Mixed Neighbourhood Local Search for Customer Order Scheduling Problem

Vahid Riahi, M M A Polash, M A Hakim Newton, and Abdul Sattar

Institute for Integrated and Intelligent Systems (IIIS)
Griffith University, Australia
{vahid.riahi,mdmasbaulalam.polash}@griffithuni.edu.au
{mahakim.newton,a.sattar}@griffith.edu.au

Abstract. Customer Order Scheduling Problem (COSP) is an NP-Hard problem that has important practical applications e.g., in the paper industry and the pharmaceutical industry. The existing algorithms to solve COSP still either find low quality solutions or scramble with large-sized problems. In this paper, we propose a new constructive heuristic called *repair-based mechanism (RBM)* that outperforms the best-known heuristics in the literature. We also propose a mixed neighbourhood local search (MNLS) algorithm. MNLS embeds a number of move operators to diversify the local exploitation making different areas around the current solution accessible. Moreover, we also propose a greedy diversification method to keep the search focussed even when it is in a plateau. Our experimental results on 960 well-known problem instances indicate statistically significant improvement obtained by the proposed MNLS over existing state-of-the-art algorithms. MNLS has found new best solutions for 721 out of the 960 problem instances.

Keywords: Scheduling · Customer Order · Local Search.

1 Introduction

A customer order scheduling problem (COSP) has n customer orders and m parallel machines. Each customer order j contains m items. Each item i can be processed by a particular machine i. Machines can execute different items of a customer order j simultaneously. Each item i of a customer order j needs a non-negative period of processing time $p_{ij} \geq 0$ at machine i. The completion time of a customer order j is the time point when processing of all corresponding items is finished. The aim is to find a sequence of the customer orders such that the summation of the completion times of the customer orders is minimised.

The COSP problem was first introduced by [2]. It has several real-life applications, e.g., car repair shops [10], pharmaceutical industry [5], manufacturing of semi finished lenses [1], and paper industry [6]. Consider a car repair shop that possesses several particular mechanics. Each entering car contains multiple broken parts. Each broken part needs a particular mechanic to fix it. The mechanics are able to work on a single car at the same time. A car leaves the shop

only when all broken parts are repaired. This car repair shop model can also be adapted to aircraft maintenance and ship repair [10].

The COSP is an NP-Hard problem [6], meaning that it is difficult to be optimally solved especially when the size of the problem is large. Therefore, researchers recently put much more attention on heuristic and metaheuristic algorithms to solve COSP. The very first heuristic proposed for COSP is *Shortest Total Processing Time (STPT)*[8]. Recently, two other heuristics are proposed based on the idea of STPT: *Earliest Completion Time (ECT)* [6] and FP [3] (named after the authors' names). As metaheuristics, a Tabu Search (TS) [6] and a Greedy Search Algorithm (GSA) [3] are proposed.

In this paper, we study possible drawbacks of existing methods (explained in Section 3) and then propose efficient approaches, which outperform the state-of-the-art methods. First, we propose a constructive heuristic called *repair-based mechanism (RBM)*. RBM is an improved version of STPT heuristic. STPT starts with an empty sequence. It then iteratively appends the unscheduled customer order with the smallest total processing time at the end of the partial sequence. However, based on the repair mechanism, RBM seeks better positions for unscheduled customer orders by inserting them to earlier positions of the partial sequence as well instead of only at the end.

As the metaheuristics, we propose a Mixed Neighbourhood Local Search (MNLS) algorithm. MNLS is made up of two main steps: intensification and diversification. In the intensification phase, we use a number of neighbourhood operators instead of only one. The idea behind this is that different neighbourhoods will generate different local optima and different landscapes [9]. Therefore, different neighbourhoods will help our algorithm search different regions around the current solution. In the diversification phase, we use the idea of a multi-start procedure that produces different initial solutions from where local search can start. In the classical multi-start procedure, the initial solutions are completely randomly generated; which typically cause exploration of inferior solutions in terms of the objective value [9]. We use a greedy procedure to generate initial solutions by keeping a part of the current local optima and changing another part of it. The intuition is that the proposed greedy approach will create a solution that shares the properties of both new and good areas of the search space in terms of the objective function.

The proposed RBM and MNLS algorithms are tested on 960 well-known large instances generated by [3]. First, our results showed the superiority of the proposed RBM against STPT as well as FP heuristic (the best-preforming heuristic in the literature) with a substantial margin. Experimental results showed that the proposed MNLS algorithm with RBM and even with FP as initialisation significantly outperform GSA, the best existing metaheuristic in the literature. Finally, for 721 out of the 960, a new upper bound is found by the proposed MNLS.

In the rest of the paper, Section 2 outlines the formulation of COSP, Section 3 describes the literature, Section 4 introduces the constructive heuristic

and MNLS algorithm, Section 5 presents the experimental results, and Section 6 concludes the paper.

2 Preliminaries

The COSP with n customer orders and m machines has $n!$ possible permutations on each machine and a total of $(n!)^m$ possible permutations. However, [4] has proved that the optimal case happens when the permutation of the customer orders is the same for all machines.

Suppose π be a permutation of the given n customer orders and π_k denotes the customer order in the kth position of π. Also, let C_{i,π_k} be the completion time of the customer order at position k on machine i. To calculate the total completion time TCT(π) of solution π, first, the completion time of each customer j order on each machine i is calculated as $C_{i,\pi_k} = C_{i,\pi_{k-1}} + p_{i\pi_k}$ where $k = 1, 2, ..., n$, $i = 1, 2, ..., m$, and for convenience of the computation, $C_{i,[0]} = 0$. Then, the completion time of customer order occupied in position k is computed $C_{\pi_k} = max\{C_{i,\pi_k}\}$. Finally, $TCT(\pi) = \sum_{k=1}^{n} C_{\pi_k}$ computes the total completion time of solution π.

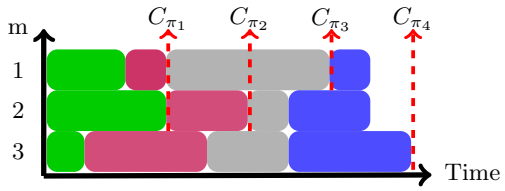

Fig. 1: A COSP with three machines and four orders.

In Fig 1, a COSP with three machines and four customer orders is depicted to help understand the problem. The total completion times are computed by summing up the four completion times shown in red arrows.

3 Related Works

To solve COSP, a heuristic named *Shortest Total Processing Time (STPT)* is proposed in [8]. STPT starts with an empty sequence and constructs a schedule by appending the unscheduled customer order with the smallest total processing time at the end of the partial sequence. This heuristic is very fast. However, the performance significantly decreases especially when the problem size increases since it only focuses on the end of the partial solution. Another existing heuristic is *Earliest Completion Time (ECT)* [6]. In ECT, all unscheduled customer orders are appended one by one at the end of the partial sequence, and the permutation with lowest objective value is picked. This process continues until all customer orders are scheduled. Another heuristic called FP [3] is also proposed. FP starts with an initial sequence obtained by another heuristic called SPT-B. The SPT-B first obtains m permutations by sorting the customer orders in a non-decreasing order of the processing times on each machine. After calculating the objective value of each m sequences, the best one is selected as the initial

sequence of FP. Next, similar to ECT, customer orders are tested on the end of the partial sequence. However, FP always considers a complete sequence considering those unscheduled customer orders based on the STP-B ordering as the trailing customer orders of the sequence. Focusing only again on the end of partial sequence leads to performance degradation for both FP and ECT especially when problem size increases.

As metaheuristics, a Tabu Search (TS) [6] and a Greedy Search Algorithm (GSA) [3] are proposed. TS uses ECT to generate an initial solution and then uses the swap operator in local search. GSA employs FP for the initial solution. Then, it goes through the main loop which contains three steps. At the first step, one customer order is randomly selected and inserted at the end of the solution. At the second step, GSA finds the best neighbour for the customer order newly placed at the end, and finally applies an exhaustive swap operator on the solution obtained from the second step. It is continued until the stopping criterion is met. The results show the efficiency of GSA over TS algorithm. Both algorithms exhaustively uses all possible swap moves; which increases the required CPU times as well as the risk of getting stuck in the local optima.

4 Our Approach

In order to solve COSP, we propose a heuristic called *repair-based mechanism (RBM)* and a metaheuristic algorithm named Mixed Neighbourhood Local Search (MNLS). The proposed algorithms are explained in the following sections.

4.1 The Proposed RBM Heuristic

RBM is an improved version of $STPT$ heuristic [8]. $STPT$ starts with an empty sequence π, calculates the total processing time $\sigma[j] = \sum_{i=1}^{m} P_{ij}$ of each order j over all machines, sorts them in a list L in a non-decreasing order of $\sigma[j]$. At each iteration $1 \leq k \leq n$, it appends the kth customer order from the list L at the end of the sequence π.

One possible criticism of $STPT$, ECT and FP is that they are position-oriented method as their focus is to find the best customer order just for the end of the current partial sequence, and the positions of previously scheduled customer orders are not changed any more as the construction phase progresses.

We propose a heuristic to modify the above-mentioned drawback and improve the performance. RBM is an order-oriented method seeking the 'best' position for each customer order. To do this, the current unscheduled customer order is inserted not only at the end of the current partial solution, but also inserted at the beginning and all other positions of the partial sequence. In addition, when a customer order is at its best position, a new partial sequence would be created. The already scheduled customer orders may then find better positions in the new partial sequence. Thus, reinserting those already scheduled customer orders may be useful as well.

In the proposed RBM shown in Algorithm 1, in each iteration k, the current unscheduled customer order $L[k]$ is inserted at the last position $k' = k$ of the

partial sequence π, and all other possible positions $1 \leq k' < k$ of the partial sequence. Then, the position with the lowest total order completion time is selected as the position $\bar{k}$ of the customer order currently being placed. After finding the best position for the current customer order $L[k]$ in the partial sequence π, we again repair the π focusing on the neighbours of the newly placed customer order. Two customer orders around position $\bar{k}$, $h = \bar{k} \pm 1$ ($h \geq 1$ or $h \leq n$), are also inserted in all positions of the partial sequence π. Note that we select two orders around position $\bar{k}$ for the repair process since they were adjacent before inserting the current order $L[k]$ at position $\bar{k}$, so they are highly likely to be affected than others. It is also worth mentioning that although our proposed heuristic requires slightly more CPU times, it obtains remarkably better results.

Algorithm 1: RBM Heuristic

```
1  σ[j] = Σ_{i=1}^{m} P_ij    for each customer order j (j = 1, 2, ..., n)
2  Sort the customer orders in the list L in a non-decreasing order of σ[j].
3  Let π be a partial sequence with only one customer order, π_1 ← L[1].
4  for k = 2 to n do
5      π[k] ← L[k]    // The STPT heuristic
6      // The proposed repair-based mechanism
7      k̄ ← k, BestObj ← TCT(π), π' ← π
8      for k' = k − 1 to 1 do
9          π'' ← Insert customer order π'_k in the position k' in π'.
10         if TCT(π'') < BestObj then
11             k̄ ← k', BestObj ← TCT(π''), π ← π''
12     π ← Sequence with lowest TCT(π) by inserting customer order π_h in all
         possible positions of π, where h = k̄ ± 1 (h ≥ 1 or h ≤ k)
13 return π
```

4.2 Search Algorithm

There exist some algorithms to solve the COSP, however those algorithms either find low quality solutions or struggle with large-sized problems. One reason is that they get stuck in the local optima because of an exhaustive use of a move operator or a weak diversification method. In order to deal with these issues, we propose an efficient local search-based algorithm, called MNLS that embeds a number of neighbourhood operators in a mixed fashion as intensification process and also a greedy diversification method as the diversification approach. The use of mixed neighbourhood operator creates different local optima and allows our algorithm to explore more search areas and diversify the intensification process. Besides, in the diversification phase, we replace the typical random method with a greedy approach that produces a new solution by keeping a part of the local optima and changing the remaining part of it, i.e., intensify the diversification process. The greedy diversification, allows our algorithm to share the information

of both good and new areas in the search space at the time of generating new solution for the intensification procedure.

The proposed search algorithm given in Algorithm 2 starts with an initial solution obtained by RBM heuristic. The initial solution is then improved by the proposed mixed neighbourhood search. Then, the algorithm goes through the loop in which the search restarts with a greedy method followed by mixed neighbourhood search. To escape from strong local optima, an acceptance criterion is also used to decide whether to accept or reject the solution obtained by the intensification phase.

Algorithm 2: Proposed Search algorithm

```
1  π ← Generate an initial solution using the RBM heuristic
2  π ← Use the intensification method on π
3  iter ← 1, π_best ← π
4  while ++iter ≤ MaxIter do
5      π' ← Use the diversification method on π
6      π'' ← Use the intensification method on π'
7      if TCT(π'') < TCT(π) then
8          π ← π''
9          if TCT(π'') < TCT(π_best) then
10             π_best ← π''
11     else if random() < P then
12         π ← π''
13 return π_best
```

Intensification Method In this paper, we propose a Mixed Neighbourhood Local Search (MNLS) method that contains four move operators $N_k(k = 1, .., 4)$. In MNLS, instead of a single neighbourhood, we explore a number of given neighbourhoods to obtain a better solution with respect to the objective value. The intuition is that different neighbourhoods create different landscapes and consequently different local optima. The use of mixed neighbourhoods thus enables our algorithm to navigate through the search area, to explore more regions around the current solution, and also to escape from local optima.

The selection of neighbourhood operators is important and is considerably affects the performance of MNLS. Among several types of operator, we select Insert, Swap, Insert-Pair, and Swap-Pair since these moves are widely used when solutions are permutations. These operators are as follows:

1. Insert (N_1): A random customer order π_j is removed from its position j and then reinserted at a random position k ($k \neq j$) (Figure 2.a).
2. Swap (N_2): Two random customer orders, π_j and π_k ($k \neq j$), are selected and their positions are exchanged (Figure 2.b).

3. Insert-Pair (N_3): Two consecutive random customer orders, π_j and π_{j+1} are removed from their positions, j and $j+1$ ($j = 1, 2, ..., n-1$), and reinserted into random positions k and $k+1$ ($k = 1, 2, ..., n-1$ and $k > j+1$ or $k+1 < j$) (Figure 2.c).
4. Swap-Pair (N_4): Two consecutive random customer orders, π_j and π_{j+1} ($j = 1, 2, ..., n-1$) are selected and their positions are exchanged with two different consecutive random customer orders, π_k and π_{k+1} ($k = 1, 2, ..., n-1$ and $k > j+1$ or $k+1 < j$) (Figure 2.d).

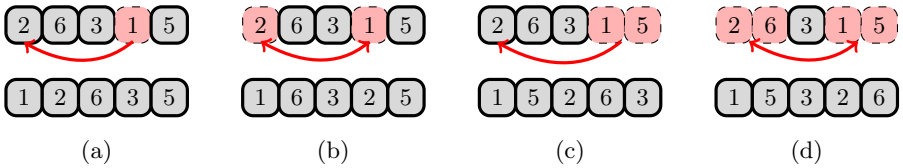

Fig. 2: Example of operators **a)** Insert **b)** Swap **c)** Insert-Pair **d)** Swap-Pair

Recall that one possible drawback of existing metaheuristic algorithms is the exhaustive use of a single operator. Evaluating all neighbouring solutions would be hugely time-consuming particularly for the large instances. For example, an exhaustive use of only Swap operator generates a total of $n(n-1)/2$ candidate solutions. To tackle this drawback, in this paper we employ two approaches: an *acceleration method* (a fast neighbourhood evaluation strategy) and a random-based operator selection.

The *acceleration method* is based on the one proposed by [7]. The idea is to compute only the completion time of the changed part of the permutation after employing each move, and reuse the computation of the unchanged part of the permutation. With this *acceleration method*, we can speed up the calculation of the total customer orders completion time to 50% that is notable. Based on the second approach, the random-based operator selection, MNLS selects an operator at each iteration randomly. This procedure gives our four operators the same chance to be applied on the current solution. This procedure, compared to exhaustive operators, keeps open the search space but saves the computation time remarkably.

In the proposed intensification method given in Algorithm 3, at each iteration, one of the operators is randomly selected and applied to the current solution. If a better solution in terms of total completion time is obtained, the current solution is updated and the iteration counter is reset to 1; otherwise the iteration counter is increased. This process is continued until the iteration counter is less than $20 \times n$ (it is set through experimentation).

Diversification Method To avoid getting stuck and convergence towards local optima, we propose a multi-start MNLS for COSP. In the classical multi-start

Algorithm 3: Proposed MNLS

```
Let π is input solution, and iter ← 1.
Set N1: Insert, N2: Swap, N3: Insert-Pair, and N4: Swap-Pair.
while ++iter ≤ 20n do
    k = random(1,4)        // Generate a random number between 1 and 4.
    π' ← Nk(π)             //Apply operator Nk on the current solution π.
    if TCT(π') < TCT(π) then
        π ← π', iter ← 1
return π
```

procedure, a solution at each iteration is randomly generated and most likely unrelated to the previously visited local optima. The randomly generated solution is then improved by the local search. However, [9] pointed out that 'the quality of the local optima obtained by a local search method depends on the initial solution' and a random solution in most cases is a low quality solution. In this paper, we propose a greedy diversification method that creates a new solution at each iteration relying on the preliminary idea of our RBM heuristic. The proposed greedy method holds a part of the current local optima and shuffles the remaining parts. The intuition is that we do not fully erase the information of the properties of the local optima and use a part of it for the next initial solution as good solutions often cluster around (e.g., in a range of mountains).

In the proposed diversification method, at each iteration, a set of customer orders, $ExtractedOrder = 2$ (it is fixed with a preliminary test), are randomly extracted and removed from the current solution π and saved into a list ω as the unscheduled customer orders. After that, the extracted customer orders, one by one, leaves the list ω and goes back to the end of the partial sequence π. Unlike the RBM heuristic, this newly placed customer order is then swapped with all other customer orders in the partial sequence π and its best position $\bar{k}$ is selected based on the objective. Then, its new neighbours (the customer orders at positions $p = \bar{k} \pm 1$ ($p \geq 1$ or $p \leq n$)) are inserted in all positions of the current partial sequence. The best solution with respect to the objective value would be considered as the partial sequence for the next unscheduled customer order.

Note that, using only one move in the diversification procedure may increase the risk of returning to a previously visited solutions at the time of using the intensification phase. Therefore, we use swap as well as insert operator to decrease this probability. Additionally, the proposed diversification method keeps some parts of the solution and destroys another part of the solution with the hope of obtaining a better search area. Consequently, although a new initial solution for intensification is created, it is not far away from the previously found local optima, and has some of their properties as well. This is why we propose a greedy diversification method for MNLS instead of a fully randomly generated solution. The proposed greedy diversification method is presented in Algorithm 4.

Algorithm 4: Proposed greedy diversification method

Input: π: Solution, $ExtractedOrder$: number of customer orders removed.
Set $\omega = \emptyset$.
for $k = 1$ *to* $ExtractedOrder$ **do**
    Remove a random customer order (without repetition) from π and save in ω
for $k = 1$ *to* $ExtractedOrder$ **do**
    $\bar{k} \leftarrow LastPos \leftarrow n - ExtractedOrder + k$
    $\pi[LastPos] \leftarrow \omega[k]$
    $BestObj \leftarrow TCT(\pi)$, $\pi' \leftarrow \pi$
    for $k' = LastPos - 1$ *down to* 1 **do**
        $\pi'' \leftarrow$ Swap the customer orders at positions $\pi'_{LastPos}$ and $\pi'_{k'}$
        if $TCT(\pi'') < BestObj$ **then**
            $\bar{k} \leftarrow k'$, $\pi \leftarrow \pi''$, $BestObj \leftarrow TCT(\pi'')$
    $\pi \leftarrow$ Schedule with lowest $TCT(\pi)$ by inserting customer orders π_p in all possible positions of π where $p = \bar{k} \pm 1$ ($p \geq 1$ or $p \leq LastPos$)
return π

Acceptance Criterion After the intensification phase, it should be decided whether to accept or reject the new solution obtained by the intensification phase as the current solution for the next iteration. Thus, we propose a simple acceptance criterion that accepts worse solutions with the probability of $P = 0.5$ (it is fixed through experimentation). This acceptance criterion helps the algorithm keep a balance between intensification and diversification. In terms of intensification, only improving solutions are acceptable (strong selection), however, based on the diversification, any solution with any quality with respect to the objective function is acceptable (weak selection).

5 Experimental Results

Experiments are carried out using Framinan's benchmark [3]. This benchmark is made up of 1680 instances: 720 small and 960 large instances. In this paper, we have used the 960 large instances. This instance set is divided into two testbeds (480 instances each): Test-1 and Test-2. In the former, each customer order requests m items with $p_{ij} > 0$, while in the latter, $p_{ij} \geq 0$ i.e., the processing times of some items are zero. Both these testbeds are categorized in 16 sets with 30 instances. The sets include different combinations of the numbers of customer orders $n \in \{20, 50, 100, 200\}$ and machines $m \in \{2, 5, 10, 20\}$.

FP and GSA [3] are respectively the best existing heuristic and metaheuristic in the literature. Thus, the proposed methods are compared with these algorithms. The programs are implemented in the programming language C and tested on the same computer. To evaluate the results, the deviation of total completion time of method A from the total completion time of the best-known solutions for each instance is calculated, $TCT(A^i)$ - $TCT(\pi^i)$. The smaller the deviation, the better the methods' performance. Note that we evaluate the idea of using mixed neighbourhood operators instead of a single one as well as a greedy diversification instead of a random one. The results confirm the efficiency of

the proposed idea of mixed neighbourhood operators and greedy diversification. However we do not show related results in detail due to space restriction.

Table 1 gives a comparison of RBM, FP and STPT heuristics. STPT is also included in this experiment since RBM is a modification of this heuristic. Table 1 shows that RBM significantly improves the performance of STPT heuristic in all instance groups. In addition, the proposed heuristic outperforms FP heuristic in 27 out of 32 instance sets. We also use a student *t-test* with level of significance $\alpha = 0.05$ between heuristics. On both Test-1 and Test-2 p-value were 0.000 ($< \alpha$) that confirm a statistically significant difference among FP and RBM heuristic.

Table 1: Deviation from best-known solutions for the heuristics.

Instances	TEST-1				TEST-2			
$n \times m$	**Best**	**STPT**	**FP**	**RBM**	**Best**	**STPT**	**FP**	**RBM**
20 × 2	8952	432	48	**24**	6298	462	65	**12**
20 × 5	10461	654	101	**78**	5385	716	86	**53**
20 × 10	11491	976	145	**138**	5805	862	109	**107**
20 × 20	12320	972	**138**	181	6231	821	105	**92**
50 × 2	51965	1968	223	**91**	35373	2594	269	**82**
50 × 5	58741	4105	643	**523**	30577	4008	601	**527**
50 × 10	63779	5638	**951**	986	30479	5048	703	**621**
50 × 20	66752	5842	**993**	1195	31916	5181	794	**733**
100 × 2	202005	6896	809	**379**	134113	8449	943	**243**
100 × 5	228602	15162	2519	**1910**	111054	12519	2568	**1381**
100 × 10	241551	18259	3395	**3302**	109886	15904	2778	**2286**
100 × 20	253401	20878	**3519**	4557	109940	16428	2654	**2596**
200 × 2	783719	21729	2718	**1330**	518665	20336	2888	**638**
200 × 5	884242	44777	10470	**5689**	433946	45810	11215	**4833**
200 × 10	935949	52661	10813	**9150**	403657	48816	10087	**7674**
200 × 20	979527	60983	**11521**	14127	407541	51613	8984	**8781**
average	-	16371	3063	**2729**	-	14973	2803	**1916**

We assess the efficiency of the proposed MNLS algorithm against GSA. GSA algorithm used FP heuristic as initialisation. To have a better view about the performance of the proposed mixed neighbourhood operators and the greedy diversification, we also use the FP heuristic as our initial solution (MNLS-FP for short). As the stopping criterion, a maximum number of iteration, *MaxIter = 100* is considered for all algorithms. However, two other values 10 and 50 are also considered for *MaxIter*to show the performance of the algorithms from short to long duration. The results are given in Table 2. To display that the differences between algorithms are statistically significant, the 95% confidence interval plot of them are also shown in Figure 3. No overlapping of intervals for a pair denotes a significant difference between the two methods compared.

From Table 2 and Figure 3, MNLS with *MaxIter = 100* (MNLS-100 for short) has the best performance in both of the testbeds on average, and statistically outperforms other algorithms. Additionally, not only MNLS-100, but MNLS-50 (MNLS with 50 iterations) is also statistically superior to GSA-100. Interestingly, MNLS-10 (with only 10 iterations) is statistically equivalent to

Table 2: Deviation from best-known solutions for the search algorithms.

TEST-1	Best	GSA	GSA	GSA	MNLS-FP	MNLS-FP	MNLS-FP	MNLS	MNLS	MNLS
n × m		10	50	100	10	50	100	10	50	100
20 × 2	8952	5	1	**0**	1	**0**	**0**	1	**0**	**0**
20 × 5	10461	15	6	4	5	1	**0**	5	1	**0**
20 × 10	11491	29	11	9	15	3	**0**	13	2	**0**
20 × 20	12320	30	12	9	22	3	1	19	3	**1**
50 × 2	51965	38	21	17	8	1	**0**	8	2	**0**
50 × 5	58741	153	86	59	69	20	8	44	13	**2**
50 × 10	63779	229	98	77	179	49	21	109	32	**7**
50 × 20	66752	273	126	93	244	56	11	147	40	**7**
100 × 2	202005	137	94	86	25	7	**2**	35	8	**2**
100 × 5	228602	449	246	229	267	89	28	169	42	**9**
100 × 10	241551	592	346	288	734	239	87	289	90	**11**
100 × 20	253401	751	296	197	1386	440	130	509	155	**17**
200 × 2	783719	299	241	227	94	23	7	90	20	**2**
200 × 5	884242	1563	872	746	884	287	110	371	98	**11**
200 × 10	935949	2548	907	664	3248	1088	552	597	136	**15**
200 × 20	979527	3651	715	290	6702	2887	1630	1467	347	**67**
average	299591	673	255	187	868	324	162	242	62	**9**
TEST-2	Best	GSA	GSA	GSA	MNLS-FP	MNLS-FP	MNLS-FP	MNLS	MNLS	MNLS
n × m		10	50	100	10	50	100	10	50	100
20 × 2	6298	4	2	1	**0**	**0**	**0**	**0**	**0**	**0**
20 × 5	5385	13	2	1	3	**0**	**0**	2	**0**	**0**
20 × 10	5805	13	2	1	9	1	**0**	5	1	**0**
20 × 20	6231	18	3	3	7	1	**0**	6	1	**0**
50 × 2	35373	37	18	13	6	1	**0**	6	1	**0**
50 × 5	30577	132	73	59	69	28	**18**	57	25	**18**
50 × 10	30479	185	96	77	120	45	30	97	40	**30**
50 × 20	31916	218	94	74	158	62	39	108	51	**33**
100 × 2	134113	127	72	66	23	9	**7**	30	10	**7**
100 × 5	111054	385	260	251	286	137	98	193	108	**77**
100 × 10	109886	519	326	294	539	241	172	337	173	**134**
100 × 20	109940	537	312	281	702	305	190	461	251	**174**
200 × 2	518665	254	220	198	63	30	22	82	33	**21**
200 × 5	433946	968	703	661	779	367	265	521	284	**214**
200 × 10	403657	1365	839	746	2067	891	578	928	546	**389**
200 × 20	407541	1523	759	545	3093	1366	791	1424	717	**465**
average	148804	394	236	204	495	218	138	266	140	**98**

GSA-100. Comparison of MNLS-FP and GSA also reveals some points that are worth to be mentioned. GSA-10 and GSA-50 have better performance compared to MNLS-FP-10 and MNLS-FP-50 respectively. However, MNLS-FP-100 has smaller average deviations than GSA-100 in 30 out of 32 instance groups. To have a better view, we present the convergence graphs of the algorithms for two instances: TEST-1-318 ($n = 200$ and $m = 10$) and TEST-2-179 ($n = 50$ and $m = 5$) in various iterations in Figure 4. It can be seen that GSA can find better solutions in the earlier stage of the search. However, when the algorithms proceed for more iterations, MNLS outperforms GSA. In fact, the proposed algorithm is able to jump out of the local optima later in the search. We do not show the convergence of other instances as they are similar.

In each iteration of the search, what MNLS does is not the same as what GSA does, e.g., MNLS uses repeatedly one of the given neighbourhood operator while GSA uses an exhaustive swap. Therefore, The CPU execution times of the

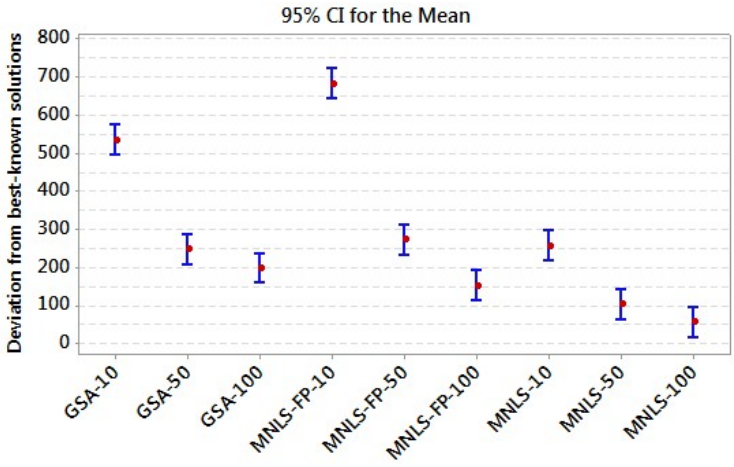

Fig. 3: 95% confidence interval plot of compared algorithms.

algorithms must be compared instead of considering the number of iterations. The CPU times used by the algorithms based on the number of orders and machines, respectively, are shown in Figure 5. This figure shows that MNLS is much faster than GSA but still obtains better results. Finally, it is worth noting that for TEST-1 and TEST-2, 379 and 349 new best-known solutions are obtained by MNLS respectively out of the 480 each.

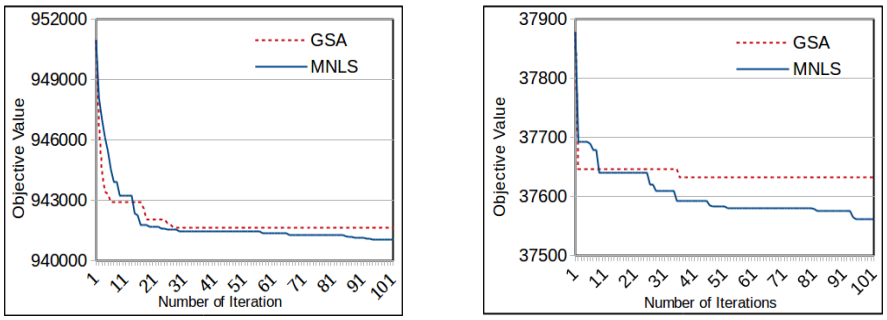

Fig. 4: Convergence of GSA and MNLS for TEST-1-318 (left), TEST-2-179 (right).

6 Conclusion

In this paper, we consider a Customer Order Scheduling Problem (COSP) that has realistic applications such as the paper industry and the pharmaceutical industries. The COSP is known to be NP-hard. To solve this problem, an effective

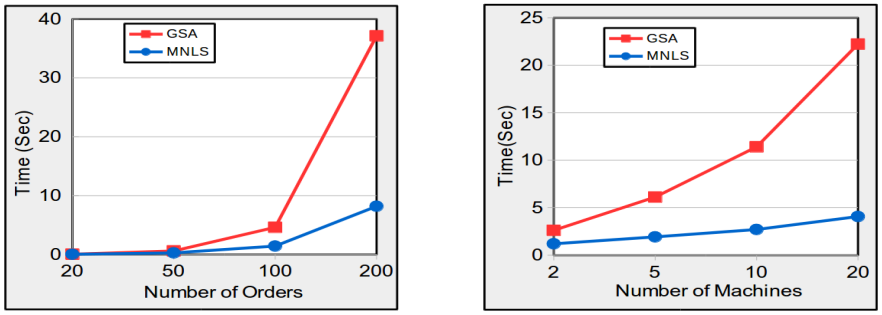

Fig. 5: Interaction between CPU times of algorithms and number of orders/machines.

local search algorithm is proposed that comprises a new constructive heuristic to generate the initial solution, a mixed neighbourhood for intensification, and also a greedy approach for diversification to keep the search focussed even when it is in a plateau. The results show that the proposed algorithm significantly outperforms the state-of-the-art algorithms. Moreover, the proposed algorithm has also found new upper bounds for 721 out of 960 problem instances.

References

1. Ahmadi, R., Bagchi, U., Roemer, T.A.: Coordinated scheduling of customer orders for quick response. Naval Research Logistics (NRL) **52**(6), 493–512 (2005)
2. Ahmadi, R., Bagchi, U.: Scheduling of multi-job customer orders in multi-machine environments. ORSA/TIMS, Philadelphia (1990)
3. Framinan, J.M., Perez-Gonzalez, P.: New approximate algorithms for the customer order scheduling problem with total completion time objective. Computers & Operations Research **78**, 181–192 (2017)
4. Lee, I.S.: Minimizing total tardiness for the order scheduling problem. International Journal of Production Economics **144**(1), 128–134 (2013)
5. Leung, J., Li, H., Pinedo, M.: Multidisciplinary scheduling: Theory and applications. Chapter Order Scheduling Models: an overview (2005)
6. Leung, J.Y.T., Li, H., Pinedo, M.: Order scheduling in an environment with dedicated resources in parallel. Journal of Scheduling **8**(5), 355–386 (2005)
7. Li, X., Wang, Q., Wu, C.: Efficient composite heuristics for total flowtime minimization in permutation flow shops. Omega **37**(1), 155–164 (2009)
8. Sung, C.S., Yoon, S.H.: Minimizing total weighted completion time at a preassembly stage composed of two feeding machines. International Journal of Production Economics **54**(3), 247–255 (1998)
9. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)
10. Yang, J.: Scheduling with batch objectives. Ph.D. thesis, The Ohio State University (1998)