ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 1-5
doi: 10.18178/wcse.2019.06.001

A new Neighborhood Structure and its fast Evaluation Strategy in using Iterated Local Search to Solve Single Machine Scheduling

Hongyun Xu , Quan OuYang[+]

School of Mathematics & Computer Science, JiangHan University, Wuhan, China

Abstract. In this paper, a new neighborhood structure called Block Exchange and its fast evaluation strategy are proposed when using the Iterated Local Search (ILS) algorithm to solve the single machine scheduling problem without setup times. Applying the proposed ILS algorithm which adopts the Block Exchange neighborhood and fast evaluation strategy to solve 125 public benchmark instances, we achieve highly competitive results compared with the other two ILS algorithms adopt two neighborhoods widely used in the previous literature for single machine scheduling, respectively. Specifically, ILS with Block Exchange neighborhood and fast evaluation strategy finds the best-known results for all the 125 instances, while the other two ILS algorithms can not obtain the best-known results for 97 and 92 instances. The experimental results show that the proposed Block Exchange neighborhood and its fast evaluation strategy are highly effective and efficient.

Keywords: single machine scheduling, neighborhood structure, Iterated Local Search, Block Exchange.

1. Introduction

The single machine scheduling problem without setup times studied in this paper can be described as follows: Suppose N is the total number of jobs to be processed in one machine. Each job j has a processing time p_j, a weight w_j, and a due date d_j. The start time of each job is the completion time of the previous job in the job sequence. Using $\pi = \{\pi_1, \pi_2, \pi_3, ..., \pi_N\}\ (\pi_i \neq \pi_j, i \neq j)$ to represent a processing sequence, where π_j denotes the j-th job in the entire processing sequence and its completion time is denoted by C_j, $\Delta f(\pi_j)$. The tardiness of the π_j is T_j, $T_j = \max\{C_j - d_{\pi_j}, 0\}$. Then the total tardiness of π is denoted by T, $T = \sum_{i=1}^{N} w_j T_j$. So, the single machine scheduling problem without setup times can be denoted as $1 || \sum w_j T_j$, a special case of $1 | s_{ij} | w_j T_j$, has been shown to be strongly NP-hard [1],[2]. The $1 || \sum w_j T_j$ problem is to find a job sequence that minimizes T.

There are several heuristic and metaheuristic algorithms, for instance, particle swarm optimization algorithm, tabu search algorithm, and variable neighborhood search, developed to solve the $1 || \sum w_j T_j$ problem in [3]-[5].

2. Iterated Local Search algorithm

Iterated Local Search (ILS) is a simple but widely used heuristic algorithm. It uses the local search technique and the perturbation mechanism to search the solution space. The local search technique makes the current solution could be converged to the local extreme. The perturbation mechanism is responsible for jumping out of the local optimal to reach a new search space. The ILS algorithm consists of four parts: [1] Generate Initial Solution: The initial solution is usually composed of some constructive heuristic algorithm or randomly generated; [2] Local Search; [3] Perturbation: the method to jump out of the local optimal; [4]

[+] Corresponding author. Tel.: +86 18086623297.
E-mail address: oyq8888@163.com (Quan OuYang).

Acceptance Criterion: the algorithm stops when the certain conditions met. The basic framework of the iterated local search algorithm is shown as follows:

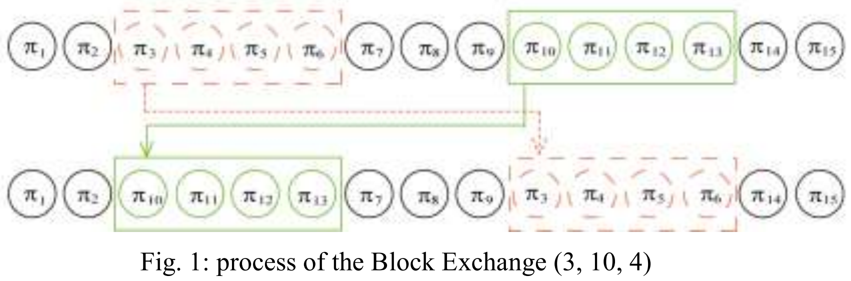
```
Step 1: s ←generate initial solution
Step 2: s′← local search (s)
Step 3: repeat
Step 4:   s* ← perturbation (s′)
Step 5:   s*′← local search (s*)
Step 6:   s′← acceptance criterion (s′, s*′)
Step 7: until stop condition met
```

In our ILS algorithm, the initial solution is generated randomly. At the end of each round of local search, if s*′ is better than s′, we replace the s′ with s*′. The proposed ILS algorithm stops when the time limit is reached. For a local search algorithm, the definition of the neighborhood plays an important role. In our ILS algorithm, we proposed a new neighborhood called Block Exchange, which has never been considered for the single machine scheduling to our best knowledge. At the same time, in order to evaluate the neighborhood solutions in an efficient way, we employ a fast evaluation strategy. The perturbation process is to randomly execute the Block Exchange for the current job sequence. In the following, we present the Block Exchange neighborhood and the fast evaluation strategy of the ILS algorithm in detail.

3. Block Exchange neighborhood structure

In this paper, we propose a new neighborhood structure, called Block Exchange, which is composed of all the feasible neighborhood solutions after Block Exchanges in solving $1||\sum w_j T_j$ problem. In author's previous study [6], a new neighbourhood structure called Block Move was proposed to solve single machine scheduling problem with setup times. Block Move and Block Exchange have some similarities and differences. For example, they both are operating on the Blocks, and Block Move is to move one Block to the other position, while Block Exchange is to exchange two Blocks' positions.

The Block Exchange is defined as follows: in the current job sequence $\pi = \{\pi_1, \pi_2, \pi_3, ..., \pi_N\}$, exchange two l consecutive jobs with the start positions of m and n, respectively (the l consecutive jobs are called block, Block). The range of m, n, and l are $m<n$, $1 \le m \le N-2l+1$, $m+l \le n \le N-l+1$, $1 \le l \le N/3$, respectively. The Block Exchange can be simply denoted as Block Exchange (*m, n, l*). For example, the Block Exchange (3, 10, 4) is performed in a job sequence of 15 jobs, the process of which is shown in Fig 1.



π1 π2 π3 π4 π5 π6 π7 π8 π9 π10 π11 π12 π13 π14 π15
π1 π2 π10 π11 π12 π13 π7 π8 π9 π3 π4 π5 π6 π14 π15

Fig. 1: process of the Block Exchange (3, 10, 4)

In the previous papers for the single machine scheduling, the other neighbourhoods, swap and 2-edge exchange, were employed in [7], which are special cases of our Block Exchange neighborhood where l=1 and 2, respectively.

4. Fast Evaluation Strategy for the Block Exchange neighborhood

The evaluation of the neighborhood solution needs the biggest amount of computation time in the local search algorithm. The efficient evaluation strategy can not only improve the efficiency of the algorithm, but also improve the probability of finding the better solution. In this paper, based on the strategy of the Block Move neighborhood structure and the fast incremental evaluation technique proposed by Xu et al. [6] in solving the single machine scheduling with sequence-dependent setup times, a new Block Exchange neighborhood structure and its fast evaluation strategy are proposed to solve the single-machine scheduling problem without setup times. The main idea is that for the job sequence π, the new job sequence π' is obtained after carrying out the Block Exchange. There is increment Δ between the jobs affected in job sequence π and π'.

Calculated the increment $\Delta f(\pi_j)$ of the jobs π_j affected by Block Exchange and obtained the objective function value $\Delta f(\pi)$ of the new job sequence π' by using the sum of $\Delta f(\pi_j)$. In this way, the computational complexity can be greatly reduced and achieving a fast evaluation strategy for the Block Exchange neighborhood solutions.

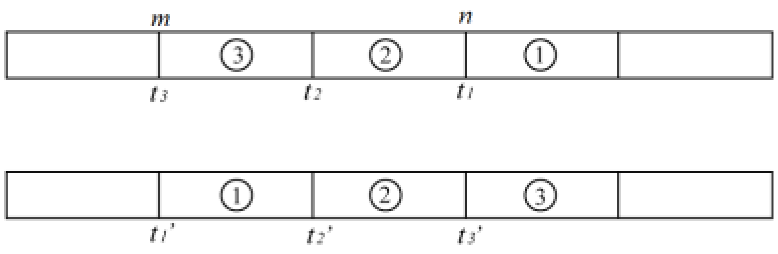


Fig. 2: illustration of the fast evaluation strategy for Block Exchange (m, n, l)

Taking the Block Exchange (m, n, l) as an example to illustrate the fast evaluation strategy proposed in this paper. As shown in Fig. 2, after the Block Exchange (m, n, l), the affected jobs are included in the blocks ①, ②, and ③. t1 represents the start time of the Block ①, the first position and the length of which are n and l; t2 represents the start time of the Block ②, the first position and the length of which are m+l and n-m-l; t3 represents the start time of the Block ③, the first position and the length of which are m and l.

t1', t2', and t3' represent the start times of the Blocks ①, ②, and ③ after executing Block Exchange (m, n, l), respectively. Δ1, Δ2, and Δ3 denote the time increments of Blocks ①, ②, and ③ after the Block Exchange (m, n, l), respectively, and the formula of which are as follows:

$\Delta 1 = t1' - t1;\ \ \Delta 2 = t2' - t2;\ \ \Delta 3 = t3' - t3$

The detailed calculation procedures of Δ1, Δ2, and Δ3 are as follows:

$$\begin{aligned} t1 &= C_{\pi(n-1)} \\ t2 &= C_{\pi(m+l-1)} \\ t3 &= C_{\pi(m-1)} \end{aligned} \tag{1}$$

$$\begin{aligned} t1' &= C_{\pi(m-1)} \\ t2' &= \Delta 1 + C_{\pi(n+l-1)} \\ t3' &= \Delta 2 + C_{\pi(n-1)} \end{aligned} \tag{2}$$

$$\begin{aligned} \Delta 1 &= t1' - t1 = C_{\pi(m-1)} - C_{\pi(n-1)} \\ \Delta 2 &= t2' - t2 = \Delta 1 + C_{\pi(n+l-1)} - C_{\pi(m+l-1)} \\ \Delta 3 &= t3' - t3 = \Delta 2 + C_{\pi(n-1)} - C_{\pi(m-1)} \end{aligned} \tag{3}$$

(1) As shown in Fig. 2, after the Block Exchange (m, n, l) on the job sequence π, the value of time increments Δ of jobs π_j varies according to the location of π_j:

When the job π_j is located in the Block ①, $j \in [n; n + l - 1], \Delta = \Delta 1$;

When the job π_j is in the Block ②, $j \in [m + l; n - 1], \Delta = \Delta 2$;

When the job π_j is located in the Block ③, $j \in [m; m + l - 1], \Delta = \Delta 3$.

(2) As shown in Fig. 2, the affected job π_j after the Block Exchange (*m, n, l*) on job sequence π, $j = \{m,...,n+l-1\}$. For the job π_j, $\Delta f(\pi_j)$ is considered in the following cases:

When the completion times of the job π_j before and after the Block Exchange (*m, n, l*) are both less than the due date, i.e., $C_{\pi_j} \le d_{\pi_j}$ and $C_{\pi_j} + \Delta \le d_{\pi_j}$ .The weighted tardiness of job π_j before and after the Block Exchange operation both are zero, i.e., $\Delta f(\pi_j) = 0$.

When the completion time of the job π_j before and after the Block Exchange (*m, n, l*) both are more than the due time, i.e., $C_{\pi_j} > d_{\pi_j}$ and $C_{\pi_j} + \Delta > d_{\pi_j}$, then $\Delta f(\pi_j) = w_{\pi_j} \Delta$.

When the completion time of the job π_j before the Block Exchange (*m, n, l*) is less than the due date, i.e., $C_{\pi_j} \le d_{\pi_j}$, and the completion time after the Block Exchange (*m, n, l*) is more than the due date, i.e., $C_{\pi_j} + \Delta > d_{\pi_j}$, then $\Delta f(\pi_j) = w_{\pi_j}(C_{\pi_j} + \Delta - d_{\pi_j})$.

When the completion time of the job π_j before the Block Exchange (*m, n, l*) is more than the due date, i.e., $C_{\pi_j} > d_{\pi_j}$, the completion time after the Block Exchange (*m, n, l*) is less than the due date, i.e., $C_{\pi_j} + \Delta \le d_{\pi_j}$, then $\Delta f(\pi_j) = -w_{\pi_j}(C_{\pi_j} - d_{\pi_j})$.

(3) Find the value Δf (π_j), $j = \{m,...,n+l-1\}$, and calculate $\Delta f(\pi) = \sum_{j=m}^{n+l-1} \Delta f(\pi_j)$. And the objective function value f (π') of the job sequence π' after the Block Exchange (*m, n, l*) is obtained, i.e., $f(\pi') = f(\pi) + \Delta f(\pi)$, in which $f(\pi)$ is the objective function value of the job sequence π.

5. Computational results

5.1. Experiment scheme

In order to show the effectiveness of the newly proposed Block Exchange neighborhood and the fast evaluation strategy, we compare them with two previously used neighborhoods, swap and 2-edge exchange. Thus, three ILS algorithms, denoted by ILS_BE, swap, and 2-edge, are designed, which adopt the Block Exchange neighborhood and fast evaluation strategy, swap neighbourhood, and 2-edge exchange neighborhood, respectively.

The benchmark data of the $1||\sum w_j T_j$ problem widely used in the literature consisted of 125 problem instances, each with 40 jobs [8]. The three ILS algorithms, ILS_BE, swap, and 2-edge, both are programmed in C++ and compiled on a PC using Linux with 2.6 GHz CPU and 64.0 Gb RAM. For the three ILS algorithms, we solved each problem instance independently for 100 times using different random seeds, subject to a time limit of 50 CPU seconds.

5.2. Comparisons with other two neighborhoods

We summarize the comparisons of three ILS algorithms, swap, 2-edge, and ILS_BE, with the best-known results for the 125 instances in Table 1, in which the numbers of better, equal, and worse solutions compared with the best-known results are presented.

Table 1.summary of comparing the results of three ILS algorithms with the best-known results

	swap	2-edge	ILS_BE
number of improved solutions	0	0	0
number of equal solutions	33	28	125
number of inferior solutions	92	97	0
number of all solutions	125	125	125

In Table 1, we observe that ILS_BE algorithm is quite competitive in terms of this criterion. Comparing the results of three ILS algorithms, swap, 2-edge, and ILS_BE, with the best-known results, the numbers of equal solutions were 33, 28, and 125, respectively. And the numbers of worse solutions were 92, 97, and 0, respectively. From Table 1, we conclude that the ILS_BE algorithm outperforms the other two ILS algorithms in terms of solution quality. These results show the efficacy of the proposed Block Exchange neighborhood and its fast evaluation strategy.

6. Conclusion

In this paper, a new neighborhood structure Block Exchange and its fast evaluation strategy are proposed to minimize the total weighted tardiness when using the ILS algorithm to solve the $1||\sum w_jT_j$ problem. The Block Exchange neighborhood expands the search space, while the fast evaluation strategy improves the search efficiency to reduce the calculation time. From the comparisons of experiments, it is proved that the Block Exchange neighborhood and the fast evaluation strategy can improve the efficiency and find the better solutions in a certain extent. Although this paper mainly discusses the single machine scheduling, it provides the new ideas to the complex scheduling problem and other combinatorial optimization problem, which has certain research significance.

7. Acknowledgments

The research was supported by the Doctoral Scientific Research Foundation of JiangHan University, WuHan (Grant no. 1008-06280004).

8. References

[1] Lawler E L. A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness. Annals of discrete Mathematics, 1977, 1:331–342

[2] Lenstra J K, Rinnooy Kan A, Brucker P. Complexity of machine scheduling problems. Annals of discrete mathematics, 1977, 1:343–362

[3] Tasgetiren MF, LiangYC, SevkliM, Gencyilmaz G. Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. Int J Prod Res 2006; 44(22): 4737–54.

[4] Ü Bilge, Kurtulan M, Kıraç F. A tabu search algorithm for the single machine total weighted tardiness problem. Eur J Oper Res2007; 176:1423–35.

[5] Wang X, Tang L. A population-based variable neighbourhood search for the single machine total weighted tardiness problem. Comput Oper Res 2009; 36: 2105–10.

[6] Xu H, Lü Z, Cheng T C. Iterated Local Search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness[J]. Journal of Scheduling, 2014, 17(3):271-287.

[7] Kirlik, G., & Oguz, C. (2012). A variable neighbourhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. Computers & Operations Research, 39(7), 1506–1520.

[8] Potts C N, Wassenhove L N V. A Branch and Bound Algorithm for the Total Weighted Tardiness Problem[J]. Operations Research, 1985, 33(2):363-377.