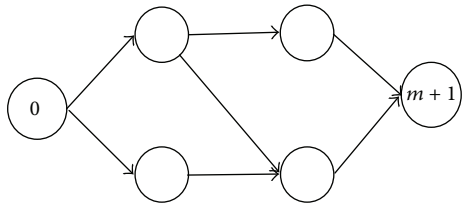

Figure 1: A network diagram.

order. Similar to 2BF algorithm, it searches dynamically to find suitable boxes to be put in the available space. The 3BF defines that blank area is at the farthest point where the boxes are to be put. The 3BF is raised by Martello et al. [10]; it solves the problem of how to select suitable boxes, and the point cannot be blocked by other boxes.

This paper is organized as follows. Section 2 gives a mathematical model. Section 3 proposes an iterated local search algorithm. Section 4 presents a set of simulation examples to demonstrate the method. Section 5 comes up with a summary for this paper.

2. Place Scheduling Problem

The project consists of m activities $M = \{1, 2, \ldots, m\}$. Every activity should be processed on the required platform, that is, place. We define it as $r_{ir} = a_{ir} \times b_{ir}$, where a_{ir} and b_{ir} denote the length of the place r that activity i requires, respectively. The duration that activity i requires is defined as t_i. S_i and F_i represent the start and finish time of the activity i, respectively. In the project, there are n kinds of place recourses, and every place resource is given by $R_{ir} = A_{ir} \times B_{ir}$, where A_{ir} and B_{ir} denote the length and width of the available place r, respectively. There are n types of places $N = \{1, 2, \ldots, n\}$. All activities have priority constraints; Q_j is the predecessors of j. We assume that the start time of the project is 0. We decide the start time of the activities so that the makespan of the project is minimized.

For modeling this problem, two dummy nodes 0 and $m+1$ are introduced. 0 is the predecessor of all the activities, and $m + 1$ is the successor of all the activities in the project. The duration and required place resource of the two dummy nodes are 0. Obviously, the finish time of $m + 1$ is the makespan of the project, so the problem can be formulated as follows:

$$\text{minimize } F_{m+1}. \tag{1}$$

Priorities between activities can be defined as

$$S_i + t_i \le S_j, \quad i \in Q_j. \tag{2}$$

We can show it by a network diagram as in Figure 1.

We regard an activity as a cuboid, and it is defined as $w_{ir} = a_{ir} \times b_{ir} \times t_i$,

$$w_{ir} \cap w_{jr} = \varphi, \quad \forall i, j \in M, \quad i < j, \quad \forall r \in N, \tag{3}$$

$$\begin{aligned} &[g_i b_{ir} + (1 - g_i) a_{ir}] \\ &\quad \le x_i + [g_i b_{ir} + (1 - g_i) a_{ir}] \\ &\quad \le B_r, \end{aligned} \tag{4}$$

$$\begin{aligned} &[g_i a_{ir} + (1 - g_i) b_{ir}] \\ &\quad \le y_i + [g_i b_{ir} + (1 - g_i) b_{ir}] \\ &\quad \le A_r, \end{aligned} \tag{5}$$

$$g_i \in \{0, 1\}. \tag{6}$$

Constraint (3) denotes that any two cuboids cannot overlap. Constraints (4) and (5) denote that any cuboid should be finished within the available place. Constraint (6) means that the cuboid can rotate ninety degrees, which is represented by two values.

3. Iterated Local Search Algorithm

3.1. The Initial Solution Method. For the initial solution, we develop a hybrid heuristic method of project scheduling problem and packing problem to solve it. Each activity is abstracted as a cuboid and is put in the available place. As we put the cuboid into the space, we divide different places according to different heights. Every place is a two-dimensional area.

There are many methods of how to place the rectangle. We will use best fit [8] heuristic algorithm to solve our problem. We cannot just rely on the size of the rectangle to judge which is more suitable, because cuboids have their sequences. But in the Best Fit method, the standard to select the cuboid to be moved only relies on the lengths and widths of the unselected cuboids. Inspired by this, we should consider the sequence before we select the cuboids using the Best Fit method. We place the larger bottom area first; if they are equal, select the one with the higher height.

Before placing it, we can regard the bottom area as an object in a two-dimensional packing problem. After packing, we can use some methods of the three-dimensional packing. We can put them according to some rules. Length, width, and height can be interchanged when we put them. But in our problem, the height is the duration, so only length and width can be interchanged. This is different from the three-dimensional packing. There are many methods regarding the three-dimensional packing problem. We mainly use the algorithm of Crainic et al. which is based on the extreme point [13]. Its main objective is to determine the placement. We place the bottom-left-rear point on the selected placement. The extreme point is represented as $\{x, y, t\}$. We select the location based on the extreme point.

Table 1: Priority rules.

Priority rule	Paper	$v(j)$
MTS	Alvar-Valdes and Tamarit [3]	$\lvert\overline{S}_j\rvert$
LST	Alvar-Valdes and Tamarit [3]	$\mathrm{LFT}_j - d_j$
GRPW	Alvar-Valdes and Tamarit [3]	$d_j + \sum_{i \in s_j} d_i$
WRUP	Ulusoy and Özdamar [11]	$0.7\left\lvert S_j\right\rvert + 0.3\sum_{r \in R} k_{jr}/K_r$
LFT	Davis and Patterson [12]	LFT_j
MSLK	Davis and Patterson [12]	$\mathrm{LFT}_j - \mathrm{EFT}'_j$

We give the description on how to select the position as follows.

(1) Put the first activity on the origin $(0, 0, 0)$, and then it will produce three new extreme points.

(2) Put the second activity by our $t - y - x$ rule which means that the smaller t among the extreme points is preferred. Then, the smaller y will be considered. Finally, the x will be selected. Meanwhile, we consider rotation when we place the boxes.

(3) If an activity cannot match the placement, we will use another activity. In the worse condition, if all activities cannot match the placement, we will find another placement to find a bigger t as a placement.

(4) If the extreme point of t_i cannot meet the activity's requirement, we will delete this point. In other words, we will not search for it again in the next stage.

Priority rule based scheduling is an important method to RCPSP. It is straightforward and easy to use. There are many priorities; the better ones are presented in Table 1.

We cannot use these rules directly, but we can learn some ideas, like MTS. It is sorted by the minimum of the latest finish time. We use it in the improvement algorithm. In our rule, the larger bottom area will be considered first; if they are the same, the longer duration (i.e., t) will be considered.

It is closely related to priority and schedule generation scheme (SGS). SGS gets a feasible schedule by local search. Local search means arranging part of activities in stage. There are two different SGSs. We call them serial and parallel. The arrangement is described as follows.

(1) We select the activities from the network graph according to the priority rule. Then, we put them in the set D_g which means that activities can be arranged.

(2) We will select an activity from D_g. The larger bottom area will be considered first; if they are the same, the longer duration (i.e., t) will be considered.

(3) We put the activities into the set S_g which means that activities have been scheduled. This is one cycle. Then, a new cycle will begin.

(4) According to the rule, we arrange new activities which are in the set D_g on the extreme point. If it cannot be placed, we will consider the rotation. If not, the second in the set D_g will be placed. The process will not finish until all the activities are scheduled.

There are some kinds of placement. We consider putting the cuboid along the x-axis or the y-axis.

3.2. Optimization Solution Method. For the initial solution, we solve it by a rule, so the result is not necessarily the best. Iterated local search will be used to improve the initial solution by searching the local neighborhood to produce local optimum.

3.2.1. Local Search. Savelsbergh proposes the ideas of local search to solve the time window problem [14]. Voudouris and Tsang apply local search in the traveling salesman problem [15]. Schaerf and Meisels use local search in the employee timetabling problems [16]. Marco et al. reviews the local search and introduces specific applications [17]. The idea of the local search is to search for the neighborhood of the initial solution and then iterate constantly. Local search is not only attractive but also very successful in practice. In recent research, combinatorial optimization and continuous optimization problems can be applied by local search. There is essential difference in search space type between combinatorial optimization problems and continuous optimization problems. The space of combinatorial optimization is finite, but the space of the continuous optimization is infinite. We use the space of the combinatorial optimization, and then the local search will be used to find the optimal.

Lucio et al. use the local search in a network planning algorithm that incorporates uncertain traffic demands [18]. Cabido et al. propose local search driven by adaptive scale estimation on Gpus [19]. Funke et al. use it in the vehicle routing and scheduling problems [20]. Tricoire proposes multidirectional local search [21]. Ricca and Simeone apply it in political districting [22]. Mills et al. propose an extended guided local search to the quadratic assignment problem [23].

3.2.2. Iterated Local Search. Iterated local search is widely used because of its simplicity and efficiency. It starts the search from the initial solution. So, before we use it, an initial solution should be obtained. In general, a more optimal solution can be found based on several initial solutions. For nondeterministic polynomial (NP) problem, it is difficult to enumerate all circumstances, so we find a generally better

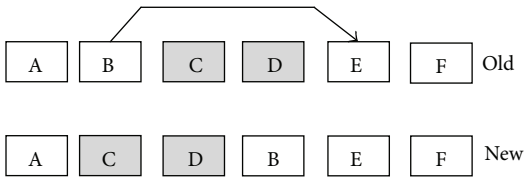

Figure 2: The demonstration of local search.

solution. From an initial solution, we can find a better solution by constantly search. It may take more time.

The key to the iterated local search is searching the local neighborhood constantly. Then, we decide whether it is the solution we want. It is as follows.

(1) Get an initial solution.

(2) Start the local search from the initial solution.

(3) Get another solution by perturbation.

(4) Start the local search from the recent solution.

(5) Compare and choose a better solution.

(6) Repeat until the termination condition is met.

For combinatorial optimization problems, the candidate feasible solutions in the neighborhood of the feasible solution are countable. Local optimum can be defined as $\forall s \in N(s^l)$. We get $f(s) \leq f(s^l)$. The f is the objective function. We can enumerate all the candidate solutions and check them if they are better than the current solution. If there is no better solution, the current solution is the optimal.

It is very important to learn the concept of neighborhood for the local search. If a feasible solution s' can change into a feasible solution s by a step, a feasible solution s' can be called a neighboring feasible solution s. Neighborhood is the set of the feasible solution s. It begins by local search to find local optimum. The initial solution changes after being constantly perturbed. We can find a new local optimum this way. The purpose of perturbation is to find different solutions. Then, we compare them to find a better one.

The local search is demonstrated in Figure 2. There are six activities, A, B, C, D, E, and F. The old stands for the previous solution. The new means the current solution after the local search. Local search is based on the old solution; the new one is the one among the neighboring feasible solutions of the old.

In Figure 2, the B is inserted into the front of the D to form a new solution. In our problem, we can consider the activities that start at the same time to exchange to get a new sequence.

4. Simulation Experiment

We use a set of simulation data to show our algorithm. This set of data is adapted from PSPLIB by Kolisch and Sprecher [24], in which any two resources are set as the length and width of a place, respectively. One of the generated instances is described as follows.

There are two places; the length and width of place 1 are 3 and 2 are 3 and 2 for place 2. The activities of odd numbers work on place 1. The activities of even numbers work on place 2. The network diagram is given in Figure 3.

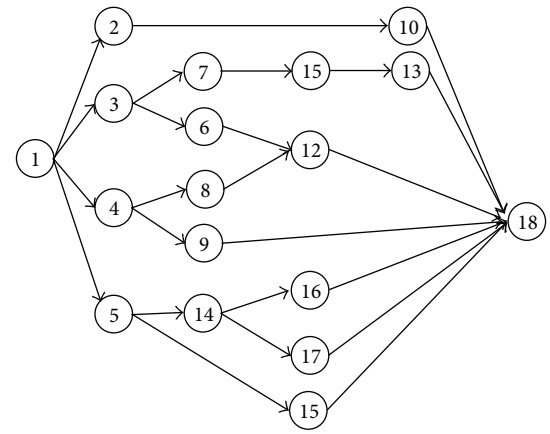

Figure 3: The network diagram.

Table 2: The parameters of activities.

Activity	Duration	Length, width
1	0	0, 0
2	8	2, 1
3	4	2, 2
4	6	2, 1
5	3	2, 2
6	8	2, 1
7	5	2, 1
8	6	3, 1
9	2	3, 2
10	3	2, 1
11	7	2, 2
12	2	2, 2
13	7	3, 1
14	9	2, 1
15	4	3, 2
16	6	2, 1
17	2	2, 1
18	0	0, 0

The data of activities is presented in Table 2.
The arrangement steps described as follows.

(1) At first, the lowest time is 0, the activities that can be arranged are {2, 3, 4, 5}, and the odd activities of place 1 are {3, 5}. The bottom area of them is the same, so we compare their duration. We arrange activity 3 at first. According to our rule, we put bottom left rear of activity 3 on the (0, 0, 0). Then, it produces three new extreme points; see Figure 4.

(2) At time 0, there is not enough space for activity 5, so we will consider it later. On place 2, both activity 2 and activity 4 can be arranged. We lay activity 2 first on the (0, 0, 0). Then, it produces three new extreme points (2, 0, 0), (0, 1, 0), (0, 0, 8); see Figure 5.

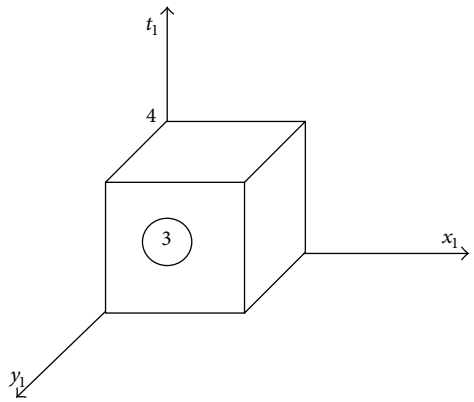

Figure 4: The status while activity 3 is put.

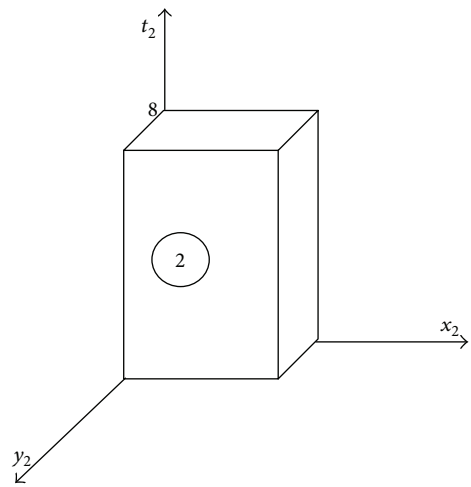

Figure 5: The status while activity 2 is put.

(3) At time 0, activity 4 will be put in order. After rotation, it lays on the $(2, 0, 0)$; see Figure 6.

(4) There is still available space at time 0, but it cannot meet the activity needs. We should find another space. According to our rule, we search for the space from time 4. At time 4, there are some activities to be arranged $\{5, 6, 7\}$, which include unscheduled activities at time 0. According to the area and duration, the activity sequence of place 1 is $\{5, 7\}$. We put activity 5 on the $(0, 0, 4)$; see Figure 7.

(5) All activities that start from time 4 are finished. We select time 6 as the start searching time based on the arranged activities. At time 6, $\{8, 9\}$ can be put. But $\{8, 9\}$ cannot match the space, so we continue to search for the next lowest time. According to the arranged activities, the next earliest time is time 7. Check the network diagram; $\{14, 15\}$ can be joined in, including $\{8, 9\}$. On place 1, we arrange activity 15 first based on our rule; see Figure 8.

(6) There is not enough space to put activity 9 at time 7. We will consider it later. On the place 2, $\{8, 14\}$ can be arranged. Activity 8 should be put first, but it cannot match the placement. So, we lay the activity 14 on $(2, 0, 7)$; see Figure 9.

(7) After we arrange all the activities at time 7, the next lowest time will be found to start the placement. Steps will continue until all the activities are arranged.

Improvement Process. The sequence of the activities is very important to complete the duration. So, we use ILS to disturb current searching point, which can cause random selection in the local optimum. The ILS includes three steps. The first step is producing the initial solution; the second step is searching in the neighbor to get a local optimum; and the third is a loop which has three parts. The first part is disturbing the local optimum to get a new sequence. The second part is searching a solution based on the new sequence. The third

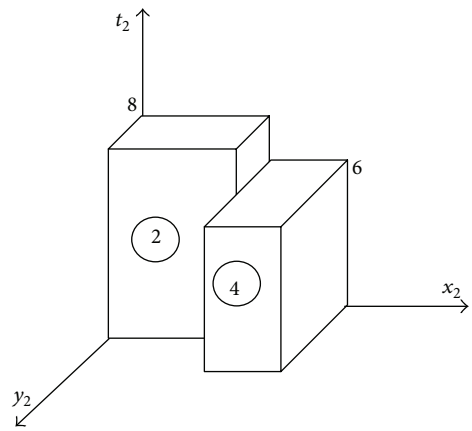

Figure 6: The status while activity 4 is put.

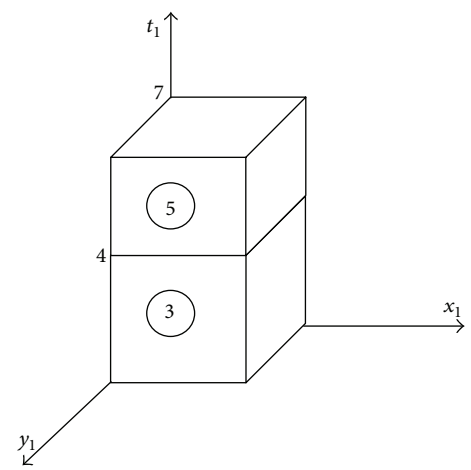

Figure 7: The status while activity 5 is put.

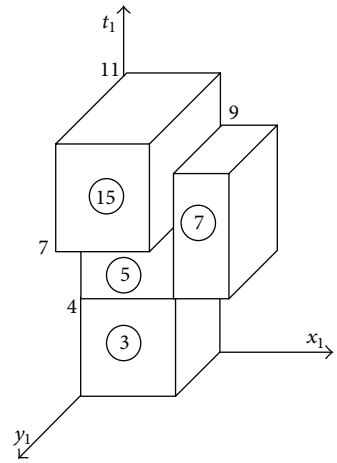

Figure 8: The status while activity 15 is put.

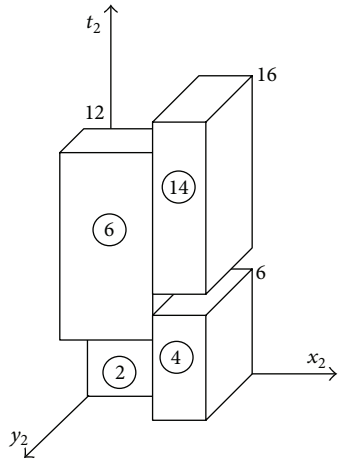

Figure 9: The status while activity 14 is put.

part is compared with previous solution to decide which is better until the termination condition is met.

As in our example, we swap two activities that start at the same time during the local search. The others are arranged according to our normal rule. The perturbation of ILS in our example is that any two activities are swapped at any time in the local optimum. The activities that have been swapped will not be considered. The others are scheduled according to our normal rule.

The initial solution of this example is 27. After ILS, the optimal solution is 21, producing about 22% improvement. For the set of instances, the average improvement is 21%.

5. Conclusion

For the proposed place scheduling problem, we have developed a hybrid heuristic method to generate the initial solution. In order to produce local optimal solution, we have designed ILS method to create improvement for the initial solution. In the improvement, the random disturbance has an impact on the result. So, the improvement based on randomness will be the future research.

Conflict of Interests

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, and we also declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by NSFC under Grant no. 2012-62000-4103036, NSFS under Grant no. ZR2012FM006, and SDPW under Grant no. IMZQWH010016. The authors thank the 3 reviewers and handling editor for their meticulous reading of the paper and their constructive comments which greatly improved the paper.

References

[1] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics*, vol. 45, no. 7, pp. 733–750, 1998.

[2] J. E. Kelley, "The critical-path method: resources planning and scheduling," *Industrial Scheduling*, no. 13, pp. 347–365, 1963.

[3] R. Alvar-Valdes and J. M. Tamarit, "Heuristic algorithms for resource-constrained project scheduling: a review and empirical analysis," *Advances in Project Scheduling*, no. 5, pp. 113–134, 1989.

[4] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," *Annals of Operations Research*, vol. 102, no. 1–4, pp. 83–109, 2001.

[5] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: theory and computation," *European Journal of Operational Research*, vol. 90, no. 2, pp. 320–333, 1996.

[6] R. Baker, *Introduction to Scheduling and Sequencing*, John Wiley & Sons, New York, NY, USA, 1974.

[7] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco, "An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation," *Management Science*, vol. 44, no. 5, pp. 714–729, 1998.

[8] E. K. Burke, G. Kendall, and G. Whitwell, "A new placement heuristic for the orthogonal stock-cutting problem," *Operations Research*, vol. 52, no. 4, pp. 655–671, 2004.

[9] J. Egeblad and D. Pisinger, "Heuristic approaches for the two- and three-dimensional knapsack packing problem," *Computers and Operations Research*, vol. 36, no. 4, pp. 1026–1049, 2009.

[10] S. Martello, D. Pisinger, and D. Vigo, "Three-dimensional bin packing problem," *Operations Research*, vol. 48, no. 2, pp. 256–267, 2000.

[11] G. Ulusoy and L. Özdamar, "A heuristic scheduling algorithm for improving the duration and net present value of a project," *International Journal of Operations and Production Management*, vol. 15, no. 1, pp. 89–98, 1995.

[12] E. W. Davis and J. H. Patterson, "A comparison of heuristic and optimum solutions in resource-constrained project scheduling," *Management Science*, vol. 21, no. 8, pp. 944–955, 1975.

[13] T. G. Crainic, G. Perboli, and R. Tadei, "Extreme point-based heuristics for three-dimensional bin packing," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 368–384, 2008.

[14] M. W. P. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations Research*, vol. 4, no. 1, pp. 285–305, 1985.

[15] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *European Journal of Operational Research*, vol. 113, no. 2, pp. 469–499, 1999.

[16] A. Schaerf and A. Meisels, "Solving employee timetabling problems by generalized local search," in *Advances in Artificial Intelligence*, vol. 1792 of *Lecture Notes in Computer Science*, pp. 380–389, Springer, New York, NY, USA, 2000.

[17] A. Marco, D. O. Montes, C. Carlos, and N. Ferrante, "Local search," in *Handbook of Memetic Algorithms*, vol. 24, pp. 21–25, 2012.

[18] G. F. Lucio, M. J. Reed, and I. D. Henning, "Guided local search as a network planning algorithm that incorporates uncertain traffic demands," *Computer Networks*, vol. 51, no. 11, pp. 3172–3196, 2007.

[19] R. Cabido, A. S. Montemayor, J. J. Pantrigo, and B. R. Payne, "Multiscale and local search methods for real time region tracking with particle filters: local search driven by adaptive scale estimation on GPUs," *Machine Vision and Applications*, vol. 21, no. 1, pp. 43–58, 2009.

[20] B. Funke, T. Grunert, and S. Irnich, "Local search for vehicle routing and scheduling problems: review and conceptual integration," *Journal of Heuristics*, vol. 11, no. 4, pp. 267–306, 2005.

[21] F. Tricoire, "Multi-directional local search," *Computers and Operations Research*, vol. 39, no. 12, pp. 3089–3101, 2012.

[22] F. Ricca and B. Simeone, "Local search algorithms for political districting," *European Journal of Operational Research*, vol. 189, no. 3, pp. 1409–1426, 2008.

[23] P. Mills, E. Tsang, and J. Ford, "Applying an extended guided local search to the quadratic assignment problem," *Annals of Operations Research*, vol. 118, no. 1–4, pp. 121–135, 2003.

[24] R. Kolisch and A. Sprecher, "PSPLIB—a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.