Breakout local search for the traveling salesman problem with job-times

Yuji Zou[a], Jin-Kao Hao[a,*], Qinghua Wu[b]

[a]*LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France*
[b]*School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China*

Expert Systems with Applications 225: 120020, September 2023
https://doi.org/10.1016/j.eswa.2023.120020

Abstract

The traveling salesman problem with job-times combines two classic NP-hard combinatorial optimization problems: the traveling salesman problem and the scheduling problem. In this problem, a traveler visits sequentially n locations with given travel-times between locations and assigns, to each visited location, one of n jobs with location-dependent job-times. When a job is assigned to a specific location, the job starts to run at that location for its given duration. The goal of the problem is to find a job assignment to minimize the maximum completion time of the n jobs. This work presents an effective heuristic algorithm for the problem based on the breakout local search method. The algorithm employs a local search procedure to find high-quality local optimal solutions and a dedicated perturbation procedure to escape local optimum traps. The local search procedure is based on the *2-opt* and *j-swap* moves adapted to the problem, while the perturbation procedure combines both informed and random *2-opt* and *j-swap* operations. To speed up the search, we introduce a dedicated strategy to identify promising neighboring solutions. We evaluate the algorithm on the 310 benchmark instances in the literature. Computational results show that the proposed algorithm outperforms the previous methods, by reporting improved best results (new upper bounds) for 291 instances and equal best results for 16 other instances. The main search components of the algorithm are investigated to shed light on their contributions to the performance of the algorithm.

Keywords: Heuristics; routing-assignment problems; combinatorial optimization; local search.

*Corresponding author

Email addresses: yujizou6@gmail.com (Yuji Zou), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), qinghuawu1005@gmail.com (Qinghua Wu)

1. Introduction

The Traveling Salesman Problem with Job-times (TSPJ) (Mosayebi et al., 2021) combines the traveling salesman problem (TSP) and the scheduling problem. In this problem, a traveler starts from the depot 0, visits n given locations exactly once, and returns to the depot. For each visited location l (except the depot), one job j among n given jobs with location-dependent processing times (job-times) jt_{lj} is assigned and the job starts to run during that time while the traveler moves to the next location. For a given job j assigned to location l, its completion time equals the arriving time at location l plus the processing time jt_{lj}. For a Hamiltonian tour with the n job-location assignment, the completion time of the n jobs is the maximum completion time among the n jobs. The goal of the TSPJ is then to find the Hamiltonian tour starting from and ending at the depot 0 and including the n job-location assignments such that the maximum completion time among the n jobs is minimized. The TSPJ belongs thus to the class of min-max problems and is at least as challenging as its composing location and assignment problems. A mathematical formulation of the problem was introduced in Mosayebi et al. (2021) based on a conventional integer programming formulation for the TSP (see Appendix A).

A TSPJ instance is defined with two input data: 1) a symmetric edge-weighted complete graph $G = (V, E)$ where a node in the vertex set V represents a location (node 0 being the depot) and the weight of an edge in the edge set E represents the travel time between two locations, and 2) a location-job matrix indicating the processing time of each job (job-time) at each location. Given a TSPJ instance with n jobs and n locations, a candidate solution can be represented by two n-dimensional permutation vectors. Let $\pi_1 : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ be a permutation representing a TSP tour starting from and ending at the depot 0 and let Π_1 denote the set of all these permutations. Let $\pi_2 : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ be a job-location assignment and let Π_2 denote the set of all these assignments. Then a candidate solution can be represented as $S = (\pi_1, \pi_2)$, $\pi_1 \in \Pi_1$, $\pi_2 \in \Pi_2$. The search space Ω is given by

$$\Omega = \{(\pi_1, \pi_2) : \pi_1 \in \Pi_1, \pi_2 \in \Pi_2\} \tag{1}$$

Let $S \in \Omega$ be a candidate solution, its objective value $f(S)$ is defined as follows (Mosayebi et al., 2021).

$$f(S) = \max\{\max\{TS_l + \sum_{j=1}^{n} Z_{lj} jt_{lj}\}, TS_l + X_{l0} D_{l0}\} \quad \forall l = 1, \dots, n \tag{2}$$

where TS_l is the start time of the job assigned to location l, the binary variable $Z_{lj} = 1$ if job j is assigned to location l and $Z_{lj} = 0$ otherwise, jt_{lj} is the job time of job j at location l, the binary variable X_{l0} equals 1 or 0 according to whether the TSP tour goes from location l to the depot 0 and D_{l0} is the travel time from location l to the depot. From Eq. (2), one observes that $f(S)$ equals either 1) the largest job completion time (makespan) (the inner max), or 2) the travel time starting and ending at the depot ($TS_l + X_{l0}D_{l0}$). Case 2) corresponds to the situation when the travel time of the TSP tour is higher than the largest job completion time. The TSPJ aims to find a solution that minimizes the objective function f.

As an illustrative example, Table 1 shows the input data of a TSPJ instance where Nodes denote the depot 0 and the locations. The left part gives the travel times between the locations and the right part indicates the job-time of each job at each location. Fig. 1 shows two candidate solutions (a) and (b). For each TSP tour (sequence of black cycles), the arriving time (i.e., the processing start time) at each location node is shown next to the node and the job-time of the assigned job at the node is indicated in red.

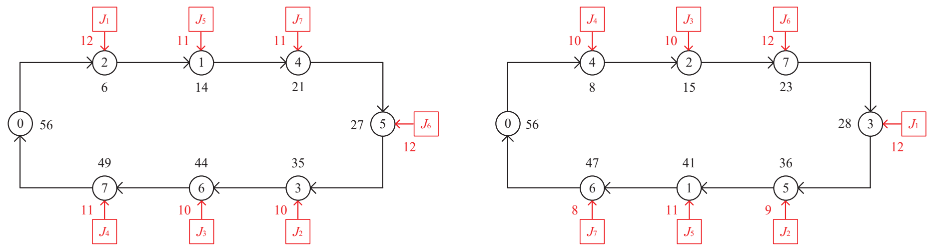
Table 1: A TSPJ instance where the left part shows the travel time from one node to another node and the right part shows the location-job matrix indicating the processing time of each job (job-time) at each location.

Nodes	Nodes (travel-time between nodes)								Jobs (location dependent job-time jt_{lj})						
	0	1	2	3	4	5	6	7	J_1	J_2	J_3	J_4	J_5	J_6	J_7
0	0	4	6	5	8	9	9	7	-	-	-	-	-	-	-
1	4	0	8	6	7	5	6	6	9	10	8	10	11	8	9
2	6	8	0	5	7	6	10	8	12	9	10	10	9	13	11
3	5	6	5	0	3	8	9	5	12	10	9	11	11	8	14
4	8	7	7	3	0	6	5	9	10	9	12	10	11	8	11
5	9	5	6	8	6	0	8	8	9	9	10	11	11	12	8
6	9	6	10	9	5	8	0	5	10	10	10	11	11	9	8
7	7	6	8	5	9	8	5	0	9	10	10	11	11	12	13

Table 2: The processing start time and completion time of each job assigned to its location node for the two candidate solutions in Fig. 1.

Nodes	solution of Fig. 1(a)		solution of Fig. 1(b)	
	start time (arriving time TS_l)	completion time	start time (arriving time TS_l)	completion time
0	56	-	**56**	-
1	14	25 (J_5)	41	52 (J_5)
2	6	18 (J_1)	15	25 (J_3)
3	35	45 (J_2)	28	40 (J_1)
4	21	32 (J_7)	8	18 (J_4)
5	27	39 (J_6)	36	45 (J_2)
6	44	54 (J_3)	47	55 (J_7)
7	49	**60 (J_4)**	23	35 (J_6)

Table 2 shows the detailed information of the start time and completion time of each job assigned to the indicated node. The value of 56 for node 0 is the travel time starting and ending at the depot. The completion time of job j assigned to node l is the arriving time at node l (TS_l in Table 2) plus the job-time (jt_{lj} in Table 1). According to Eq. (2), the objective value of a solution is determined by the maximum job completion time or the time required to return to the depot. So the objective value of the solution of Fig. 1(a) equals the job completion time of job J_4 at node 7 ($49 + 11 = 60$). For solution of Fig. 1(b), since the travel time starting and ending at the depot ($47 + 9 = 56$ where 9 is the travel time from the last node 6 to the depot, see Table 1) is greater than



(a) Node 7 in this solution has a completion time of 60. So its objective is 60.

(b) Depot in this solution has a return time of 56. So its objective is 56.

Figure 1: Two candidate solutions for the TSPJ instance of Table 1 are shown. The solution (a) has an objective value of 60, which is the maximum completion time of job J_4 assigned to location node 7. The solution (b) has an objective value of 56 because the maximum completion time among the jobs is 55 with job J_7 assigned to location node 6, which is however smaller than the travel time starting and ending at the depot 0.

the largest completion time ($47+8=55$ of J_7 at location node 6), the objective value of solution of Fig. 1(b) is 56.

It is easy to see that the NP-hard TSP is a special case of the TSPJ when the job-times equal 0. As a result, the TSPJ is at least as difficult as the TSP and solving the problem is computationally challenging.

Along with the introduction of the TSPJ, Mosayebi et al. (2021) introduced four heuristic algorithms as well as four sets of 310 benchmark instances. They performed comprehensive assessments of these heuristic algorithms on these benchmark instances. They also used the CPLEX MIP solver to solve the integer programming model shown in Appendix A. We review these heuristic algorithms and related works in Section 2.

Given the relevance of the TSPJ and its computational challenge, it is worth developing effective methods able to provide satisfactory solutions. Currently, such methods are still scarce in the literature, this work aims to fill the gap by presenting a new heuristic algorithm for the TSPJ. Our main contributions are summarized as follows.

We propose an effective algorithm based on the breakout local search method (BLS) (Benlic & Hao, 2013a,b,c) that features two key complementary search components. First, BLS uses a dedicated tabu search procedure to explore candidate solutions based on two specific neighborhoods combined with a neighborhood reduction strategy to make the neighborhood examination more focused. Second, BLS employes a combined perturbation strategy that adaptively applies a random perturbation and a frequency-based perturbation to help the algorithm to escape local optimum traps.

To show the effectiveness of the algorithm, we carry out extensive computational experiments on the four sets of 310 benchmark instances of (Mosayebi et al., 2021). We show that our BLS algorithm significantly outperforms the existing algorithms in the literature. Specifically, we report improved best-known results (new upper bounds) for 291 instances and equal best-known results for 16 other instances. We present additional experiments to shed light on the influences of the main search components over the performance of the algorithm.

Finally, we will make the code of our algorithm publicly available, which can be used by researchers and practitioners working on the TSPJ and related problems.

In Section 2, we review related works. In Section 3, we present the proposed algorithm. In Section 4, we provide experimental results and comparisons with existing methods. In Section 5, we analyze the main components of the algorithm to shed light on their roles. Conclusions are provided in Section 6.

2. Related works

2.1. Existing algorithms for the TSPJ

In Mosayebi et al. (2021), four heuristic procedures were presented for the TSPJ. These four algorithms are different combinations of the following basic heuristics: Nearest Neighbor for TSPJ (TSPJ-NN), 2-*opt* for TSPJ (TSPJ-2-*opt*) and local search improvement (LSI).

The TSPJ-NN heuristic is adapted from the popular NNH-X heuristic for the TSP. NNH-X starts with *any* node and then chooses the nearest node among the unvisited nodes as the next node to be visited. After all the nodes have been chosen as the first node, the tour with the shortest length is selected as the final solution. TSPJ-NN extends NNH-X to fit the TSPJ and applies the so-called reverse assignment strategy to assign the jobs. Specifically, for each tour given by NNH-X, TSPJ-NN considers the location nodes one by one in the reserve order they are visited by starting from the last node and assigns to the node under consideration the job with the minimum job-time. After all the nodes have been chosen as the first node, the solution with the minimum job completion time is retained as the final solution.

The TSPJ-2-*opt* heuristic uses the popular TSP 2-*opt* heuristic to generate new tours and the reverse assignment to assign jobs to locations. It starts with the first node in the current tour and applies the 2-*opt* move to swap all combinations of node pairs in the tour. For each new tour from 2-*opt*, the jobs are assigned to the location nodes using the reverse assignment as in the TSPJ-NN heuristic. The 2-*opt* move is accepted as long as the resulting tour improves the current solution. The process is repeated until no further improvement can be achieved by swapping any pair of nodes in the tour.

The local search improvement heuristic LSI is composed of two main steps. Step 1 reassigns jobs to different location nodes while step 2 changes the sequence of location nodes. Let l_m be the node which results in the maximum completion time C_{max} (e.g., node 7 in Fig 1(a)). Step 1 repetitively reduces C_{max} by re-assigning to l_m another job with less job-time (e.g., J_2 or J_3 in Fig 1(a)) without increasing C_{max}, followed by re-assigning the remaining jobs with the reverse assignment strategy. Then the solution is further improved by swapping the job assigned to l_m with another job assigned to a location node behind l_m in the tour. When step 1 cannot decrease C_{max} anymore, step 2 is used to change the sequence of location nodes with the so-called "predecessor node swap" and "multi-node swap". The predecessor node swap exchanges the node l_m with the node immediately preceding l_m in the tour and this is repeated as long as the solution is improved. The multi-node swap, which follows, can be viewed as a modification of 2-*opt* and swaps node l_m with one of its predecessors in the tour while reversing the sequence of the location nodes between the two nodes in the tour. Like the predecessor node swap, the multi-node swap is accepted as long as it leads to an improvement of the solution. The LSI heuristic repeats step 1 and step 2 until no improvement is possible.

Based on these basic heuristics, Mosayebi et al. (2021) defined four TSPJ heuristic procedures.

- Procedure I first uses TSPJ-NN to obtain an initial solution and then uses TSPJ-2-*opt* to further improve the solution.
- Procedure II successively applies NNH-X to obtain an initial tour, 2-*opt* to improve the tour, reverse assignment to assign the jobs and local search improvement LSI to further raise the quality of the solution.
- Procedure III successively applies TSPJ-NN to obtain an initial solution, local search improvement LSI followed by TSPJ-2-*opt* to further improve the quality of the solution.
- Procedure IV first identifies the node with the largest distance from the depot as the last node in the tour and uses this node as the last node in the tour. It then runs TSPJ-NN to complete the tour and local search improvement LSI to further improve the solution.

As shown in (Mosayebi et al., 2021), these procedures have reached competitive results compared to the CPLEX MIP solver. Meanwhile, one notices that these procedures are mainly based on the principle of greedy or descent search. They have a limited capability to escape local optimum traps and can miss high-quality solutions. In this work, we investigate a stochastic local search approach (Hoos & Stützle, 2004) that is able to visit multiple local optima solutions to find the best possible solution.

2.2. Related works on iterated local search

Our proposed BLS algorithm is based on iterated local search (ILS) (Lourenço et al., 2003) and especially breakout local search (BLS), which enhances the ILS framework with an adaptive multi-perturbation strategy. In what follows, we provide a brief review of some representative works on these frameworks applied to several difficult problems, especially TSP-like problems.

Zhou et al. (2022) introduced a multi-neighborhood simulated annealing-based iterated local search approach for the colored traveling salesman problem. The algorithm uses the intra-route and inter-route moves to explore candidate

TSP tours and applies an enhanced edge assembly crossover to find nearby high-quality solutions around a local optimum. It relies on the the Metropolis condition and a solution reconstruction procedure to escape local optimum traps.

Nogueira et al. (2021) proposed an iterated local search with tabu search for the weighted vertex coloring problem. The algorithm iterates a multi-neighborhood local search procedure to find local optimal solutions and a perturbation procedure to escape from local optima. The perturbation randomly selects p vertices and assigns to each chosen vertex a random color, where p is adjusted by an adaptive mechanism such that p is increased if the search is considered to be stagnating.

Archetti et al. (2018) proposed an iterated local search algorithm to solve the traveling salesman problem with release dates and completion time minimization. The algorithm relies on a local search to find local optimal solutions and a a destroy-and-repair procedure to escape from local optimum traps. The destroy-and-repair procedure removes α customers from the route and reassign them in the route, where α is adjusted in an adaptive manner during the search such that α varies between a lower bound α_{min} and an upper bound α_{max}.

Benlic & Hao (2013c) presented the first breakout local search to solve the quadratic assignment problem. To go beyond local optima found by the descent search, the algorithm probabilistically applies three types of perturbations (directed, frequency-based and random perturbations) depending on the search state. Additionally, the number of moves applied by the chosen perturbation (called jump magnitude or perturbation length) is adjusted through an adaptive mechanism. Starting from a weak jump magnitude, it is gradually increased if the jump is not sufficient to escape the current local optimum trap.

Krari et al. (2018) introduced a breakout local search approach for solving the TSP. The algorithm performs a local search phase and a perturbation-based diversification phase. The local search procedure uses the2-opt based steepest descent. The perturbation is achieved by varying the type of moves (2-*opt*, insert, and swap) and the jump magnitude. The most fitting perturbation for each diversification period is adaptively selected by an adaptive mechanism similar to Benlic & Hao (2013c) according to the search state.

Ghandi & Masehian (2015) applied breakout local search to the assembly sequence planning problem. The algorithm first utilizes a simple hill-climbing local search to reach a local optimum. Then the perturbation is applied to escape the local optimum. Four types of move operators (flip, exchange, insertion, and inversion) are employed to perform the perturbation, and the selection of the applied perturbation operators is based on the consecutive non-improving local search phases.

2.3. Possible applications of the TSPJ

As discussed in Mosayebi et al. (2021), the TSPJ is a relevant model for a variety of practical scenarios including autonomous robotics (Bays & Wettergren, 2017), equipment maintenance (Rashidnejad et al., 2018), highly automated manufacturing (Das & Nagendra, 1997), agricultural harvesting (Basnet et al., 2006), and disaster recovery (Barbarosoğlu et al., 2002).

To provide a representative application example, we consider the Sequence-Dependent Robotic Assembly Line Balancing Problem of type 2 (SDRALBP-2) (Lahrichi et al., 2020). In this problem, there are a set of operations, a set of stations and a set of robots of different types with different abilities. There are three decision problems. One needs to assign the operations to the stations placed in a straight line and sequence the operations at the same station while satisfying precedence relations between the operations. Finally, one needs to assign a robot to process the operations at each station. The start time of an operation is sequence-dependent since the operations should be processed one by one, and there is a setup time between the operations. The operation processing time is dependent on the robot assigned to the station. The goal of the SDRALBP-2 is to minimize the maximum workload time among all the stations to achieve the balance purpose. Another relevant application is the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources, and learning effect (Zhang et al., 2021). In this problem, a set of jobs need to be processed with a set of parallel machines. The setup time is sequence and machine-dependent. And the job processing time is dependent on the machine. The purpose of this problem is to minimize the maximum completion time among all jobs. Both applications can be conveniently formulated with the TSPJ model.

Given the relevance of the TSPJ and the limited number of algorithms for solving the problem, we propose in this work an effective algorithm based on the breakout local search method, which advances the state of the art of solving this challenging problem. Indeed, the BLS method has been successfully applied to several difficult optimization problems including maximum clique (Benlic & Hao, 2013a), max-cut (Benlic & Hao, 2013b), quadratic assignment (Benlic & Hao, 2013c; Aksan et al., 2017), constrained Steiner tree problem (Fu & Hao, 2014), assembly sequence planning (Ghandi & Masehian, 2015), and TSP (Krari et al., 2018). As this work shows, BLS is also a highly competitive approach for solving the TSPJ.

3. Breakout local search for the TSPJ

This section is dedicated to the breakout local search algorithm for solving the TSPJ. A typical BLS algorithm iterates a dedicated local search procedure to find high-quality local optimal solutions and an adaptive perturbation procedure to escape local optimum traps. BLS enhances the popular iterated local search, which typically applies random perturbations, by employing an adaptive multi-perturbation strategy to ensure a suitable search diversification. This is achieved by dynamically determining the perturbation strength for different types of perturbation (e.g., random or informed perturbations). By iterating the local search and the adaptive perturbation, BLS favors the balance of search intensification and diversification and helps to better explore the search space.

3.1. The BLS procedure

Algorithm 1: Pseudo-code of BLS for the TSPJ

Input: Problem instance, time limit t_{max}, search depth ω, minimum perturbation length L_{min}, maximum perturbation length L_{max}.
Output: The best solution S_b found so far.

1 $L \leftarrow L_{min}$; `/* perturbation length */`
2 $NoImprove \leftarrow 0$; `/* counter of consecutive loops` f_{best} `is not improved */`
3 $S_{initial} \leftarrow$ TSPJ-NN() ; `/* generation of initial solution with TSPJ-NN */`
4 $S \leftarrow S_{initial}$; `/* current solution */`
5 $S_b \leftarrow S$; `/* best solution found so far */`
6 **while** t_{max} *is not reached* **do**
7     $S_l, S_c \leftarrow$ tabu search(S, ω, L_{min}) ; `/* Section 3.4 */`
8   **if** $f(S_l) < f(S_b)$ **then**
9         $S_b \leftarrow S_l$;
10     $S \leftarrow$ perturbation(S_c, L) ; `/* Section 3.5 */`
11   **if** $L < L_{max}$ **then**
12         $L \leftarrow L + 1$;
13 **return** S_b; `/* return the best solution found during the search */`

The proposed BLS algorithm for the TSPJ (see Algorithm 1) integrates two key complementary ingredients responsible for its effectiveness: a dedicated tabu search procedure exploring two neighborhoods enhanced with a neighborhood reduction technique and a combined perturbation strategy for search diversification. BLS starts from an initial solution built with the TSPJ-NN heuristic (line 3). Then it alternates iteratively between a tabu search phase and a dedicated perturbation phase (lines 7-10). The best solution S_b found so far is updated each time an improved local best solution S_l is discovered during the tabu search (lines 8-9). Meanwhile, the perturbation length is updated during the tabu search according to the search information. When the tabu search phase stops upon reaching its search depth ω, the search is considered to stagnate in a local optimum. In this case, the perturbation phase is triggered to modify the current solution S_c with the perturbation length L to help the algorithm to escape the current local optimum (line 10). Following this, the perturbation length is increased by 1 so long as it does not reach the limit L_{max}. The solution produced by the perturbation becomes the starting point of the next round of the tabu search.

3.2. Initial solution

BLS needs an initial solution to start its search and a good initial solution is helpful for the algorithm to discover high-quality solutions. To obtain a starting solution of reasonable quality, BLS adopts the TSPJ-NN heuristic presented in (Mosayebi et al., 2021) and reviewed in Section 2.1.

The time complexity of constructing a solution using NNH is $O(n^2)$. Assigning the jobs using reverse assignment requires $O(n)$ time. Computing the objective value takes $O(n^2)$ time. In TSPJ-NN, every node is used to be the

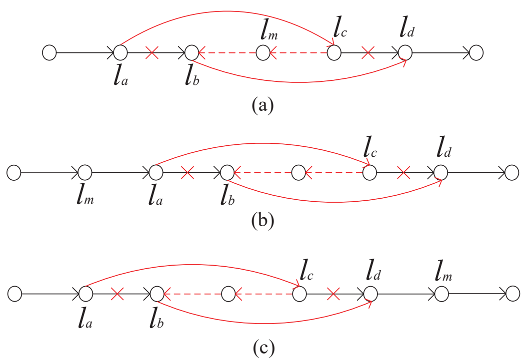


Figure 2: Three possible 2-*opt* moves according to the position of the node with the maximum completion time denoted by l_m. (a) l_m belongs to the reversed edge, (b) l_m is before the first node of 2-*opt*, (c) l_m is behind the second node of 2-*opt*.

first node of the tour. Thus, the time complexity of obtaining an initial solution is bounded by $O(n^3)$.

3.3. Neighborhoods

Tabu search examines candidate solutions by exploring two neighborhoods induced by the basic 2-*opt* and *j-swap* move operators.

3.3.1. 2-opt neighborhood

2-*opt* (Lin, 1965) is a well-known operator to generate neighboring solutions for the TSP. The 2-*opt* operator basically breaks the given tour by deleting two edges and reconnects the broken tour by adding two new edges (see Fig. 2 for an illustrative example). For the TSP, the objective variation between the current solution and a neighboring solution is related to the tour length difference between the two added edges and the two removed edges. The TSPJ has a quite different objective which depends not only on the arriving time at a node (i.e., job start time), but also the location-dependent job time. So the objective variance after a 2-*opt* move for the TSPJ is much more complex to calculate. The basic way to obtain the objective value after a 2-*opt* move is to calculate, for each location node of the tour, the arriving time, i.e., job start time, thus getting the assigned job completion time. However, this is time-consuming.

To limit the computational burden in terms of the objective evaluation of neighboring solutions, our BLS algorithm uses a neighborhood reduction method to eliminate unpromising neighboring solutions. Let l_m denote the node that has the maximum completion time in the current solution. As shown in Fig. 2(c), when l_m is visited behind the second node involved in the 2-*opt* move, if $\Delta = D_{ac} + D_{bd} - D_{ab} - D_{cd} > 0$, where D_{xy} is the travel time from node x to node y, then we know that such a move results in a longer tour, increases the job start time at node l_m, and raises the job completion time. Thus, it is no use to consider such moves. In our BLS algorithm, we ignore the neighboring solutions generated by these unpromising 2-*opt* moves. Our experiments show that l_m is often visited late in the sequence. So most neighboring solutions produced by the 2-*opt* move correspond to the situation shown in Fig. 2(c), that are not considered by BLS.

Formally, the reduced neighborhood $N_1(S)$ is defined as Eqs. (3) - (5), where S is the given solution, $N_1'(S)$ is the whole 2-*opt* neighborhood, and $N_{f1}(S)$ is the set of neighboring solutions generated by unpromising 2-*opt* moves discussed above, $e_1 = (l_a, l_b)$ and $e_2 = (l_c, l_d)$ are the two deleted edges with their nodes l_a, l_b, l_c, l_d, $S \oplus 2\text{-}opt(l_a, l_b, l_c, l_d)$ is the resulting neighboring solution. Let p represent the position of the location node in the tour such that l_a is the p_ath location node to be visited. We use Δ'' to denote the tour length variation resulting from the 2-*opt* move.

$$N_1(S) = N_1'(S) \setminus N_{f1}(S) \tag{3}$$

$$N_1'(S) = \{S' : S' = S \oplus 2\text{-}opt(l_a, l_b, l_c, l_d), (l_a, l_b) = e_1, (l_c, l_d) = e_2, e_1, e_2 \in E\} \tag{4}$$

$$N_{f1}(S) = \{S'' : S'' = S \oplus 2\text{-}opt(l_a, l_b, l_c, l_d), p_m \geq max\{p_a, p_b, p_c, p_d\}, \Delta'' > 0\} \tag{5}$$

Since 2-*opt* can delete and reconnect all possible pairs of edges, the whole 2-*opt* neighborhood $N_1'(S)$ has the size of $O(n^2)$. $N_{f1}(S)$ is the set of neighboring solutions omitted from the $N_1'(S)$ and its size is bounded by $O(n^2)$. In the worst case, none of the neighboring solutions is contained in $N_{f1}(S)$ (i.e., $N_{f1}(S) = \emptyset$), then the size of $N_1(S)$ is $O(n^2)$. For a given neighboring solution, its objective value can be calculated in $O(n)$. As a result, the time complexity of exploring the reduced neighborhood $N_1(S)$ is bounded by $O(n^3)$ in the worst case. In the best situation (when l_m is the last location node in the tour and $\Delta'' > 0$ for all neighboring solutions induced by 2-*opt* moves), all neighboring solutions contained in $N_1'(S)$ are also included in $N_{f1}(S)$, implying that $N_1(S) = \emptyset$. In this case, we only need to calculate the Δ'' value for each neighboring solution in $N_1'(S)$ to decide whether it can be omitted for further objective evaluation. Given that the calculation of Δ'' for a neighboring solution in $N_1'(S)$ is achieved in $O(1)$ and the size of $N_1'(S)$ is $O(n^2)$, the time complexity of exploring $N_1(S)$ is bounded by $O(n^2)$ in the best case.

3.3.2. j-swap neighborhood

The *j-swap* operator is inspired by the popular *swap* operator, which is extremely effective for permutation-based assignment problems like quadratic assignment (Benlic & Hao, 2013c, 2015; Taillard, 1991). Since the TSPJ includes a job assignment task, we naturally adopt *j-swap* for this problem.

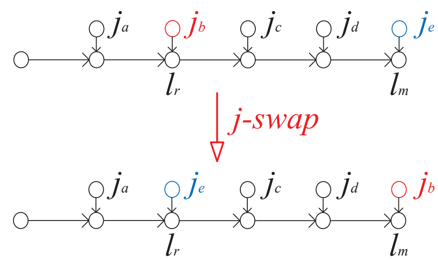


Figure 3: *j-swap*, exchange the job j_e assigned to the node having the maximum completion time l_m with another job j_b assigned to location node l_r.

The *j-swap* operator exchanges the assignments of two jobs (see Fig. 3 for an example). Thus only the job processing times of the two involved nodes are exchanged without changing the processing start time of each exchanged job. It is easy to observe that only changing the job assigned to the node with the maximum completion time (i.e. l_m) may decrease (improve) the objective value. So in our algorithm, we force *j-swap* to focus on the moves related to that job only. In other words, *j-swap* only changes the job at the location with the maximum completion time with another job.

Formally, the constrained *j-swap* neighborhood N_2 is defined as follows, where j_b and j_e are the jobs exchanged in *j-swap*, and m is the index of the location node with the maximum completion time l_m.

$$N_2(S) = \{S' : S' = S \oplus j\text{-}swap(j_b, j_e), jt_{mb} < jt_{me}\} \quad (6)$$

As shown in Fig. 3, one job involved in *j-swap* is selected to be the job assigned to l_m, the size of the *j-swap* neighborhood is bounded by $O(n)$. As the objective value of each *j-swap* neighboring solution can be calculated in $O(1)$, the time complexity of exploring the constrained *j-swap* neighborhood is bounded by $O(n)$, reducing significantly the time complexity $O(n^2)$ for exploring the whole neighborhood induced by the unconstrained *j-swap* move.

3.4. Examination of candidate solutions with tabu search

BLS uses tabu search (TS) (Glover & Laguna, 1997) to examine candidate solutions by exploring the *2-opt* and *j-swap* neighborhoods presented in Section 3.3.

3.4.1. General tabu search procedure

The general scheme of the tabu search procedure is summarized in Algorithm 2 while its main components are presented in the following subsections. Starting from a given input solution S, the algorithm iteratively examines other candidate solutions by exploring the two neighborhoods N_1 and N_2 of Section 3.3.

Algorithm 2: Pseudo-code of tabu search

Input: Input solution S, reduced neighborhood N_1, N_2, search depth ω, minimum perturbation length L_{min}, current global best solution found so far S_b.
Output: The current solution S_c, the local optimal solution found during tabu search S_l.

1 $NoImprove \leftarrow 0$; /* initialization of non-improvement iteration counter */
2 $S_c \leftarrow S$; /* S_c is the current solution */
3 $S_l \leftarrow S$; /* S_l records the local best solution found during tabu search */
4 **while** $NoImprove < \omega$ **do**
5   **if** $N_1(S_c) \cup N_2(S_c) \neq \emptyset$ **then**
6   Choose the best eligible neighboring solution $S' \in N_1(S_c) \cup N_2(S_c)$;
7         $S_c \leftarrow S'$;
8   **else**
9         $S_c \leftarrow perturbation(S_c, 1)$; /* Section 3.5 */
10   **if** $f(S_c) < f(S_l)$ **then**
11         $S_l \leftarrow S_c$;
12   **if** $f(S_c) < f(S_b)$ **then**
13         $NoImprove \leftarrow 0$;
14         $L \leftarrow L_{min}$;
15   **else**
16         $NoImprove \leftarrow NoImprove + 1$;
17   Update the tabu list TL and the frequency vectors F_e,F_j ; /* Section 3.5 */
18 **return** S_l, S_c;

At each iteration, TS chooses the best eligible neighboring solution among the available neighboring solutions in N_1 and N_2 to become the current solution S_c (lines 5-7). If no neighboring solution is available due to neighborhood reduction (see Section 3.3), the current solution is slightly perturbed (with perturbation length of 1) (line 9). Each time the current solution S_c becomes better than the recorded local best solution S_l found by the current tabu search run, S_l is updated by S_c (lines 10-11). If S_c is also better than the global best solution S_b from the BLS algorithm, the counter for consecutive non-improvement loops is reset to 0 and the perturbation length L is reset to its minimum L_{min} (lines 12-14). Otherwise, if the current iteration does not update S_b, the consecutive non-improvement counter $NoImprove$ is incremented by 1 (line 16). After the move operation (i.e., 2-*opt* or *j-swap*), the tabu list is updated according to the information of the move and current iteration. Meanwhile, we update the frequency vectors F_e and F_j, which respectively record the number of times an edge or a job is involved in a 2-*opt* or *j-swap* operation. When the $NoImprove$ counter reaches ω (a parameter), the search is considered to be trapped in a deep local optimum. In this case, the TS procedure terminates and returns the local best solution S_l and its current solution S_c. As shown in Algorithm 1 (Section 3.1), S_l will be used by the BLS algorithm to conditionally update the global best solution, while S_c will be used as input of the perturbation procedure (Section 3.5).

3.4.2. Tabu list management

Our BLS algorithm employs two tabu lists to avoid short-term cycling: an edge tabu list for 2-*opt* move and a job tabu list for *j-swap*. For 2-*opt*, once an edge e is deleted from the solution by the 2-*opt* move, the edge is added to the edge tabu list and is forbidden to be added again to the solution during the next consecutive β iterations (β is the so-called tabu tenure). So if any of the two new edges used by the 2-*opt* move is in tabu status, this move will not be performed. Similarly, for *j-swap*, when a job j is removed from a location node l, the job is forbidden to be assigned to this location node l again during the next consecutive β iterations.

During the search process, the best neighboring solution not forbidden by any tabu list is selected to replace the current solution. Notice that the tabu statue of a move is ignored if it can produce a solution better than the best solution ever found, which is called aspiration criterion (Glover & Laguna, 1997).

Now we consider the time complexity of the tabu search procedure. Each iteration of the tabu search jointly explores the 2-*opt* and *j-swap* neighborhoods (see Sections 3.3.1 and 3.3.2). Thus, the time complexity for each iteration of the tabu search is between between $O(n^2)$ and $O(n^3)$. As the search depth of the tabu search is ω, then the time complexity for the whole tabu search procedure is between $O(n^2\omega)$ and $O(n^3\omega)$.

3.5. Combined perturbation

Algorithm 3: Pseudo-code of combined perturbation

Input: Input solution S_c, frequency vectors F_e and F_j.
Output: Perturbed S.
Identify the job assigned to the node with the maximum completion time j_m in S_c
if $rand(0,1) < 0.5$ **then**
  //With probability 0.5, apply random perturbation;
  if $rand(0,1) < 0.5$ **then**
    $e_1, e_2 \leftarrow$ Two randomly choosed edges in S_c;
    $S \leftarrow$ Execute the 2-*opt* operation with e_1 and e_2;
  else
    $j_r \leftarrow$ A randomly choosed job;
    $S \leftarrow$ Exchange j_m and j_r;
else
  //With probability 0.5, apply frequency-based perturbation;
  if $rand(0,1) < 0.5$ **then**
    $e_3, e_4 \leftarrow$ Two edges with the least and the second least move frequency;
    $S \leftarrow$ Execute the 2-*opt* operation with e_3 and e_4;
  else
    $j_f \leftarrow$ The job with the least move frequency;
    $S \leftarrow$ Exchange j_m and j_f;
Update the frequency matrices F_e and F_j;
return S;

The tabu mechanism can prevent the search from the short-term cycles, but it may fail to prevent the algorithm from being trapped into deep local optima. To boost the global diversification of the algorithm, we introduce a dedicated perturbation strategy, which is triggered when the search is judged to be stagnating. According to the general BLS approach, our BLS algorithm for the TSPJ mixes two types of perturbations: informed perturbation (guided by historical search information related to move frequencies) and random perturbation. Indeed, frequency-guided perturbation has proved to be quite successful in several algorithms (Zhou & Hao, 2017; Li et al., 2020), while random perturbation is very popular in iterated local search algorithms.

The perturbation strategy is described in Algorithm 3. We choose the perturbation type with an equal probability. There are two move operators for perturbation, which are applied with an equal probability as well. For the random perturbation, we randomly select two edges to be replaced for the 2-*opt* move and a job to exchange with the job assigned to the location node with the maximum completion time (lines 3-9). For the frequency-based perturbation, we use the move frequency of the edges and the jobs to guide the perturbation (lines 10-17). For this, we maintain a long-term memory represented by two vectors F_e and F_j to record the number of times an edge or a job is involved in a 2-*opt* or j-*swap* move. We select the edges or location nodes that have the least and second least frequency to perform the perturbation. The specific steps are as follows:

- Initially, set the frequency of all edges and jobs to 0, i.e., $F_e(e) = 0, F_j(j) = 0$ for each edge $e \in E$, job $j \in J$, where E is the edge set and J is the job set.
- Subsequently, during the search process, F_e and F_j are updated each time a 2-*opt* or j-*swap* move is performed.
- Finally, the random or frequency-based perturbation is applied with equal probability and each chosen perturbation performs, with equal probability, either the 2-*opt* or j-*swap* move L times (L is the perturbation length). During the perturbation process, the frequency vectors are updated.

The time complexity of the perturbation procedure can be estimated as follows. Each step of the perturbation consists of applying either the 2-*opt* or j-*swap* move to the solution, requiring $O(1)$ time. Given that the perturbation repeats L times and the objective calculation after a perturbation move can be achieved in $O(n)$, the time complexity of the combined perturbation is bounded by $O(nL)$.

3.6. The time complexity analysis of the BLS algorithm

Our BLS algorithm starts with an initial solution produced by the TSPJ-NN procedure and then iterates the tabu search procedure and the combined perturbation procedure until the given time limit is reached. As indicated previously, the time complexity of the TSPJ-NN procedure is $O(n^3)$, the time complexity of the tabu search is bounded by $O(n^3\omega)$ (ω is the tabu search depth), and the time complexity of the combined perturbation procedure is bounded by $O(nL)$ (L is the perturbation length). Therefore, the time complexity for one iteration of our BLS algorithm is no more than $O(n^3\omega)$.

4. Computational results

We now report extensive computational results of the proposed BLS algorithm on benchmark instances and comparisons with state-of-the-art algorithms.

4.1. Benchmark instances

Our experiments are based on four sets of 310 instances with different sizes introduced in Mosayebi et al. (2021). The traveling distance between different locations of the instances in Set I is from the TSPLIB (Reinelt, 1991), whose optimal solutions for the TSP are known. The job processing time in different location nodes is generated randomly and is required to be between 50 to 80 percent of the optimal tour length. The traveling time and the job processing time in other sets are all produced randomly. The job processing time is required to be under 50 to 80 percent of the tour length obtained by the NNH-X heuristic and 2-*opt*. More information about these instances can be found in Mosayebi et al. (2021)[1].

Set I (10 instances): These instances are constructed based on 10 TSP instances from the TSPLIB: gr17, gr21, gr24, fri26, bays29, gr48, eil51, berlin52, eil76, and eil101.

Set II, Set III, Set IV (100 instances per set): The instances from these sets are constructed randomly. The node and job numbers are from 40 to 50, 400 to 500, and 1000 to 1200, respectively, which cover small, medium, and large instances.

4.2. Experimental protocol and reference algorithms

Table 3: Parameters tuning results.

Parameters	Section	Description	Considered values	Final value
ω	3.4	search depth	$\{0.14, 0.56, 0.30, 0.08\}$	0.08
β	3.4.2	tabu tenure	$\{0.35, 0.50, 0.07, 0.14\}$	0.07
L_{min}	3.5	minimum perturbation length	$\{0.06, 0.03, 0.04, 0.07\}$	0.04
L_{max}	3.5	maximum perturbation length	$\{0.27, 0.20, 0.15, 0.30\}$	0.15

Parameter setting. BLS has four parameters: tabu tenure β, search depth ω, minimum perturbation length l_0, maximum perturbation length l_{max}. In order to calibrate these parameters, we used the "IRACE" (López-Ibáñez et al., 2016) package to automatically identify a set of suitable parameter values. In this experiment, we randomly selected 1 instance from Set I, 3 instances from Set II, Set III, Set IV, respectively. The maximum number of runs (tuning budget) was set to be 1000. The candidate values of these parameters and the final selected values are shown in Table 3.

Reference algorithms. For our comparative study, we use as our reference methods the four heuristic algorithms proposed in Mosayebi et al. (2021)

[1]The instances are available at `https://github.com/TSPJLIB`

(detonated by Pro.I, Pro.II, Pro.III and Pro.IV), which represent the state-of-the-art for solving the TSPJ. Given that the source codes of these algorithms are unavailable, we faithfully re-implemented them, and verified that the results from our implementation match the results initially reported in Mosayebi et al. (2021). In addition to these main reference algorithms, we also run the CPLEX solver on the mathematical model presented in Appendix A with a time limit of 7200 seconds.

Experimental setting. BLS and the re-implemented reference algorithms were programmed in C++[2] and complied with the g++ compiler with the -O3 option. All the experiments were conducted on a computer with an Intel Xeon E5-2670 processor of 2.5 GHz CPU and 6 GB RAM running Linux. In order to eliminate stochastic factors, each algorithm was run 10 times on each instance with a different random seed per run.

Stopping condition. The reference algorithms are of deterministic nature and stop when no improvement can be reached. To make a fair comparison between BLS and the reference algorithms, we identify the average running time required by the four reference algorithms from Mosayebi et al. (2021) to reach their best solutions for the 10 instances of Set I and the average running time required by them to find their best solutions for the 100 instances of Set II, Set III and Set IV.

Specifically, for Set I, the cutoff time is set to be 60 seconds for eil101-J, 30 seconds for gr48-J, eil51-J, berlin52-J and eil76-J, 10 seconds for the 5 remaining instances. For the other sets, the cutoff time is 0.0012 seconds for Set II, 2.19 seconds for Set III and 33.93 seconds for Set IV. As such, our BLS algorithm is run under a fair stopping condition compared to the reference algorithms.

Finally, in order to better show the long term behavior of our BLS algorithm, we additionally run BLS with a relaxed stopping condition, 30 seconds for Set II, 50 seconds for Set III and 70 seconds for Set IV.

4.3. Computational results and comparison

This section reports the comparative results between the proposed BLS algorithm and reference algorithms (denotated by Pro.I, Pro.II, Pro.III and Pro.IV). The results are obtained according to the experimental protocol above.

The comparative results of the BLS and the four reference algorithms are summarized in Table 4 while the detailed results on each instance are provided in Appendix B (Table B.7 - Table B.13). In Table 4, the first column indicates the benchmark set. Column 2 presents the cut-off time running by our BLS, for Set I which has detailed results in Mosayebi et al. (2021). Column 3 shows the compared algorithms including the best-known solutions (BKS). Columns 4 - 6 indicate the number of instances for which BLS obtains a better, equal, or worse f_{best} value compared to each reference algorithm. To check the statistical significance of the compared results, the p-values from the Wilcoxon signed-rank

[2]The source codes of our BLS algorithm and the four re-implemented reference algorithms will be available at `https://github.com/YujiZou/TSPJ`

Table 4: Summary of the number of instances where BLS reports a better (B), equal (E) or worse (W) f_{best} value compared to the the best-known solutions (BKS) reported in Mosayebi et al. (2021) and each reference procedure (Pro.I, Pro.II, Pro.III and Pro.IV) presented in Mosayebi et al. (2021). We also show the p-values from the Wilcoxon singed-rank test on the benchmark sets between BLS and each reference algorithm.

Instance	Cut-off time(s)	Pair algorithms	B	E	W	p-value
		BLS vs. BKS	9	1	0	0.0077
		BLS vs. Pro.I	9	1	0	0.0077
Set I	60 or 30, 10	BLS vs. Pro.II	10	0	0	0.0020
		BLS vs. Pro.III	9	1	0	0.0077
		BLS vs. Pro.IV	9	1	0	0.0077
		BLS vs. BKS	46	18	36	0.6662
		BLS vs. Pro.I	80	13	7	2.389e-12
Set II	0.0012	BLS vs. Pro.II	69	7	29	1.384e-4
		BLS vs. Pro.III	76	16	8	8.332e-12
		BLS vs. Pro.IV	79	11	10	3.580e-12
		BLS vs. BKS	98	0	2	4.500e-18
		BLS vs. Pro.I	100	0	0	3.876e-18
Set III	2.19	BLS vs. Pro.II	98	0	2	4.371e-17
		BLS vs. Pro.III	100	0	0	3.877e-18
		BLS vs. Pro.IV	100	0	0	3.877e-18
		BLS vs. BKS	97	0	3	9.246e-17
		BLS vs. Pro.I	99	0	1	4.874e-17
Set IV	33.93	BLS vs. Pro.II	98	0	2	8.479e-16
		BLS vs. Pro.III	99	0	1	5.166e-17
		BLS vs. Pro.IV	98	0	2	5.399e-17
		BLS vs. BKS	84	15	1	1.363e-15
		BLS vs. Pro.I	91	8	1	8.580e-17
Set II	30.00	BLS vs. Pro.II	97	3	0	1.130e-17
		BLS vs. Pro.III	88	11	1	2.754e-16
		BLS vs. Pro.IV	91	8	1	1.231e-16
		BLS vs. BKS	99	0	1	3.980e-18
		BLS vs. Pro.I	100	0	0	3.881e-18
Set III	50.00	BLS vs. Pro.II	99	0	1	3.998e-18
		BLS vs. Pro.III	100	0	0	3.882e-18
		BLS vs. Pro.IV	100	0	0	3.883e-18
		BLS vs. BKS	99	0	1	7.547e-17
		BLS vs. Pro.I	99	0	1	3.186e-19
Set IV	70.00	BLS vs. Pro.II	99	0	1	7.548e-17
		BLS vs. Pro.III	99	0	1	3.234e-17
		BLS vs. Pro.IV	99	0	1	2.711e-17

test on the f_{best} values over the instances from the same set between BLS and the compared algorithms are shown in column 7 and a p-value smaller than 0.05 indicates a significant difference.

From the upper part of Table 4 (i.e., Rows 2 - 5), where we show the results of BLS under the same cut-off times as the reference algorithms, we observe that in terms of BKS, BLS performs remarkably well on Set I, Set III, and Set IV, by reporting 204 better results, five worse results, and equal results for the remaining instances for the 210 instances in total. Meanwhile, the dominance of BLS is less important on Set II under the very short cut-off time condition (0.0012 second). Still BLS gets 46 better results, 18 equal results, and 36 worse results.

From the lower part of Table 4 (i.e., Rows 6 - 8) and the detailed results of Appendix B, we observe that our BLS algorithm performs extremely well compared to the algorithms proposed in (Mosayebi et al., 2021) when the running time is prolonged (30 seconds for Set II, 50 seconds for Set III and 70 seconds for Set IV). In particular, BLS discovers 291 record-breaking results (new upper bounds) out of the 310 instances while matching the best-known results for 16 other instances. BLS reports a slightly worse result only on 3 instances (instance 36 in Set II, instance 48 in Set III, instance 19 in Set IV), with a small gap to the best-known result of 0.38%, 0.03%, and 0.2% respectively.

The small p-values (< 0.05) from the Wilcoxon signed-rank test between the results of BLS and each reference algorithm confirm that BLS significantly dominates each compared algorithm on each benchmark set.

4.4. Convergence analysis

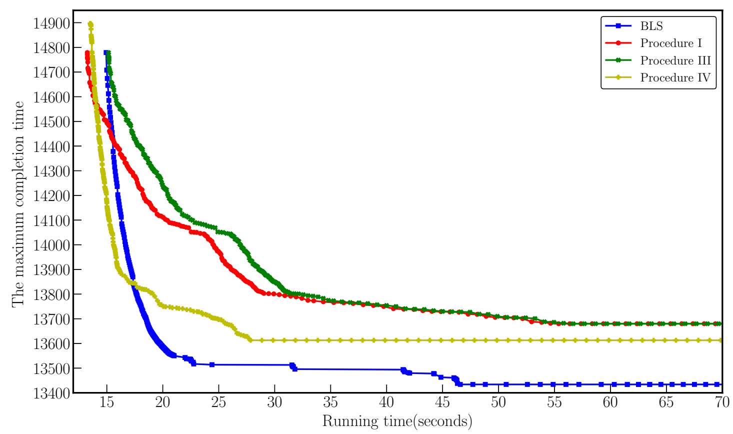


Figure 4: Convergence curves (running profiles) of BLS and three reference procedures for solving instance 61 from set IV.

To illustrate the running behavior of the compared algorithms during the search process, we provide in Fig. 4 their convergence curves (also called running profiles) of the BLS algorithm and three reference procedures on instance 61 from set IV, where the X-axis and Y-axis show the running time in seconds and the best objective value, respectively. The procedure II is ignored in this study because it focuses on the tour length optimization first, then constructs the final solution with the job reverse procedure. Hence, it is impossible to get the objective value of this procedure during the search process.

From Fig. 4, we observe that BLS and Pro.IV improve their best solutions more drastically than Pro.I and Pro.II. More importantly, BLS is able to continue its improvement along the time while the reference procedures fail to do so from some time point. This indicates that BLS has a lasting search capacity, making it possible for the algorithm to reach high-quality solutions that the reference algorithms cannot attain.

5. Assessment of algorithmic components

In this section, we analyze the two essential components of our BLS algorithm: neighborhood reduction and frequency-based perturbation.

5.1. Importance of the neighborhood reduction

To verify the importance of the neighborhood reduction on the BLS algorithm, we created a BLS variant named BLS-NoReduction, which did not use the neighborhood reduction. We run the two algorithms independently 10 times on the large size instances of set IV, using the parameters in Section 4.2 and a cut-off time of 70 seconds per run and per instance. Table 5 summarizes the comparative results of BLS and BLS-NoReduction on these 100 instances. From these two tables, one observes that BLS significantly dominates BLS-NoReduction, indicating that the performance of BLS deteriorates greatly if the neighborhood reduction is removed from the algorithm. This experiment confirms thus the usefulness of the neighborhood reduction as a critical technique contributing to the effectiveness of the BLS algorithm.

Table 5: Summarized results of BLS and BLS-NoReduction on the 100 large instances of Set III in terms of average result and running time together with the p-values from the Wilcoxon singed-rank test.

	BLS-NoReduction	BLS
Average	14932.40	13653.00
T_{avg}	70.01	31.09
p-value	4.960e-18	

5.2. Influence of the frequency-based perturbation

The frequency-based perturbation is designed to help the algorithm to escape local optimum traps. To show the influence of this strategy, we created a variant of BLS named BLS-Random which only applies the random perturbation mentioned in Section 3.5. We ran BLS-Random on all the instances under the relaxed stopping condition discussed in Section 4.2. The results were summarized in Table 6. In this table, columns 2 and 3 show for BLS and BLS-Random the grand average of the best objective values of all the instances of each set. Columns 5-7 present the number of instances for which the BLS algorithm reached a better, the same and a worse result compared to BLS-Random.

Table 6 shows that BLS with the frequency-based perturbation performs significantly better than the variant with the random perturbation only. This experiment confirms the benefit of the frequency-based perturbation for the performance of the BLS algorithm.

Table 6: Summarized results of BLS and BLS-Random, including the number of instances for which BLS reports a better (B), equal (T) or worse (W) average value compared to BLS-Random and the p-values from the Wilcoxon singed-rank test.

Instance	BLS_{Avg}	BLS-Random_{Avg}	p-value	B	T	W
Set I	3729.90	3730.41	-	2	8	0
Set II	280.85	281.58	0.00018	46	56	8
Set III	3463.19	3469.73	3.176e-7	62	24	14
Set IV	13625.51	13642.81	4.960e-18	69	26	5

6. Conclusion

As a combined routing and scheduling problem, the Traveling Salesman Problem with Job-time has a number of relevant practical applications in real life. This paper introduced a breakout local search algorithm, which employs a tabu search to explore two dedicated neighborhoods and applies a combined perturbation to escape local optima. A neighborhood reduction strategy was designed to identify promising neighbor solutions and accelerate the search process.

The proposed algorithm has been assessed on four sets of 310 instances in the literature and showed a highly competitive performance compared to the current best methods. Specifically, our algorithm has established new best-known results (updated upper bounds) for 291 out of the 310 benchmark instances ($> 93\%$ cases). Additional experiments have confirmed the usefulness of the neighborhood reduction and combined perturbation for the performance of the algorithm.

The algorithm in this work can be further improved. First, since the TSPJ involves two classic NP-hard problems, the search space is extremely complex. It is worth investigating other neighborhoods based on dedicated features of the problem to be able to explore the space more effectively. Second, the algorithm needs to make decisions during its search process (e.g., when should the perturbation be triggered, which type of perturbation should be applied...). To ensure informative decisions, reinforcement learning techniques could be useful. Third, memetic algorithms have been successfully applied to solve many NP-hard problems such as critical node problems (Zhou et al., 2019, 2023), TSP and routing problems (He et al., 2021; He & Hao, 2023; Vidal et al., 2013), and quadratic assignment problem (Benlic & Hao, 2015). It would be interesting to investigate this approach for solving the TSPJ as well. In particular, the proposed BLS algorithm or its variant can be beneficially used as the main local optimization component of a memetic algorithm. Finally, the existing exact approach for the problem relies on the general MIP solver CPLEX and can only obtain optimal solutions for some small instances. It would be interesting to design dedicated exact algorithms able to solve larger instances.

Acknowledgments

We are grateful to the reviewers for their useful comments and suggestions, which helped us to improve the work. We thank the authors of (Mosayebi et al., 2021) for making the TSPJ benchmark instances available and answering our questions. This work was partially supported by the National Natural Science Foundation Program of China (Grant No. 72122006). Support from CSC (Grant No. 202106050037) for the first author is acknowledged.

References

Aksan, Y., Dökeroglu, T., & Cosar, A. (2017). A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering*, *103*, 105–115.

Archetti, C., Feillet, D., Mor, A., & Speranza, M. G. (2018). An iterated local search for the traveling salesman problem with release dates and completion time minimization. *Computers & Operations Research*, *98*, 24–37.

Barbarosoğlu, G., Özdamar, L., & Cevik, A. (2002). An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. *European Journal of Operational Research*, *140*, 118–133.

Basnet, C. B., Foulds, L. R., & Wilson, J. M. (2006). Scheduling contractors' farm-to-farm crop harvesting operations. *International Transactions in Operational Research*, *13*, 1–15.

Bays, M. J., & Wettergren, T. A. (2017). Service agent–transport agent task planning incorporating robust scheduling techniques. *Robotics and Autonomous Systems*, *89*, 15–26.

Benlic, U., & Hao, J.-K. (2013a). Breakout local search for maximum clique problems. *Computers & Operations Research*, *40*, 192–206.

Benlic, U., & Hao, J.-K. (2013b). Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*, *26*, 1162–1173.

Benlic, U., & Hao, J.-K. (2013c). Breakout local search for the quadratic assignment problem. *Applied Mathematics and Computation*, *219*, 4800–4815.

Benlic, U., & Hao, J.-K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, *42*, 584–595.

Das, S. K., & Nagendra, P. (1997). Selection of routes in a flexible manufacturing facility. *International Journal of Production Economics*, *48*, 237–247.

Fu, Z.-H., & Hao, J.-K. (2014). Breakout local search for the steiner tree problem with revenue, budget and hop constraints. *European Journal of Operational Research*, *232*, 209–220.

Gavish, B., & Graves, S. C. (1978). The travelling salesman problem and related problems. *Working paper*, *GR-078-78*, Operations Research Center, Massachusetts Institute of Technology.

Ghandi, S., & Masehian, E. (2015). A breakout local search (BLS) method for solving the assembly sequence planning problem. *Engineering Applications of Artificial Intelligence*, *39*, 245–266.

Glover, F., & Laguna, M. (1997). *Tabu search*. Springer.

He, P., & Hao, J.-K. (2023). Memetic search for the minmax multiple traveling salesman problem with single and multiple depots. *European Journal of Operational Research*, *307*, 1055–1070.

He, P., Hao, J.-K., & Wu, Q. (2021). Grouping memetic search for the colored traveling salesmen problem. *Inf. Sci.*, *570*, 689–707.

Hoos, H. H., & Stützle, T. (2004). *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann.

Krari, M. E., Ahiod, B., & Benani, B. E. (2018). Breakout local search for the travelling salesman problem. *Computing and Informatics*, *37*, 656–672.

Lahrichi, Y., Deroussi, L., Grangeon, N., & Norre, S. (2020). A minmax path approach for balancing robotic assembly lines with sequence-dependent setup times. In *Proceedings of 13ème Conference Internationale de Modélisation, Optimisation et Simulation (MOSIM 2020), 12-14 November, Agadir, Marocco*.

Li, M., Hao, J.-K., & Wu, Q. (2020). General swap-based multiple neighborhood adaptive search for the maximum balanced biclique problem. *Computers & Operations Research*, *119*, 104922.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, *44*, 2245–2269.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In *Handbook of Metaheuristics* (pp. 320–353). Springer.

Mosayebi, M., Sodhi, M., & Wettergren, T. A. (2021). The traveling salesman problem with job-times (TSPJ). *Computers & Operations Research*, *129*, 105226.

Nogueira, B., Tavares, E., & Maciel, P. (2021). Iterated local search with tabu search for the weighted vertex coloring problem. *Computers & Operations Research*, *125*, 105087.

Rashidnejad, M., Ebrahimnejad, S., & Safari, J. (2018). A bi-objective model of preventive maintenance planning in distributed systems considering vehicle routing problem. *Computers & Industrial Engineering*, *120*, 360–381.

Reinelt, G. (1991). TSPLIB−A traveling salesman problem library. *ORSA Journal on Computing*, *3*, 376–384.

Taillard, É. D. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing*, *17*, 443–455.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers and Operations Research*, *40*, 475–489.

Zhang, L., Deng, Q., Lin, R., Gong, G., & Han, W. (2021). A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect. *Expert Systems with Applications*, *175*, 114843.

Zhou, Y., & Hao, J.-K. (2017). Frequency-driven tabu search for the maximum s-plex problem. *Computers & Operations Research*, *86*, 65–78.

Zhou, Y., Hao, J.-K., & Glover, F. W. (2019). Memetic search for identifying critical nodes in sparse graphs. *IEEE Transactions on Cybernetics*, *49*, 3699–3712.

Zhou, Y., Wang, G., Hao, J.-K., Geng, N., & Jiang, Z. (2023). A fast tri-individual memetic search approach for the distance-based critical node problem. *European Journal of Operational Research*, *308*, 540–554.

Zhou, Y., Xu, W., Fu, Z.-H., & Zhou, M. (2022). Multi-neighborhood simulated annealing-based iterated local search for colored traveling salesman problems. *IEEE Transactions on Intelligent Transportation Systems*, *23*, 16072–16082.

Appendix A. Mathematical model of the TSPJ (Mosayebi et al., 2021)

The mathematical model of the TSPJ presented in (Mosayebi et al., 2021) is based on the classical TSP model proposed by Gavish & Graves (1978). The TSPJ can be defined on an complete graph $G = (L, E)$ and a job set J. Let E be a set of edges, and $L = (l_0, l_1, \ldots, l_n)$ be the set of location nodes with the node 0 being the depot. Some other notions used in the model are:

Parameters:
Y_{lk}: The sequence number of edges visited;
C_{max}: The maximum completion time among all the jobs;
TS_l: The arriving time at node l, also the time to start work of the job assigned to this node;
D_{lk}: The travel-time from node l to node k;
jt_{lj}: The processing time of job j assigned to node l;

Variables:
X_{lk}: Indicating whether the edges E_{lk} is traversed from node l to node k;
Z_{lj}: Indicating whether the job j is assigned to the location node l;

On the basis of the above mentioned parameters and decision variables, the TSPJ is formulated as the following mixed integer program:

$$\min \quad C_{max} \tag{A.1}$$

$$\text{s.t.} \quad C_{max} \geq TS_l + \sum_{j=1}^{n} Z_{lj} jt_{lj} \quad \forall l = 1, \ldots, n \tag{A.2}$$

$$C_{max} \geq TS_l + Z_{l0} jt_{l0} \quad \forall l = 1, \ldots, n \tag{A.3}$$

$$\sum_{l=1}^{n} Z_{lj} = 1 \quad \forall j = 1, \ldots, n \tag{A.4}$$

$$\sum_{j=1}^{n} Z_{lj} = 1 \quad \forall l = 1, \ldots, n \tag{A.5}$$

$$\sum_{k=0}^{n} X_{lk} = 1 \quad \forall l = 0, \ldots, n \quad l \neq k \tag{A.6}$$

$$\sum_{l=0}^{n} X_{lk} = 1 \quad \forall k = 0, \ldots, n \quad l \neq k \tag{A.7}$$

$$\sum_{k=0}^{n} Y_{lk} - \sum_{k=0}^{n} Y_{kl} = 1 \quad \forall l = 1, \ldots, n \quad l \neq k \tag{A.8}$$

$$Y_{lk} \leq n X_{lk} \quad \forall l = 1, \ldots, n \quad \forall k = 0, \ldots, n \quad l \neq k \tag{A.9}$$

$$TS_l + D_{lk} - (1 - X_{lk}) M \leq TS_k \quad \forall l = 0, \ldots, n \quad \forall k = 1, \ldots, n \quad l \neq k \tag{A.10}$$

$$X_{lk}, Z_{lj} \in \{0, 1\}, \quad TS_l, Y_{lk} \geq 0 \quad \forall l = 1, \ldots, n \quad k = 0, \ldots, n \quad j = 1, \ldots, n \tag{A.11}$$

Eqs. (A.1)-(A.3) are used to calculate the objective value. Eqs. (A.4) and (A.5) are the job assignment constraints, which require that one job is assigned to one location node and vice versa. Eqs. (A.6) and (A.7) are the route restrictions that force that all the nodes are visited only once. Eqs. (A.8) and (A.9) are the subtour eliminators. Eq. (A.10) is the job processing start time restriction between different location nodes; the node arriving time (i.e., the job processing start time) cannot be less than the arriving time of the precedent location node plus the traveling time between them. M is a large enough number. Eq. (A.11) defines the domain of each variable X,Z,Y and TS.

Appendix B. Detailed results

This section shows detailed computational results of the proposed BLS algorithm compared to the results of the reference algorithms in (Mosayebi et al., 2021) on the four sets of TSPJ benchmark instances. The results of the CPLEX

MIP solver with the model of Appendix A with a time limit of 7200 seconds are also included for the (small) instances of Set I.

Table B.7 shows the results on the 10 instances of Set I. In this table, column 1 gives the name of instances, columns 2 and 3 show the upper bound and lower bound obtained by GAMS/CPLEX, columns 4 and 5 show the best results among the reference algorithms and the time needed to reach each result. The data in columns 2 - 5 are directly extracted from (Mosayebi et al., 2021). Columns 6 and 7 give the results obtained by our BLS algorithm under the cutoff conditions given in Section 4.2. gap-1 and gap-2 in columns 8 and 9 are the gaps between our results with respect to the best upper bound obtained by CPLEX and the reference algorithms, respectively. A negative gap (in bold) indicates an improved upper bound. We observe that BLS can find improved upper bounds for all instances except one case.

Table B.8 shows the results on the 100 instances of Set II. Column 1 is the name of the instance, columns 2 - 3 show the best results among the four reference algorithms of (Mosayebi et al., 2021) and the times needed to attain these best results. Columns 4 - 7 are the results of our BLS algorithm according to two stopping conditions (see Section 4.2) and the time to get these results. BLS-1 shows the results under the cutoff condition of (Mosayebi et al., 2021) (0.0012 seconds for Set II). BLS-2 shows the results under the relaxed cutoff condition (30 seconds for Set II). Columns 8 and 9 show the upper and lower bound from the CPLEX MIP solver using the model of Appendix A. Columns 10 - 11 show the gaps between BLS with the best-known results. A negative gap (in bold) indicates an improved upper bound. We observe that BLS can find improved upper bounds for all instances except 17 cases.

Tables B.9 and B.10 shows the results on the 200 instances of sets III and IV with the same information as in Table B.8 except that the results of CPLEX are not reported due to the fact that CPLEX reports within 7200 seconds bad results for the instances of Set III and even fails to find a feasible solution for the instances of Set IV. From these results, we observe that BLS is able to improve all previous upper bounds for the 200 instances of sets III and IV except 2 cases.

Finally, Tables B.11 - B.13 show the best results and the times to get these results of the four re-implemented algorithms of (Mosayebi et al., 2021) on each instance of Set II, Set III, and Set IV, respectively. These results were obtained under the cutoff condition of (Mosayebi et al., 2021). We observe that compared to the results reported in (Mosayebi et al., 2021), our reimplementation obtains slightly better results for Set II and better results for sets III and IV.

Table B.7: Computational results of the proposed BLS algorithm and comparison with the best-known results from the four references of (Mosayebi et al., 2021) on instances from Set I.

Instance	CPLEX		Reference algorithm		BLS		gap-1	gap-2
	UB	LB	f_{bks}	t_{bks}	f_{best}	t_{best}		
gr12-J	2760.00	2760.00	2760.00	0.13	2760.00	0.0001	0.000	0.00
gr21-J	7788.00	7712.00	7956.00	0.21	7788.00	0.001	0.000	-2.11
gr24-J	1806.00	1802.00	1818.00	0.34	1806.00	0.001	0.000	-0.66
fri26-J	1283.00	1282.94	1326.00	0.22	1283.00	0.001	0.000	-3.24
bays29-J	2937.00	2892.88	2940.00	0.57	**2916.00**	0.001	-0.715	-0.82
gr48-J	7288.00	7215.36	7499.00	2.48	**7282.00**	0.019	-0.082	-2.89
eil51-J	630.00	627.94	640.20	5.69	**628.51**	0.014	-0.237	-1.83
berlin52-J	11087.50	10976.96	11225.77	1.41	**11087.21**	0.001	-0.003	-1.23
eil76-J	802.27	799.47	822.46	3.32	**801.91**	6.260	-0.045	-2.50
eil101-J	947.42	940.59	975.94	14.75	**946.33**	59.340	-0.115	-3.03

Table B.8: Computational results of the proposed BLS algorithm and comparison with the best results from the four references of (Mosayebi et al., 2021) on the instances from Set II.

Instance	Reference algorithms		BLS-1		BLS-2		UB	LB	gap-1	gap-2
	f_{bks}	t_{bks}	f_{best}	t_{best}	f_{best}	t_{best}				
1	319	0.0011	301	0.0011	301	0.0022	367	125.00	-5.64	-5.64
2	273	0.0012	279	0.0010	267	0.0016	300	106.00	2.20	-2.20
3	288	0.0012	285	0.0010	**280**	0.0026	319	120.00	-1.04	-2.78
4	289	0.0019	291	0.0011	**283**	0.0018	376	114.00	0.69	-2.08
5	282	0.0025	281	0.0011	**280**	0.0034	356	118.00	-0.35	-0.71
6	305	0.0006	303	0.0008	**299**	0.0009	347	127.00	-0.66	-1.97
7	286	0.0025	288	0.0012	**285**	0.0021	332	113.00	0.70	-0.35
8	254	0.0007	257	0.0010	**252**	0.0023	300	104.00	1.18	-0.79
9	290	0.0007	287	0.0007	**285**	0.0019	313	121.21	-1.03	-1.72
10	313	0.0017	314	0.0011	313	0.0019	379	127.00	0.32	0.00
11	299	0.0008	303	0.0010	**297**	0.0025	334	121.00	1.34	-0.67
12	284	0.0032	282	0.0010	**280**	0.0027	338	124.00	-0.70	-1.41
13	268	0.0013	268	0.0012	**263**	0.0018	336	105.00	0.00	-1.87
14	270	0.0011	267	0.0006	267	0.0008	335	105.00	-1.11	-1.11
15	271	0.0006	271	0.0012	**270**	0.0015	313	111.82	0.00	-0.37
16	299	0.0013	303	0.0011	**297**	0.0033	371	127.00	1.34	-0.67
17	265	0.0010	270	0.0011	**263**	0.0027	317	112.00	1.89	-0.75
18	258	0.0008	252	0.0010	252	0.0010	286	100.15	-2.33	-2.33
19	248	0.0008	241	0.0011	**239**	0.0020	264	98.67	-2.82	-3.63
20	289	0.0006	283	0.0011	283	0.0012	369	120.00	-2.08	-2.08
21	285	0.0009	291	0.0010	**284**	0.0009	329	117.00	2.11	-0.35
22	236	0.0008	242	0.0011	236	0.0030	257	99.00	2.54	0.00
23	279	0.0007	283	0.0010	**276**	0.0016	313	108.00	1.43	-1.08
24	298	0.0010	295	0.0012	**292**	0.0021	376	121.00	-1.01	-2.01
25	240	0.0007	248	0.0009	**238**	0.0013	273	97.00	3.33	-0.83
26	291	0.0008	298	0.0012	291	0.0034	356	125.00	2.41	0.00
27	297	0.0018	297	0.0011	**296**	0.0030	403	131.00	0.00	-0.34
28	285	0.0019	283	0.0012	283	0.0012	331	117.00	-0.70	-0.70
29	299	0.0023	299	0.0009	299	0.0015	328	132.00	0.00	0.00
30	304	0.0021	300	0.0012	**298**	0.0025	369	125.00	-1.32	-1.97
31	261	0.0008	259	0.0012	259	0.0017	314	114.00	-0.77	-0.77
32	325	0.0010	319	0.0019	319	0.0012	362	137.26	-1.85	-1.85
33	306	0.0018	306	0.0011	306	0.0019	359	122.00	0.00	0.00
34	294	0.0015	294	0.0012	294	0.0014	337	128.00	0.00	0.00
35	283	0.0018	280	0.0019	280	0.0014	314	120.00	-1.06	-1.06
36	**263**	0.0016	264	0.0008	264	0.0008	299	119.00	0.38	0.38
37	279	0.0008	271	0.0010	271	0.0015	294	111.00	-2.87	-2.87
38	249	0.0006	250	0.0011	**247**	0.0010	295	105.00	0.40	-0.80
39	276	0.0009	279	0.0009	**274**	0.0007	348	126.00	1.09	-0.72
40	256	0.0012	258	0.0009	**255**	0.0021	293	107.00	0.78	-0.39
41	287	0.0005	291	0.0009	**282**	0.0013	303	121.00	1.39	-1.74
42	297	0.0012	295	0.0010	**292**	0.0024	328	116.26	-0.67	-1.68
43	297	0.0013	294	0.0011	294	0.0018	325	124.00	-1.01	-1.01
44	271	0.0011	267	0.0011	267	0.0014	301	109.00	-1.48	-1.48
45	325	0.0016	327	0.0009	**322**	0.0039	437	133.00	0.62	-0.92
46	248	0.0007	246	0.0011	**242**	0.0011	256	103.00	-0.81	-2.42
47	325	0.0009	319	0.0012	**317**	0.0013	368	133.00	-1.85	-2.46
48	300	0.0015	299	0.0011	**295**	0.0018	330	124.00	-0.33	-1.67
49	265	0.0011	266	0.0019	**264**	0.0016	337	115.99	0.38	-0.38

50	293	0.0020	292	0.0009	**291**	0.0035	325	125.00	-0.34	-0.68
51	275	0.0011	275	0.0009	275	0.0011	316	118.00	0.00	0.00
52	341	0.0016	338	0.0010	338	0.0022	412	138.00	-0.88	-0.88
53	286	0.0012	286	0.0009	**285**	0.0025	349	117.00	0.00	-0.35
54	267	0.0007	269	0.0010	**265**	0.0017	307	114.00	0.75	-0.75
55	304	0.0005	305	0.0010	**300**	0.0023	342	130.00	0.33	-1.32
56	289	0.0008	292	0.0007	**288**	0.0008	339	127.00	1.04	-0.35
57	257	0.0015	257	0.0011	**254**	0.0026	305	111.00	0.00	-1.17
58	341	0.0013	336	0.0008	336	0.0011	370	142.00	-1.47	-1.47
59	288	0.0010	285	0.0009	**284**	0.0021	371	118.00	-1.04	-1.39
60	257	0.0010	256	0.0009	256	0.0029	308	100.00	-0.39	-0.39
61	307	0.0019	313	0.0011	307	0.0026	366	131.00	1.95	0.00
62	289	0.0016	289	0.0010	289	0.0012	332	120.00	0.00	0.00
63	274	0.0015	274	0.0009	274	0.0011	377	113.00	0.00	0.00
64	290	0.0008	309	0.0010	**283**	0.0039	289	111.06	6.55	-2.41
65	269	0.0011	268	0.0008	268	0.0012	422	118.00	-0.37	-0.37
66	290	0.0019	287	0.0012	**286**	0.0030	482	132.00	-1.03	-1.38
67	320	0.0012	325	0.0012	**319**	0.0023	331	126.00	1.56	-0.31
68	301	0.0018	299	0.0013	**295**	0.0037	369	121.00	-0.66	-1.99
69	300	0.0023	311	0.0012	**297**	0.0023	299	114.00	3.67	-1.00
70	273	0.0012	273	0.0008	273	0.0008	299	114.00	0.00	0.00
71	295	0.0012	295	0.0009	295	0.0009	384	118.00	0.00	0.00
72	259	0.0009	256	0.0009	256	0.0008	271	107.00	-1.16	-1.16
73	278	0.0018	277	0.0009	**275**	0.0015	320	119.00	-0.36	-1.08
74	271	0.0009	271	0.0012	271	0.0011	305	113.00	0.00	0.00
75	212	0.0006	212	0.0008	212	0.0005	259	88.00	0.00	0.00
76	297	0.0008	294	0.0009	**293**	0.0010	321	120.00	-1.01	-1.35
77	274	0.0011	273	0.0010	**271**	0.0018	313	122.00	-0.36	-1.09
78	258	0.0010	261	0.0011	**255**	0.0027	291	111.00	1.16	-1.16
79	281	0.0013	280	0.0010	280	0.0010	315	113.00	-0.36	-0.36
80	280	0.0008	281	0.0009	**271**	0.0024	299	117.00	0.36	-3.21
81	259	0.0015	261	0.0009	259	0.0012	284	112.00	0.77	0.00
82	345	0.0019	348	0.0012	**339**	0.0016	482	146.00	0.87	-1.74
83	259	0.0018	257	0.0011	257	0.0012	341	104.00	-0.77	-0.77
84	272	0.0011	267	0.0006	**264**	0.0020	274	114.00	-1.84	-2.94
85	300	0.0012	300	0.0008	**296**	0.0027	333	123.00	0.00	-1.33
86	338	0.0006	342	0.0007	**333**	0.0023	374	139.00	1.18	-1.48
87	261	0.0009	261	0.0012	**257**	0.0022	303	107.00	0.00	-1.53
88	305	0.0017	299	0.0010	**295**	0.0028	328	126.00	-1.97	-3.28
89	288	0.0008	289	0.0007	**287**	0.0010	302	113.00	0.35	-0.35
90	322	0.0011	327	0.0012	**321**	0.0014	346	127.00	1.55	-0.31
91	281	0.0012	280	0.0010	**274**	0.0030	360	115.00	-0.36	-2.49
92	300	0.0006	309	0.0007	**296**	0.0025	389	130.00	3.00	-1.33
93	309	0.0012	299	0.0010	**295**	0.0022	312	120.00	-3.24	-4.53
94	309	0.0011	308	0.0012	**307**	0.0030	323	123.00	-0.32	-0.65
95	260	0.0013	256	0.0012	256	0.0013	269	103.00	-1.54	-1.54
96	268	0.0010	273	0.0012	**263**	0.0043	273	106.00	1.87	-1.87
97	256	0.0008	249	0.0007	249	0.0013	259	99.00	-2.73	-2.73
98	298	0.0007	298	0.0012	**289**	0.0019	304	116.00	0.00	-3.02
99	278	0.0014	277	0.0012	**265**	0.0017	277	106.00	-0.36	-4.68
100	249	0.0008	248	0.0012	248	0.0009	263	99.00	-0.40	-0.40
Avg	284.44	0.0012	284.33	0.0010	**280.85**	0.0019	-	-	-0.03	-1.25

Table B.9: Computational results of the proposed BLS algorithm and comparison with the best results from the four references of (Mosayebi et al., 2021) on the instances of Set III.

Instance	Reference algorithms		BLS-1		BLS-2		gap-1	gap-2
	f_{bks}	t_{bks}	f_{best}	t_{best}	f_{best}	t_{best}		
1	3235	0.71	3219	2.18	**3187**	2.64	-0.49	-1.48
2	3494	0.57	3452	0.91	**3442**	2.09	-1.2	-1.49
3	3467	3.59	3436	1.59	**3426**	3.36	-0.89	-1.18
4	3465	4.44	3447	0.81	3447	1.10	-0.52	-0.52
5	3673	0.66	3647	1.94	**3637**	3.47	-0.71	-0.98
6	3273	0.61	**3233**	1.12	3237	0.97	-1.22	-1.1
7	3571	0.93	3537	1.72	**3521**	5.09	-0.95	-1.4
8	3604	0.68	3562	2.06	**3558**	3.72	-1.17	-1.28
9	3335	0.76	3330	2.07	**3320**	2.31	-0.15	-0.45
10	3544	0.89	3514	1.60	**3497**	2.48	-0.85	-1.33
11	3401	0.73	**3362**	2.18	3363	1.38	-1.15	-1.12
12	3510	1.88	3473	1.63	**3457**	2.87	-1.05	-1.51
13	3523	1.14	3514	1.96	**3476**	4.63	-0.26	-1.33
14	3480	0.42	3456	1.09	**3444**	1.38	-0.69	-1.03
15	3462	2.04	3444	1.26	3444	2.03	-0.52	-0.52
16	3699	7.22	3661	1.87	**3655**	3.59	-1.03	-1.19
17	3571	7.89	3502	1.79	**3491**	4.05	-1.93	-2.24
18	3593	0.99	3572	1.20	**3558**	2.70	-0.58	-0.97
19	3501	3.68	**3432**	1.77	3438	1.30	-1.97	-1.8
20	3504	0.72	3476	0.88	3476	1.30	-0.80	-0.80
21	3589	0.50	3564	2.08	**3561**	2.24	-0.70	-0.78
22	3685	9.75	3654	1.58	**3649**	3.52	-0.84	-0.98
23	3398	2.66	3354	1.34	**3349**	2.29	-1.29	-1.44
24	3476	2.41	3455	2.07	**3412**	10.41	-0.60	-1.84
25	3671	7.15	3617	2.00	3606	**6.36**	-1.47	-1.77
26	3477	0.73	3433	1.90	3433	2.49	-1.27	-1.27
27	3576	0.90	3584	2.19	**3554**	2.66	0.22	-0.62
28	3505	0.82	**3456**	2.03	3467	1.83	-1.4	-1.08
29	3327	0.53	3269	0.85	3269	0.90	-1.74	-1.74
30	3255	1.69	3215	0.93	**3210**	3.37	-1.23	-1.38
31	3324	0.37	3284	2.08	**3256**	1.39	-1.2	-2.05
32	3559	0.91	3563	1.62	**3538**	2.57	0.11	-0.59
33	3667	0.96	3615	2.05	**3612**	1.63	-1.42	-1.5
34	3492	5.32	3434	1.29	**3424**	2.90	-1.66	-1.95
35	3505	1.87	3429	1.49	**3427**	3.06	-2.17	-2.23
36	3467	0.40	**3433**	1.54	3438	1.46	-0.98	-0.84
37	3663	4.94	3651	1.83	3651	4.36	-0.33	-0.33
38	3621	0.91	3581	1.72	**3578**	2.82	-1.10	-1.19
39	3432	0.53	3390	1.75	**3387**	2.66	-1.22	-1.31
40	3293	0.35	3250	1.11	**3248**	1.31	-1.31	-1.37
41	3465	0.8	3429	1.05	**3419**	2.66	-1.04	-1.33
42	3506	0.79	3479	2.18	**3471**	2.87	-0.77	-1.00
43	3406	1.68	3390	1.56	**3383**	1.95	-0.47	-0.68
44	3616	6.17	3550	2.02	**3548**	5.00	-1.83	-1.88
45	3587	2.61	**3529**	1.48	3526	2.52	-1.62	-1.70
46	3386	2.27	3347	1.46	**3333**	0.96	-1.15	-1.57
47	3335	0.69	3306	0.97	**3290**	3.14	-0.87	-1.35
48	**3558**	0.78	3559	2.15	3559	1.98	-0.25	0.03
49	3602	0.83	**3550**	2.02	3562	1.73	-1.44	-1.11
50	3594	8.24	3529	1.85	**3526**	3.15	-1.81	-1.89
51	3373	0.52	3336	0.72	3336	1.14	-1.10	-1.10
52	3317	0.67	3299	1.45	**3293**	4.29	-0.54	-0.72
53	3528	0.44	**3473**	2.08	3480	1.14	-1.56	-1.36
54	3368	7.44	3347	1.07	**3340**	2.23	-0.62	-0.83
55	3739	0.57	3677	2.06	3677	1.56	-1.66	-1.66
56	3682	0.51	3648	2.16	**3626**	4.59	-0.92	-1.52
57	3709	8.86	3671	1.88	**3665**	3.73	-1.02	-1.19
58	3633	0.63	3591	1.85	3591	2.47	-1.16	-1.16
59	3302	0.4	3250	0.93	**3244**	1.12	-1.57	-1.76
60	3392	3.37	3364	0.85	3364	0.94	-0.83	-0.83
61	3421	3.92	3367	1.48	**3365**	1.60	-1.58	-1.64
62	3609	0.74	3574	1.61	**3567**	2.03	-0.97	-1.16
63	3558	0.51	3532	1.67	**3526**	3.35	-0.73	-0.90
64	3470	2.03	3452	1.16	3452	1.27	-0.52	-0.52
65	3732	1.01	3691	1.50	**3687**	2.11	-1.10	-1.21
66	3645	4.25	3610	1.81	**3603**	3.35	-0.96	-1.15
67	3427	2.38	3421	2.05	**3420**	2.14	-0.18	-0.20
68	3343	2.11	3303	1.61	**3296**	2.07	-1.20	-1.41
69	3641	0.56	3598	1.90	**3568**	3.61	-1.18	-2.00
70	3520	0.89	3495	1.16	**3482**	2.08	-0.71	-1.08
71	3599	0.76	3564	1.00	**3563**	2.04	-0.97	-1.00
72	3623	5.24	3576	2.12	**3558**	2.63	-1.30	-1.79
73	3413	4.51	3339	1.88	**3335**	2.25	-2.17	-2.29
74	3281	0.6	**3252**	1.86	3255	1.61	-0.88	-0.79
75	3390	3.65	3368	0.83	**3358**	2.25	-0.65	-0.94
76	3645	0.85	3601	1.70	**3595**	2.87	-1.21	-1.37
77	3472	0.56	3445	1.62	3445	1.91	-0.78	-0.78
78	3444	3.48	3421	0.95	3421	1.07	-0.67	-0.67
79	3676	1.07	3646	2.14	**3608**	3.25	-0.82	-1.85
80	3554	0.76	3535	1.87	**3528**	7.22	-0.53	-0.73
81	3437	3.66	3381	0.64	3381	0.80	-1.63	-1.63
82	3475	1.92	3429	1.89	**3419**	4.39	-1.32	-1.61
83	3428	0.73	**3364**	2.08	3369	1.86	-1.87	-1.72

84	3567	4.36	3523	1.98	**3503**	2.07	-1.23	-1.79
85	3508	4.96	**3445**	1.49	3446	1.44	-1.80	-1.77
86	3588	1.54	3535	1.39	**3503**	2.87	-1.48	-2.37
87	3541	0.62	3503	1.64	**3494**	3.01	-1.07	-1.33
88	3374	3.48	3339	1.25	**3334**	2.09	-1.04	-1.19
89	3442	0.65	3382	1.14	3382	1.29	-1.74	-1.74
90	3839	1.14	3813	1.92	**3798**	5.44	-0.68	-1.07
91	3448	2.46	3436	1.57	**3431**	1.91	-0.35	-0.49
92	3522	0.81	3503	1.63	**3502**	3.94	-0.54	-0.57
93	3315	2.13	3294	1.91	**3281**	3.96	-0.63	-1.03
94	3486	1.21	**3431**	1.80	3442	0.90	-1.58	-1.26
95	3629	0.89	3576	1.61	3576	2.12	-1.46	-1.46
96	3441	0.63	3410	1.26	**3409**	1.42	-0.90	-0.93
97	3637	5.41	3623	2.03	**3594**	3.72	-0.38	-1.18
98	3311	2.80	3275	1.63	**3264**	3.51	-1.09	-1.42
99	3746	1.92	3710	1.20	**3709**	1.74	-0.96	-0.99
100	3510	5.33	**3470**	1.61	3474	1.59	-1.14	-1.03
Avg	3506.92	2.19	3470.46	1.60	**3463.19**	2.63	-1.04	-1.25

Table B.10: Computational results of the proposed BLS algorithm and comparison with the best results from the four references of (Mosayebi et al., 2021) on the instances of Set IV.

Instance	Reference algorithms		BLS-1		BLS-2		gap-1	gap-2
	f_{bks}	t_{bks}	f_{best}	t_{best}	f_{best}	t_{best}		
1	13946	8.61	13735	33.85	13735	39.17	-1.51	-1.51
2	14434	87.97	14327	32.82	**14264**	49.50	-0.74	-1.18
3	13313	12.45	13192	32.43	**13178**	30.60	-0.91	-1.01
4	13682	25.26	13480	25.01	13480	37.93	-1.48	-1.48
5	13835	16.46	13732	29.15	**13708**	31.91	-0.74	-0.92
6	14204	16.16	14114	30.70	**14087**	41.38	-0.63	-0.82
7	13745	49.29	**13619**	25.02	13620	28.37	-0.92	-0.91
8	13831	13.34	13714	23.37	**13673**	56.85	-0.85	-1.14
9	14016	226.50	13857	27.91	13857	30.40	-1.13	-1.13
10	13736	60.12	13553	27.87	**13543**	31.42	-1.33	-1.41
11	13192	9.78	13080	25.22	**13058**	25.61	-0.85	-1.02
12	13563	14.71	13461	28.37	**13446**	33.03	-0.75	-0.86
13	13149	11.79	13078	20.30	**13067**	29.83	-0.54	-0.62
14	14003	62.32	13914	33.13	**13900**	50.29	-0.64	-0.74
15	13619	12.19	13568	21.32	13568	25.77	-0.37	-0.37
16	13309	43.21	13154	17.71	**13147**	28.34	-1.16	-1.22
17	13273	12.02	13108	19.61	**13092**	22.71	-1.24	-1.36
18	13677	13.04	13574	25.67	13574	28.51	-0.75	-0.75
19	**13978**	16.26	14009	30.77	14006	57.04	0.22	0.20
20	14095	14.55	14009	32.70	**13973**	69.30	-0.61	-0.87
21	13535	8.21	13341	16.36	13341	26.83	-1.43	-1.43
22	13971	13.50	**13967**	20.51	13974	29.67	-0.03	0.02
23	13392	52.99	13224	31.24	**13211**	33.29	-1.25	-1.35
24	13991	16.06	13964	30.86	13964	35.25	-0.19	-0.19
25	14261	19.24	14155	25.19	**14154**	41.11	-0.74	-0.75
26	14023	53.14	14039	28.40	14015	35.74	0.11	-0.06
27	13240	23.55	13161	18.32	13161	19.87	-0.60	-0.60
28	13698	12.06	13611	27.69	**13575**	28.87	-0.64	-0.90
29	13462	27.81	**13332**	30.25	13345	37.92	-0.97	-0.87
30	13608	21.15	13495	26.58	**13486**	27.72	-0.83	-0.90
31	13771	84.10	13583	27.97	**13548**	28.24	-1.37	-1.62
32	13745	15.77	13634	18.20	**13630**	32.29	-0.81	-0.84
33	13709	13.27	**14087**	39.40	14068	72.28	2.76	2.62
34	14352	17.80	14244	28.83	**14206**	40.31	-0.75	-1.02
35	12854	9.98	12667	26.09	**12616**	45.64	-1.45	-1.85
36	13869	7.05	13715	24.62	**13714**	49.30	-1.11	-1.12
37	13729	12.83	13675	28.29	**13653**	38.10	-0.39	-0.55
38	13775	16.77	13632	31.54	**13628**	35.84	-1.04	-1.07
39	14243	9.98	14147	28.95	**14128**	62.30	-0.67	-0.81
40	13962	17.01	13909	29.19	**13908**	26.05	-0.38	-0.39
41	13463	8.67	13400	33.55	**13390**	29.61	-0.47	-0.54
42	13917	39.85	13843	28.37	**13810**	41.86	-0.53	-0.77
43	13698	91.51	**13521**	26.51	13526	29.75	-1.29	-1.26
44	12963	10.09	12704	30.60	**12673**	17.93	-2.00	-2.24
45	13222	25.29	13087	23.57	**13078**	29.93	-1.02	-1.09
46	13883	14.89	13761	33.94	**13739**	47.84	-0.88	-1.04
47	14677	9.78	14552	33.99	**14545**	58.54	-0.85	-0.90
48	14090	18.41	14024	31.02	**14016**	38.96	-0.47	-0.53
49	14340	10.19	14145	27.09	**14143**	35.93	-1.36	-1.37
50	13896	8.95	13819	31.18	**13814**	49.70	-0.55	-0.59
51	14157	83.48	13992	29.98	**13975**	63.56	-1.17	-1.29
52	13892	229.90	13796	32.56	**13759**	62.43	-0.69	-0.96
53	13917	39.03	13786	33.26	**13776**	36.39	-0.94	-1.01
54	13731	39.07	13630	19.10	**13613**	64.17	-0.74	-0.86
55	13776	8.72	13602	25.21	13602	29.26	-1.26	-1.26
56	13803	21.43	13673	31.91	**13666**	35.44	-0.94	-0.99
57	13881	33.41	13802	25.75	**13793**	30.41	-0.57	-0.63
58	14371	40.68	**14187**	32.08	14193	39.89	-1.28	-1.24
59	13669	6.55	13557	33.50	**13509**	36.85	-0.82	-1.17

60	13799	7.51	13678	26.51	**13669**	30.32	-0.88	-0.94
61	13584	23.87	13476	30.07	**13434**	41.71	-0.80	-1.10
62	13641	200.77	13492	26.85	**13463**	32.58	-1.09	-1.30
63	13923	15.61	13809	32.68	**13795**	37.01	-0.82	-0.92
64	13176	26.31	13053	21.17	13053	26.19	-0.93	-0.93
65	13719	8.03	13594	33.96	**13588**	28.63	-0.91	-0.95
66	14478	9.74	14378	33.51	**14321**	57.54	-0.69	-1.08
67	13597	39.40	13520	22.60	13520	24.48	-0.57	-0.57
68	13441	61.90	13345	21.61	**13333**	45.08	-0.71	-0.80
69	13667	29.06	13637	32.16	**13609**	58.51	-0.22	-0.42
70	13769	16.39	13615	32.27	**13602**	42.12	-1.12	-1.21
71	13506	11.19	13336	28.59	**13289**	62.25	-1.26	-1.61
72	13859	7.99	13701	21.93	**13660**	36.12	-1.14	-1.44
73	14165	41.19	14051	28.13	**14034**	62.26	-0.80	-0.92
74	14189	133.59	14076	30.08	**14066**	47.77	-0.80	-0.87
75	13610	14.23	13565	31.31	**13537**	40.55	-0.33	-0.54
76	13982	15.76	13813	23.10	13813	41.84	-1.21	-1.21
77	13331	5.86	13139	22.44	**13129**	38.59	-1.44	-1.52
78	14157	100.31	14063	23.25	**14008**	42.93	-0.66	-1.05
79	13644	8.09	13495	26.62	**13474**	28.82	-1.09	-1.25
80	13606	29.39	**13487**	32.02	13511	25.87	-0.87	-0.70
81	13109	11.71	13003	18.06	**12949**	44.17	-0.81	-1.22
82	13979	15.17	13865	31.62	**13833**	55.26	-0.82	-1.04
83	13240	19.54	13036	26.31	**13015**	31.86	-1.54	-1.70
84	13710	23.06	13558	32.88	**13548**	25.89	-1.11	-1.18
85	13194	6.26	13123	26.36	**13095**	42.50	-0.54	-0.75
86	14443	8.14	14335	23.56	**14295**	69.77	-0.75	-1.02
87	13115	28.09	**13022**	33.97	13041	22.35	-0.71	-0.56
88	13559	6.87	13432	19.51	**13404**	28.86	-0.94	-1.14
89	13460	26.10	13330	25.61	13330	23.65	-0.97	-0.97
90	14117	12.97	13984	33.43	**13980**	57.62	-0.94	-0.97
91	14093	28.81	13997	30.54	**13990**	45.51	-0.68	-0.73
92	13648	15.04	13481	21.65	**13472**	37.91	-1.22	-1.29
93	13764	13.44	13662	28.79	**13658**	66.20	-0.74	-0.77
94	13889	13.85	13769	31.39	**13739**	54.22	-0.86	-1.08
95	13888	13.40	**13850**	31.45	13855	27.41	-0.27	-0.24
96	13540	15.88	13366	23.18	**13331**	23.35	-1.29	-1.54
97	13180	128.84	13111	25.86	**13097**	69.16	-0.52	-0.63
98	13606	101.99	13357	32.85	**13349**	58.25	-1.83	-1.89
99	14919	18.12	14758	31.25	**14725**	65.41	-1.08	-1.30
100	13236	161.45	13092	21.33	**13091**	27.79	-1.09	-1.10
Avg	13756.68	33.93	13642.04	27.75	**13627.03**	39.80	-0.84	-0.94

Table B.11: Results of the four re-implemented algorithms of (Mosayebi et al., 2021) on the instances of Set II.

Instance	Pro.I		Pro.II		Pro.III		Pro.IV	
	f_{best}	t_{best}	f_{best}	t_{best}	f_{best}	t_{best}	f_{best}	t_{best}
1	319	0.0011	320	0.0006	319	0.0011	319	0.0012
2	280	0.0009	275	0.0006	280	0.0009	273	0.0012
3	296	0.0019	288	0.0012	296	0.0019	292	0.0023
4	289	0.0019	293	0.0011	289	0.0015	289	0.0018
5	282	0.0025	282	0.0012	282	0.0018	282	0.0026
6	309	0.0013	305	0.0006	309	0.0010	316	0.0013
7	288	0.0015	292	0.0008	288	0.0012	286	0.0025
8	264	0.0010	254	0.0007	264	0.0009	264	0.0016
9	293	0.0008	290	0.0007	293	0.0008	298	0.0017
10	316	0.0015	320	0.0009	316	0.0016	313	0.0017
11	305	0.0015	299	0.0008	305	0.0016	306	0.0017
12	284	0.0032	287	0.0012	284	0.0029	284	0.0026
13	276	0.0017	268	0.0013	276	0.0018	272	0.0017
14	270	0.0011	270	0.0008	270	0.0011	270	0.0011
15	277	0.0010	271	0.0006	277	0.0009	273	0.0015
16	303	0.0024	299	0.0013	303	0.0020	303	0.0019
17	271	0.0011	265	0.0010	271	0.0010	271	0.0012
18	258	0.0008	258	0.0008	258	0.0007	258	0.0010
19	251	0.0010	248	0.0008	251	0.0008	257	0.0012
20	293	0.0019	289	0.0006	293	0.0015	295	0.0022
21	291	0.0011	285	0.0009	291	0.0010	291	0.0013
22	236	0.0008	239	0.0006	236	0.0008	236	0.0008
23	288	0.0013	279	0.0007	288	0.0011	288	0.0012
24	299	0.0020	298	0.0010	299	0.0021	299	0.0021
25	256	0.0014	240	0.0007	256	0.0014	256	0.0012
26	305	0.0010	291	0.0008	305	0.0009	305	0.0008
27	297	0.0018	305	0.0012	297	0.0018	297	0.0019
28	285	0.0019	288	0.0010	285	0.0018	290	0.0014
29	299	0.0023	299	0.0011	299	0.0024	304	0.0016
30	307	0.0018	305	0.0010	307	0.0018	304	0.0021
31	268	0.0012	261	0.0008	267	0.0011	267	0.0011
32	332	0.0012	328	0.0008	332	0.0011	325	0.0010
33	312	0.0017	309	0.0015	306	0.0018	312	0.0017
34	294	0.0015	311	0.0010	294	0.00151	294	0.0016
35	283	0.0018	290	0.0012	283	0.0019	297	0.0016

36	263	0.0016	265	0.0008	263	0.0015	263	0.0016
37	284	0.0011	279	0.0008	284	0.0014	280	0.0017
38	251	0.0011	249	0.0006	251	0.0009	251	0.0008
39	280	0.0014	276	0.0009	280	0.0012	280	0.0014
40	261	0.0029	256	0.0012	261	0.0023	261	0.0025
41	295	0.0009	287	0.0005	295	0.0009	299	0.0016
42	302	0.0010	300	0.0008	302	0.0009	297	0.0012
43	298	0.0023	297	0.0013	299	0.0018	299	0.0024
44	272	0.0014	277	0.0008	271	0.0011	271	0.0012
45	337	0.0015	325	0.0016	337	0.0015	342	0.0023
46	252	0.0009	248	0.0007	252	0.0008	252	0.0009
47	330	0.0018	325	0.0009	330	0.0011	330	0.0013
48	300	0.0015	302	0.0011	300	0.0015	307	0.0018
49	269	0.0017	265	0.0011	280	0.0016	276	0.0012
50	293	0.0020	294	0.0007	293	0.0019	293	0.0015
51	278	0.0010	282	0.0007	275	0.0011	275	0.0010
52	341	0.0016	351	0.0009	341	0.0016	341	0.0018
53	288	0.0016	286	0.0012	288	0.0017	294	0.0016
54	273	0.0008	267	0.0007	273	0.0008	273	0.0008
55	307	0.0022	304	0.0005	307	0.0019	307	0.0018
56	289	0.0008	289	0.0005	289	0.0009	311	0.0011
57	257	0.0015	260	0.0018	257	0.0015	260	0.0017
58	341	0.0013	347	0.0010	341	0.0013	341	0.0015
59	293	0.0021	288	0.0011	293	0.0019	293	0.0022
60	259	0.0011	257	0.0010	259	0.0010	261	0.0014
61	307	0.0019	309	0.0007	307	0.0018	318	0.0021
62	289	0.0016	309	0.0010	289	0.0016	289	0.0017
63	275	0.0015	286	0.0008	275	0.0017	274	0.0015
64	297	0.0018	290	0.0008	297	0.0018	311	0.0013
65	272	0.0011	275	0.0006	272	0.0009	269	0.0011
66	294	0.0016	296	0.0011	294	0.0014	290	0.0019
67	325	0.0016	320	0.0012	325	0.0013	325	0.0017
68	301	0.0018	302	0.0009	301	0.0015	302	0.0019
69	309	0.0019	307	0.0010	309	0.0016	300	0.0023
70	273	0.0012	280	0.0008	273	0.0009	291	0.0011
71	295	0.0012	298	0.0008	295	0.0011	301	0.0012
72	259	0.0009	259	0.0008	259	0.0008	259	0.0010
73	278	0.0018	281	0.0009	278	0.0015	281	0.0022
74	271	0.0009	277	0.0010	271	0.0012	271	0.0014
75	216	0.0007	212	0.0006	212	0.0006	212	0.0009
76	300	0.0010	297	0.0008	300	0.0010	300	0.0011
77	275	0.0017	281	0.0006	275	0.0017	274	0.0011
78	272	0.0014	258	0.0010	272	0.0014	263	0.0020
79	281	0.0013	282	0.0009	281	0.0013	286	0.0023
80	283	0.0012	280	0.0008	283	0.0013	281	0.0015
81	273	0.0011	271	0.0008	273	0.0011	259	0.0015
82	361	0.0015	347	0.0010	361	0.0015	345	0.0019
83	259	0.0018	261	0.0009	259	0.0017	259	0.0017
84	274	0.0009	275	0.0007	274	0.0008	272	0.0011
85	300	0.0012	306	0.0010	300	0.0012	302	0.0013
86	341	0.0012	338	0.0006	341	0.0012	343	0.0014
87	275	0.0009	261	0.0009	275	0.0009	270	0.0012
88	312	0.0010	311	0.0010	312	0.0010	305	0.0017
89	293	0.0011	288	0.0008	293	0.0011	297	0.0012
90	329	0.0013	322	0.0011	329	0.0013	324	0.0033
91	290	0.0018	281	0.0012	290	0.0016	290	0.0013
92	323	0.0015	300	0.0006	323	0.0015	319	0.0014
93	318	0.0023	309	0.0012	318	0.0019	318	0.0016
94	311	0.0018	309	0.0011	311	0.0015	312	0.0015
95	262	0.0018	265	0.0009	260	0.0013	260	0.0013
96	280	0.0013	268	0.0009	280	0.0011	277	0.0020
97	256	0.0008	266	0.0005	256	0.0011	256	0.0013
98	320	0.0011	298	0.0007	320	0.0009	314	0.0014
99	287	0.0020	278	0.0014	287	0.0017	281	0.0021
100	258	0.0015	261	0.0007	249	0.0009	249	0.0007

Table B.12: Results of the four re-implemented algorithms of (Mosayebi et al., 2021) on the instances of Set III.

Instance	Pro.I		Pro.II		Pro.III		Pro.IV	
	f_{best}	t_{best}	f_{best}	t_{best}	f_{best}	t_{best}	f_{best}	t_{best}
1	3253	5.22	3235	0.71	3275	3.14	3257	2.19
2	3525	2.66	3494	0.57	3525	1.88	3525	1.69
3	3467	3.59	3477	0.80	3485	1.83	3481	2.23
4	3480	1.39	3486	0.68	3480	1.37	3465	4.44
5	3726	2.82	3673	0.66	3726	1.84	3700	3.21
6	3289	2.16	3273	0.61	3289	1.93	3284	1.59
7	3614	10.17	3571	0.93	3601	8.00	3601	6.79
8	3638	3.59	3604	0.68	3638	2.41	3635	6.16
9	3372	3.25	3335	0.76	3372	2.30	3373	1.77
10	3588	9.20	3544	0.89	3588	8.57	3551	3.98
11	3421	4.38	3401	0.73	3421	4.00	3421	3.86
12	3510	1.88	3521	0.43	3510	1.78	3555	3.23

13	3586	3.69	3523	1.14	3586	3.18	3550	2.86
14	3498	2.21	3480	0.42	3494	2.04	3494	1.79
15	3472	3.38	3484	0.85	3462	2.04	3493	2.50
16	3699	7.22	3725	1.00	3699	6.72	3723	6.76
17	3579	6.45	3588	0.76	3579	4.62	3571	7.89
18	3619	2.08	3593	0.99	3619	1.91	3658	3.64
19	3534	3.48	3530	0.62	3534	2.48	3501	3.68
20	3510	2.16	3504	0.72	3510	1.51	3524	7.37
21	3611	5.03	3589	0.50	3590	5.93	3628	4.10
22	3695	7.47	3714	1.15	3695	6.96	3685	9.75
23	3412	1.66	3416	0.38	3398	2.66	3401	1.35
24	3476	2.41	3481	0.99	3476	1.53	3491	1.95
25	3671	7.15	3676	0.99	3671	6.97	3671	4.75
26	3551	3.30	3477	0.73	3551	3.13	3551	2.81
27	3601	1.91	3576	0.90	3601	1.95	3606	5.00
28	3562	1.59	3528	0.73	3562	1.52	3505	0.82
29	3365	1.85	3327	0.53	3365	1.74	3376	2.66
30	3255	1.69	3276	0.67	3255	1.60	3256	1.48
31	3335	2.82	3324	0.37	3335	2.02	3346	2.98
32	3605	2.27	3559	0.91	3605	1.51	3608	2.91
33	3703	3.15	3667	0.96	3703	2.24	3681	1.51
34	3523	4.95	3502	0.48	3492	5.32	3492	4.05
35	3507	1.96	3509	0.48	3507	1.36	3505	1.87
36	3523	2.07	3467	0.40	3523	1.39	3532	3.06
37	3663	4.94	3668	0.98	3663	4.61	3679	8.39
38	3643	4.05	3621	0.91	3643	3.84	3660	6.25
39	3435	3.84	3432	0.53	3435	3.58	3435	3.24
40	3297	2.51	3293	0.35	3309	2.02	3352	2.41
41	3473	2.13	3465	0.80	3473	2.07	3486	2.50
42	3527	2.65	3506	0.79	3527	2.39	3516	7.20
43	3434	1.97	3415	0.71	3434	1.95	3406	1.68
44	3616	6.17	3634	0.82	3616	5.87	3648	4.77
45	3599	1.76	3612	0.80	3599	2.26	3587	2.61
46	3431	1.44	3388	0.61	3431	0.93	3386	2.27
47	3373	1.64	3335	0.69	3373	1.11	3342	4.50
48	3620	2.01	3558	0.78	3617	1.78	3605	2.07
49	3613	4.83	3602	0.83	3613	3.42	3613	3.19
50	3608	4.39	3601	0.54	3608	4.25	3594	8.24
51	3409	2.15	3373	0.52	3409	2.05	3409	1.72
52	3346	2.53	3317	0.67	3346	2.20	3328	0.94
53	3543	4.12	3528	0.44	3543	3.79	3538	3.55
54	3368	7.44	3382	0.75	3368	6.64	3375	5.09
55	3801	7.30	3739	0.57	3801	7.39	3801	5.69
56	3709	2.68	3682	0.51	3709	2.50	3749	2.47
57	3709	8.86	3748	1.08	3709	7.97	3709	7.71
58	3658	3.77	3633	0.63	3650	3.36	3661	4.94
59	3338	4.43	3302	0.40	3330	2.93	3338	1.96
60	3392	3.37	3404	0.42	3392	2.53	3405	2.13
61	3422	5.27	3422	0.47	3421	3.92	3435	2.66
62	3641	2.59	3609	0.74	3630	3.70	3625	3.68
63	3570	8.53	3558	0.51	3570	8.27	3583	7.63
64	3470	2.03	3470	0.48	3470	1.93	3472	2.92
65	3758	1.82	3732	1.01	3758	3.26	3758	2.95
66	3650	4.22	3661	0.78	3645	4.25	3660	3.30
67	3427	2.38	3460	0.65	3427	2.38	3427	2.18
68	3343	2.11	3375	0.80	3350	1.37	3350	1.14
69	3658	5.09	3641	0.56	3658	3.65	3658	3.01
70	3532	4.20	3520	0.89	3532	3.94	3532	3.38
71	3600	5.06	3599	0.76	3600	4.75	3633	4.53
72	3627	10.32	3630	1.06	3627	9.01	3623	5.24
73	3413	4.51	3428	0.40	3413	4.16	3413	3.35
74	3330	4.14	3281	0.60	3330	3.79	3330	3.34
75	3390	3.65	3416	0.69	3390	2.66	3424	1.88
76	3662	3.63	3645	0.85	3662	2.57	3657	4.65
77	3495	3.23	3472	0.56	3495	3.04	3478	2.98
78	3490	4.22	3480	0.51	3490	3.12	3444	3.48
79	3695	1.55	3676	1.07	3695	2.78	3680	2.80
80	3609	2.34	3554	0.76	3609	2.13	3561	3.18
81	3456	1.51	3446	0.32	3456	1.48	3437	3.66
82	3475	1.92	3482	0.77	3475	1.87	3475	1.84
83	3433	2.52	3428	0.73	3433	2.36	3443	3.66
84	3567	4.36	3588	0.47	3567	3.18	3570	8.63
85	3514	3.77	3541	0.77	3514	2.71	3508	4.96
86	3588	1.54	3595	0.36	3588	0.97	3611	3.14
87	3589	4.27	3541	0.62	3589	3.08	3554	9.34
88	3374	3.48	3386	0.66	3374	3.45	3396	4.76
89	3451	2.27	3442	0.65	3451	2.14	3459	1.92
90	3876	3.57	3839	1.14	3876	3.70	3895	2.34
91	3470	1.70	3502	0.60	3470	2.22	3448	2.46
92	3559	3.64	3522	0.81	3559	2.68	3531	4.55
93	3316	3.90	3319	0.58	3315	2.13	3315	2.53
94	3486	1.21	3495	0.62	3486	0.78	3493	3.34
95	3645	4.54	3629	0.89	3645	4.22	3645	4.34
96	3458	1.65	3441	0.63	3458	1.55	3458	1.39
97	3676	3.34	3646	1.04	3676	3.20	3637	5.41
98	3311	2.80	3325	0.70	3311	2.61	3354	2.25
99	3752	5.11	3752	0.87	3752	8.32	3746	1.92
100	3510	5.33	3526	0.97	3510	5.12	3523	4.01

Table B.13: The results of the four re-implemented algorithms of (Mosayebi et al., 2021) on the instances of Set IV.

Instance	Pro.I		Pro.II		Pro.III		Pro.IV	
	f_{best}	t_{best}	f_{best}	t_{best}	f_{best}	t_{best}	f_{best}	t_{best}
1	13980	112.97	13946	8.61	13982	119.20	13982	110.52
2	14542	186.77	14467	18.05	14434	87.97	14434	78.37
3	13441	61.15	13313	12.45	13450	68.74	13366	46.72
4	13696	61.44	13690	13.80	13690	80.70	13682	115.70
5	13901	109.25	13835	16.46	13901	111.60	13937	168.09
6	14286	142.60	14204	16.16	14286	88.96	14250	29.02
7	13745	49.29	13753	13.17	13745	35.94	13745	30.62
8	13855	180.92	13831	13.34	13855	137.84	13858	130.55
9	14134	143.34	14097	15.98	14016	226.50	14117	161.14
10	13736	60.12	13754	12.06	13736	43.62	13762	117.78
11	13282	42.88	13192	9.78	13282	40.61	13296	18.58
12	13610	177.31	13563	14.71	13698	115.34	13647	91.19
13	13257	69.66	13149	11.79	13257	62.79	13193	84.14
14	14003	62.32	14034	15.42	14003	72.62	14013	143.03
15	13730	50.69	13619	12.19	13730	46.23	13709	55.82
16	13309	43.21	13356	10.43	13309	38.51	13372	38.96
17	13310	26.48	13273	12.02	13323	20.85	13296	54.96
18	13761	55.35	13677	13.04	13761	49.35	13857	116.39
19	14143	145.96	13978	16.26	14108	53.59	14123	126.20
20	14134	88.64	14095	14.55	14134	86.17	14152	69.08
21	13566	48.70	13535	8.21	13566	47.47	13569	184.87
22	14141	61.07	13971	13.50	14141	60.34	14137	118.61
23	13392	52.99	13393	12.50	13392	49.91	13468	55.71
24	14161	136.83	13991	16.06	14142	119.77	14108	155.87
25	14381	74.53	14261	19.24	14379	71.52	14360	51.34
26	14206	67.36	14068	8.44	14206	66.57	14023	350.21
27	13245	58.94	13296	9.47	13245	54.93	13240	60.40
28	13757	57.58	13698	12.06	13707	43.05	13709	47.04
29	13528	86.76	13535	12.45	13528	85.30	13462	100.85
30	13759	93.67	13656	12.01	13759	90.48	13608	67.34
31	13771	84.10	13799	6.32	13771	81.20	13792	122.84
32	13889	157.23	13745	15.77	13873	150.20	13859	137.10
33	13843	103.04	13709	13.27	13843	98.40	13843	86.08
34	14404	36.25	14352	17.80	14404	35.48	14403	186.12
35	12885	146.16	12854	9.98	12885	146.65	12912	124.49
36	13940	32.36	13869	7.05	13940	30.06	13906	169.63
37	13862	64.51	13729	12.83	13862	63.53	13895	67.03
38	13850	82.97	13775	16.77	13850	80.65	13837	124.29
39	14324	199.25	14243	9.98	14324	191.65	14324	160.37
40	14038	45.76	13962	17.01	14038	43.80	13979	75.72
41	13629	144.78	13463	8.67	13629	135.82	13628	122.49
42	13974	63.52	13947	14.55	13974	61.07	13917	207.20
43	13698	91.51	13724	12.66	13698	87.37	13768	35.62
44	12991	103.86	12963	10.09	12987	97.94	13076	22.98
45	13253	123.30	13302	10.49	13253	115.81	13222	95.02
46	13950	99.24	13883	14.89	13950	94.75	13950	84.63
47	14748	43.61	14677	9.78	14748	44.40	14709	69.29
48	14142	66.63	14090	18.41	14142	66.19	14162	95.22
49	14341	75.18	14340	10.19	14341	71.03	14418	62.71
50	13987	89.82	13896	8.95	13987	84.89	13945	47.70
51	14157	83.48	14158	16.57	14157	81.57	14198	63.87
52	13892	229.90	13956	16.96	13892	217.55	14004	235.72
53	13925	243.13	13992	16.42	13925	216.86	13917	88.71
54	13753	193.36	13879	12.80	13753	178.52	13731	151.36
55	13840	59.93	13776	8.72	13840	55.43	13842	54.29
56	13846	129.60	13840	6.61	13846	95.87	13803	160.76
57	14015	35.61	13952	15.15	14015	32.60	13881	124.94
58	14550	128.35	14441	15.89	14550	116.80	14371	128.47
59	13742	54.31	13669	6.55	13742	50.29	13825	73.69
60	13835	88.57	13799	7.51	13809	134.73	13893	109.28
61	13680	55.56	13584	11.78	13680	57.03	13613	29.10
62	13641	200.77	13656	12.02	13707	127.78	13678	122.81
63	13948	47.70	13923	15.61	13948	45.79	14025	132.62
64	13299	81.65	13190	11.63	13299	60.08	13176	89.47
65	13773	56.59	13719	8.03	13773	54.44	13824	134.28
66	14553	326.79	14478	9.74	14553	313.50	14553	277.38
67	13617	37.20	13642	7.58	13597	39.40	13639	123.22
68	13530	32.35	13468	5.89	13441	61.90	13494	80.73
69	13702	70.72	13792	13.92	13702	67.02	13667	41.69
70	13826	88.44	13769	16.39	13826	81.64	13800	84.63
71	13565	197.43	13506	11.19	13565	190.37	13579	180.11
72	13888	80.25	13859	7.99	13873	78.43	13896	86.59
73	14343	98.04	14200	13.56	14343	91.99	14165	79.80
74	14202	182.64	14235	14.40	14189	133.59	14259	121.24
75	13792	46.84	13610	14.23	13792	44.38	13792	95.29
76	14063	149.41	13982	15.76	14063	138.40	14063	121.26
77	13386	29.01	13331	5.86	13386	26.89	13467	59.70
78	14263	120.30	14184	9.02	14157	100.31	14160	92.35
79	13779	104.36	13644	8.09	13779	132.84	13712	23.01
80	13614	184.91	13717	13.65	13606	164.04	13606	150.70

81	13155	74.37	13109	11.71	13155	55.86	13195	65.91
82	13986	35.53	13979	15.17	13986	24.29	13986	27.46
83	13263	122.13	13243	11.88	13263	114.18	13240	67.73
84	13754	33.23	13710	7.77	13754	32.88	13715	138.62
85	13301	48.17	13194	6.26	13301	43.81	13273	138.34
86	14483	110.51	14443	8.14	14483	103.65	14544	152.88
87	13122	91.57	13150	11.74	13116	53.45	13115	59.96
88	13658	47.88	13559	6.87	13658	44.86	13584	127.06
89	13460	26.10	13497	11.40	13460	25.84	13595	112.50
90	14156	52.33	14117	12.97	14232	25.71	14214	23.33
91	14126	140.15	14101	17.48	14126	144.31	14093	170.07
92	13747	70.04	13648	15.04	13705	66.87	13686	80.52
93	13823	28.77	13764	13.44	13823	27.64	13768	142.08
94	13927	81.01	13889	13.85	13927	79.25	13916	66.63
95	13927	119.82	13888	13.40	13927	113.75	14017	180.84
96	13540	81.05	13575	11.30	13540	104.72	13540	94.26
97	13247	29.07	13196	12.12	13180	128.84	13182	93.69
98	13663	91.97	13618	11.00	13606	101.99	13606	91.88
99	15056	226.23	14919	18.12	15014	179.76	14960	282.83
100	13236	161.45	13242	10.13	13236	148.62	13330	134.05