International Journal of Autonomous and Adaptive Communications Systems

ISSN online: 1754-8640 - ISSN print: 1754-8632
https://www.inderscience.com/ijaacs

An improved salp swarm algorithm for collaborative scheduling of discrete manufacturing logistics with multiple depots

Huajun Chen, Yanguang Cai

DOI: 10.1504/IJAACS.2025.10059647

Article History:

Received:	27 July 2022
Last revised:	10 December 2022
Accepted:	13 March 2023
Published online:	04 February 2025

An improved salp swarm algorithm for collaborative scheduling of discrete manufacturing logistics with multiple depots

Huajun Chen and Yanguang Cai*

School of Automation,
Guangdong University of Technology,
Guangdong Province, 510006, China
Email: chj@mail2.gdut.edu.cn
Email: caiyg99@163.com
*Corresponding author

Abstract: Aiming at the situation of a single factory, multiple depots and multiple customers, considering storage, time windows and capacity constraints, a collaborative scheduling of discrete manufacturing logistics with multiple depots (CSDMLMD) model is established. This problem includes discrete manufacturing process, depot storage process and logistics transportation scheduling process. Based on the basic principle of salp swarm algorithm, an improved salp swarm algorithm (ISSA) is proposed to solve the CSDMLMD problem. It is compared with simulated annealing algorithm, genetic algorithm and particle swarm algorithm (PSO), and relatively good results can be obtained. The experimental results presented in this paper verify the feasibility of solving this problem.

Keywords: open shop scheduling; vehicle routing; discrete manufacturing; collaborative scheduling.

Reference to this paper should be made as follows: Chen, H. and Cai, Y. (2025) 'An improved salp swarm algorithm for collaborative scheduling of discrete manufacturing logistics with multiple depots', *Int. J. Autonomous and Adaptive Communications Systems*, Vol. 18, No. 1, pp.1–22.

Biographical notes: Huajun Chen is a PhD student of Guangdong University of Technology. His research interest is discrete manufacturing, logistics and intelligent algorithms.

Yanguang Cai received his Master of Science from Chongqing University in 1988 and his Doctor of Engineering from Zhejiang University in 1996. He is now a Professor and Doctoral Supervisor of Guangdong University of Technology. His research interests include complex network system modelling, control and optimisation, logistics control and optimisation, intelligent transportation system, combinatorial optimisation, intelligent optimisation, information processing and optimisation control of Internet of Things, etc.

1 Introduction

In this paper, the problem of collaborative scheduling of discrete manufacturing logistics with multiple depots (CSDMLMD) is studied. The problem can be briefly described as follows: the orders are processed in the factory, the orders are produced according to the open shop scheduling (OSS) problem, the completed orders are stored in the depots, and then transported to each customer by the vehicles. The objective function is to minimise the sum of production cost, storage cost and transportation cost. The problem includes discrete manufacturing process, depot storage process and logistics transportation scheduling process. Among them, the discrete manufacturing process considers OSS problem; the logistics transportation scheduling process considers vehicle routing problem with time windows (VRPTW).

In recent years, the issue of production, storage and transportation scheduling has attracted the interest of many experts and scholars. Ahmadian et al. (2021) provided a state-of-the-art and comprehensive review and discussed potential research opportunities on the OSS problem of minimising maximum makepan. Ozolins (2019) proposed an accurate dynamic programming algorithm to solve the OSS problem, and the computational results showed that the algorithm could solve moderate benchmark instances. Abdelmaguid (2020) developed two neighborhood search functions and two solution composition functions, and used a path relay meta-heuristic in scattering search to solve the OSS problem. Sheikhalishahi et al. (2019) proposed an OSS model considering human error and preventive maintenance and developed Non-Dominated Sorting Genetic Algorithm II (NSGA-II), Multi-Objective Particle Swarm Optimization (MOPSO) and Strength Par Reciprocal Evolutionary Algorithm II (SPEA-II) three metaheuristic algorithms to find approximate optimal solutions.

Commonly used optimisation algorithms include simulated annealing (Liaw, 1999), genetic algorithm (Liaw, 2000), particle swarm optimisation (Marinakis et al., 2019), ant colony optimisation algorithm (Zhang et al., 2019) and bat algorithm (Cai et al., 2019). Based on the basic principle of the salp swarm algorithm (SSA) (Ewees et al., 2021), this paper proposes an improved salp swarm algorithm (ISSA) to solve the CSDMLMD problem. For the discrete manufacturing process, the crossover and mutation operations in the genetic algorithm and simulated annealing algorithm are introduced; for the logistics transportation scheduling process, the salp swarm algorithm is applied to the discrete domain, and local search strategy is adopted to enhance the search effect of the algorithm. The algorithm is compared with simulated annealing, genetic algorithm and particle swarm algorithm (PSO), and the experimental results are given.

The remainder of this paper is organised as follows: Section 2 introduces the problem description and mathematical model. Section 3 gives the algorithm to solve the testing problem. The simulation analysis results are presented in Section 4. Section 5 draws the conclusion.

2 Problem description and mathematical model

2.1 Problem description

This paper studies the CSDMLMD problem. The problem can be described as follows: the orders are processed in the factory, there are a total of n orders and m machines,

each order has m processes, the processing sequence of each order is arbitrary, the completed orders are stored in the depot, and then the orders are delivered to each customer by the vehicles. The objective function is to minimise the sum of production cost, storage cost and transportation cost. The problem includes discrete manufacturing process, depot storage process and logistics transportation scheduling process. Among them, the discrete manufacturing process considers OSS problem (Hosseinabadi et al., 2019); the logistics transportation scheduling process takes into account VRPTW (Liu et al., 2019).

Figure 1 shows the schematic diagram of CSDMLMD problem, including 6 orders, 1 factory, 3 depots and 6 customers. Among them, there are 3 processing machines and 3 vehicles.

Figure 1 The schematic diagram of CSDMLMD problem (see online version for colours)

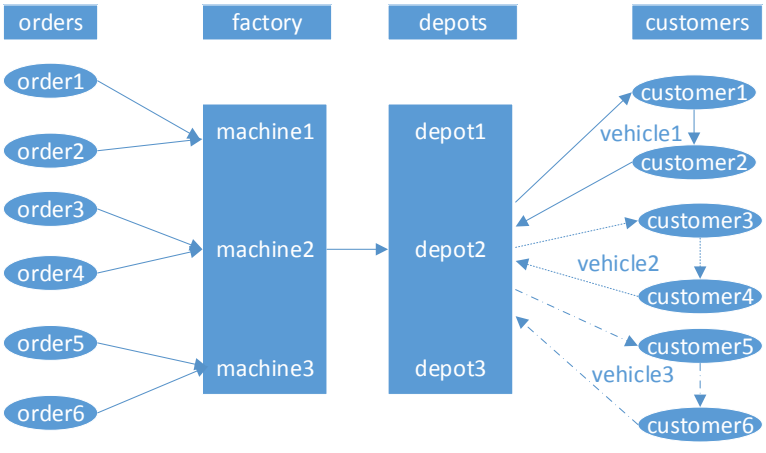


Table 1 presents an example of OSS. In total, 4 orders and 4 processing machines are included. Among them, each order has 4 processes. As shown in Table 1, the number corresponding to the order process in

a indicates the processing time of the order process; the number corresponding to the order process

b represents the processing machine number of the order process.

Table 1 An example for OSS

(a) *Processing times of an example for OSS*

	Operations			
Orders	*1*	*2*	*3*	*4*
1	54	34	61	2
2	9	15	89	70
3	38	19	28	87
4	95	34	7	29

Table 1 An example for OSS (continued)

(*b*) *Machines of an example for OSS*				
	Operations			
Orders	*1*	*2*	*3*	*4*
1	3	1	4	2
2	4	1	2	3
3	1	2	3	4
4	1	3	2	4

The CSDMLMD problem makes the following assumptions.

1 The machine is always available.

2 The machine shall not process more than one order at a time.

3 All orders are available at zero time.

4 The processing sequence of each order is arbitrary.

5 Orders cannot be processed on multiple machines at the same time.

6 The processes of the same order cannot be processed at the same time.

7 Orders are not allowed to be preempted.

8 Cancellation of orders is not allowed.

9 Each order belongs to a different customer.

10 All depots and one factory are located in the same location.

11 The number of depots is limited.

12 The depot has a limited number of vehicles.

13 The capacity of each vehicle is the same.

14 Each customer has its own time window.

2.2 Mathematical model

2.2.1 Mathematical symbols

The mathematical symbols and meanings of the CSDMLMD problem are shown in Table 2.

Table 2 Mathematical symbols for CSDMLMD

Symbol	*Meaning*
n	Number of orders (here, the number of orders equals the number of customers)
i, l	The index of the order (here, the index of the customer is the same as the index of the order)
e	Index of the process

Table 2 Mathematical symbols for CSDMLMD (continued)

Symbol	*Meaning*
m	Number of machines (here, the number of processes of each order is equal to the number of machines)
j	Index of the machine
K	Number of vehicles
k	Index of the vehicle
R	Number of depots
r	Index of the depot
p_{iej}	The processing time of process e of order i processed on machine j
$c^{'}$	Unit time cost of order processing
c_{iej}	The completion time of process e of order i processed on machine j
c_{ij}	The completion time of the order i on the machine j
C_i	The completion time of the order i
$C_{\max} = \max_{1 \le i \le n} \{C_i\}$	Maximum completion time of the orders
uc	Unit storage cost of unit time for the orders in the depot
T_i	The storage time of order i in the depot
$b_k^{'}$	Unit distance cost of the vehicle k
Q	Maximum load capacity of factory vehicle k
$d_i^{'}$	Distance between the depot and customer i
$d_{il}^{''}$	Distance from customer i to the customer l
S_k	Customer collection for the vehicle k
OR_i	The requirements of the customer i
$[a_i, b_i]$	Time window of the customer i; a_i is the allowed service start time and b_i is the allowed service end time
st_i	Actual service start time of the customer i
x_{iej}	If process e of the order i is assigned to the machine j, $x_{iej} = 1$; otherwise, $x_{iej} = 0$
x_{ij}	If the order i is assigned to the machine j, $x_{ij} = 1$; otherwise, $x_{ij} = 0$
y_{ki}	If the customer i is the first customer of the vehicle k, $y_{ki} = 1$; otherwise, $y_{ki} = 0$
$y_{kil}^{'}$	If the vehicle k is directly transported to the customer l after the customer i, $y_{kil}^{'} = 1$; otherwise, $y_{kil}^{'} = 0$
$y_{ki}^{''}$	If the customer i is the last customer of the vehicle k, $y_{ki}^{''} = 1$; otherwise, $y_{ki}^{*} = 0$

2.2.2 Objective function

The objective function is to minimise the sum of production cost, storage cost and transportation cost.

$$\begin{aligned}\min\ F = & \sum_{i=1}^{n} OR_i C_i c' + \\ & \sum_{i=1}^{n} OR_i T_i uc + \\ & \sum_{k=1}^{K}\sum_{i=1}^{n} (y_{ki} + y''_{ki}) d'_i b'_k + \\ & \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{l=1}^{n} y'_{kil} d''_{il} b'_k\end{aligned} \tag{1}$$

Among them, the first item in formula (1) is the production cost; the second item is the storage cost; the third and fourth items are the transportation cost.

2.2.3 Constraints

$$C_{\max} - \sum_{i=1}^{n}\sum_{e=1}^{m} p_{iej} x_{iej} \geq 0, \quad j = 1,\ldots,m \tag{2}$$

$$C_i \leq C_{\max}, \quad i = 1,2,\ldots,n \tag{3}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i = 1,\ldots,n \tag{4}$$

$$\sum_{i=1}^{n} x_{ij} > 0, \quad j = 1,\ldots,m \tag{5}$$

$$c_{ij} > 0, \quad i = 1,\ldots,n, j = 1,\ldots,m \tag{6}$$

$$\sum_{i=1}^{n} y_{ki} \leq 1, \quad k = 1,2,\ldots,K \tag{7}$$

$$y_{ki} + \sum_{l=1}^{n} y'_{kli} = \sum_{l=1}^{n} y'_{kil} + y''_{ki}, \quad k = 1,2,\ldots,K, \quad i = 1,2,\ldots,n \tag{8}$$

$$\sum_{i=1}^{n} (y_{ki} + y''_{ki}) OR_i + \sum_{i=1}^{n}\sum_{l=1}^{n} y'_{kil} OR_l \leq Q, \quad k = 1,2,\ldots,K \tag{9}$$

$$\sum_{i,l \in S_h} y'_{kil} \leq |S_k| - 1, \quad k = 1,2,\ldots,K \tag{10}$$

$$a_i \leq st_i \leq b_i, \quad i = 1,2,\ldots,n \tag{11}$$

$$x_{ij}, y_{ki}, y'_{kil}, y''_{ki} \in \{0,1\}, \quad i,l = 1,\ldots,n, j = 1,\ldots,m, k = 1,\ldots,K \tag{12}$$

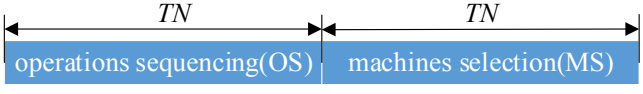
Here, formula (2) ensures that the maximum completion time is greater than or equal to the total completion time of the processes of the orders on each machine; formula (3) ensures that the completion time of each order shall not exceed the maximum completion time; formula (4) ensures that each order is processed on only one machine; formula (5) ensures that each machine has orders for processing; formula (6) ensures that the completion time of all processes is greater than zero; equation (7) ensures that each vehicle can only transport one order at most; formula (8) ensures that the vehicle is continuous during transportation; formula (9) ensures that the order of vehicle transportation does not exceed its maximum of its capacity; formula (10) ensures that the vehicle does not generate sub-loops during transportation; equation (11) ensures the constraints of time windows for each customer; formula (12) ensures the value range of decision variables.

3 Algorithm design

3.1 Coding of solutions

This section mainly introduces the coding of the solution of OSS. The encoding of the solution of OSS consists of two parts, operations sequencing (OS) and machines selection (MS). The encoding of the solution is shown in Figure 2. Here, $TN = nm$.

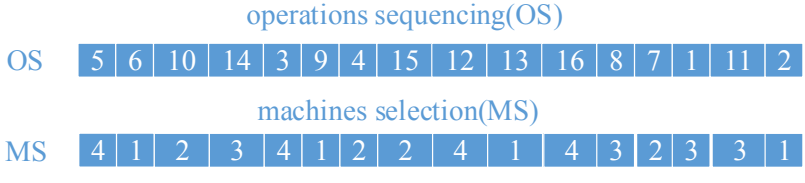
Figure 2 Coding of the solution (see online version for colours)



3.1.1 Operations sequence (OS)

The value on each bit of operations sequencing is represented by a number. For example, the number 4 indicates the fourth process of the first order, and the number 6 represents the second process of the second order. For the example in Table 1, operations sequence can be encoded as OS = [5 6 10 14 3 9 4 15 12 13 16 8 7 1 11 2], as shown in Figure 3.

Figure 3 Coding of solutions for the example of Table 1 (see online version for colours)



3.1.2 Machines selection (MS)

The value on each bit of machines selection is represented by the machine number. Based on the selected operation, determine the machine number to which the order is to be processed. The number of occurrences of each machine number is equal to the number of machines m. For the example in Table 1, machines selection can be encoded as MS = [4 1 2 3 4 1 2 2 4 1 4 3 2 3 3 1], as shown in Figure 3.

3.2 SSA

SSA can be expressed by the following formulas.

$$x_j^1 = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j) & 0.5 \le c_3 < 1 \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j) & 0 < c_3 < 0.5 \end{cases} \tag{13}$$

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \tag{14}$$

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}), \quad i \ge 2 \tag{15}$$

Here, x_j^1 represents the position of the first salp(the leader) in the dimension j. F_j is the position of the food source in the dimension j. ub_j shows the upper bound of the dimension j, and lb_j shows the lower bound of the dimension j. c_2 and c_3 are the random numbers with the range from zero to one. l is the number of the current iteration, and L is the maximum number of iterations. x_j^i represents the position of the salp i (the follower) in the dimension j.

3.3 ISSA

ISSA to solve OSS and ISSA to solve VRPTW are detailed below.

3.3.1 ISSA to solve OSS

This section describes in detail ISSA to solve OSS. The crossover and mutation operations in genetic algorithm and the neighbourhood search operation in simulated annealing are introduced into SSA to form an ISSA. These operations are described separately below.

1 *One point crossover*

First, a crossover position is randomly determined and the gene from the first position in parent 1 to the crossover position is passed to the offspring in the same position. Then, the genes in the parent 2 that are the same as the first position to the crossover position of the parent 1 are removed, and the remaining genes in the parent 2 are embedded into the positions in the offspring respectively.

2 *Two point crossover*

First, two crossover positions are randomly determined and the genes between these two crossover positions are passed from parent 1 to offspring. Then remove the same gene between the two intersection positions of parent 2 and parent 1, and embed the remaining genes in parent 2 into the empty positions of the offspring respectively.

3 *Displacement mutation*

Two genes are randomly selected and replaced.

4 *Shift mutation*

Two mutation positions are randomly selected, and the genes located between these two positions are rotated to the left.

5 *Neighbourhood search*

The methods used in the neighbourhood search part to generate neighbourhood solutions are swap operation, reverse operation and insert operation. These operations are described below.

5.1 *Swap operation*

Randomly select two elements and swap the two elements.

5.2 *Reverse operation*

Two reversal positions are randomly selected, and the elements between the two reversal positions are reversed.

5.3 *Insert operation*

First, copy a new identical individual, and randomly locate the two insertion points s and e. Suppose $s < e$. Then, take out the elements that are from the position $s+1$ to the position e in the individual and assign them to the positions from s to $e-1$ in the new individual. And assign the element of the position s in the individual to the position e in the new individual.

The steps of ISSA to solve OSS are shown in Algorithm 1.

Algorithm 1 ISSA to solve OSS

Input: salp population P, initial temperature $T = T0$, maximum number of iterations L, random number $rand \in [0,1]$,

1.Initialize each individual of the population $x_i (i = 1, 2, \ldots, P)$

2.**for** l **in** L

3. Calculate the fitness W_i of each individual x_i

4. x_*=the best individual

W_*=the best fitness

5. **for** $i = 1 : P$

6. **if** $i == 1$

7. if $rand > 0.3$

8. Apply displacement mutation operation to update the individual x_i

9. else

10. Apply shift mutation operation to update the individual x_i

end

else

if $rand > 0.4$

Apply one point crossover operation to update the individual x_i

else

Apply two point crossover operation to update the individual x_i

end

end

Apply neighborhood search to generate neighborhood solution

Apply simulated annealing to update the individual x_i

Calculate the fitness W_i of the individual x_i

if $W_i < W_*$

$x_* = x_i$

end

end

end

end

Output: optimal solution x_*.

3.3.2 ISSA to solve VRPTW

According to formulas (13)–(15), we redefine the formulas in the discrete domain as follows.

Formula (13) can be transformed in the following. Define the food source F as x_* which refers to the best solution. x_1 is the leader which is the first salp. Suppose c_0 as $(ub - lb)c_2 + lb$, where $c_2 \in [0,1]$. Here, c_r represents $2 \times c_0 \times random$, where $random \in [0,1]$. However, formula (14) in the continuous domain is unchanged. We directly use formula (14) in the discrete domain. If $c_1 \times c_0 > c_r$, $x_1 = x_*$. Otherwise, x_1 remains unchanged.

Formula (15) can be described as follows. x_i is the i th follower salp where $i \geq 2$. Randomly generate $rand$, where $rand \in [0,1]$. If $rand > 0.5$, $x_i = x_{i-1}$. Here, x_{i-1} refers to the last salp. Otherwise, x_i stays the same.

Local search method adopts the large neighbourhood search algorithm.

The steps of ISSA used to solve VRPTW are shown in Algorithm 2.

Algorithm 2 ISSA to solve VRPTW

Input: salp population P, upper bound ub, lower bound lb,

1.Initialize each individual of the population $x_i (i = 1, 2, \ldots, P)$

2.**for** l **in** L

3. Calculate the fitness W_i of each individual x_i

4. x_*=the best individual

W_*=the best fitness

5. Update c_1 by equation (14)

6. **for** i **in** P

7. **if** $i == 1$

8. Apply the operations of section 3.3.2 to update the leading salp's position

9. **else**

10. Apply the operations of section 3.3.2 to update the following salp's position

end

11. Apply local search method in section 3.3.2 to find out the best position x_i^{pbest} and the best fitness W_i^{pbest} in the neighborhood of x_i

12. **if** $W_i^{pbest} < W_i$

13. $x_i = x_i^{pbest}$

end

14. **if** $W_i^{pbest} < W_*$

15. $x_* = x_i^{pbest}$

$W_* = W_i^{pbest}$

end

end

end

Output: global optimal position x_*.

4 Simulation analysis

4.1 Simulation example

The examples of OSS uses 60 benchmarks of Taillard's instances of open shops scheduling. These benchmarks are tai4_4, tai5_5, tai7_7, tai10_10, tai15_15, and tai20_20. Among them, tai4_4 has a total of 10 examples, namely tai4_4_1 to tai4_4_10, and so on.

There are a total of 60 instances in the depot storage section, and the number of warehouses is set to 3. Each instance stores orders according to customer needs, and the corresponding instance numbers are S001 to S060.

The examples used in VRPTW considers the situation of large customers, that is, the customer's demand is large, which is more in line with the actual application scenario. The specific parameter settings are as follows: CUST NO. indicates the number of the depot and customer, such as 0 for the depot, 1 for the customer 1. XCOORD and YCOORD represent the abscissa and ordinate of the depot and customer respectively, and the value range is [0,99]. DEMAND represents the customer's demand, the value range is [100, 999], and the depot is set to 0. READY TIME indicates the allowable service start time, the value range is [0,990], DUE DATE indicates the service termination time, the value range is [10,1000], SERVICE TIME indicates the service time, the value range is [10,90]. Table 3 shows the range of parameter values for a VRPTW with 4 customers.

Table 3 Parameter value range of VRPTW for 4 customers

CUST NO	*XCOORD*	*YCOORD*	*DEMAND*	*READY TIME*	*DUE DATE*	*SERVICE TIME*
0	[0,99]	[0,99]	0	0	1000	0
1	[0,99]	[0,99]	[100,999]	[0,990]	[10,1000]	[10,90]
2	[0,99]	[0,99]	[100,999]	[0,990]	[10,1000]	[10,90]
3	[0,99]	[0,99]	[100,999]	[0,990]	[10,1000]	[10,90]
4	[0,99]	[0,99]	[100,999]	[0,990]	[10,1000]	[10,90]

There are a total of 60 instances of VRPTW, namely 4 major customers, 5 major customers, 7 major customers, 10 major customers, 15 major customers and 20 major customers. 4 of them have a total of 10 examples, and so on. The corresponding instance numbers are V001 to V060.

Examples of CSDMLTW problems are shown in Table 4.

Table 4 CSDMLTW problem example

Problem	*Instances*	*Problem*	*Instances*
C001	tai4_4_1 + S001 + V001	C031	tai10_10_1 + S031 + V031
C002	tai4_4_2 + S002 + V002	C032	tai10_10_2 + S032 + V032
C003	tai4_4_3 + S003 + V003	C033	tai10_10_3 + S033 + V033
C004	tai4_4_4 + S004 + V004	C034	tai10_10_4 + S034 + V034
C005	tai4_4_5 + S005 + V005	C035	tai10_10_5 + S035 + V035

Table 4 CSDMLTW problem example (continued)

Problem	*Instances*	*Problem*	*Instances*
C006	tai4_4_6 + S006 + V006	C036	tai10_10_6 + S036 + V036
C007	tai4_4_7 + S007 + V007	C037	tai10_10_7 + S037 + V037
C008	tai4_4_8 + S008 + V008	C038	tai10_10_8 + S038 + V038
C009	tai4_4_9 + S009 + V009	C039	tai10_10_9 + S039 + V039
C010	tai4_4_10 + S010 + V010	C040	tai10_10_10 + S040 + V040
C011	tai5_5_1 + S011 + V011	C041	tai15_15_1 + S041 + V041
C012	tai5_5_2 + S012 + V012	C042	tai15_15_2 + S042 + V042
C013	tai5_5_3 + S013 + V013	C043	tai15_15_3 + S043 + V043
C014	tai5_5_4 + S014 + V014	C044	tai15_15_4 + S044 + V044
C015	tai5_5_5 + S015 + V015	C045	tai15_15_5 + S045 + V045
C016	tai5_5_6 + S016 + V016	C046	tai15_15_6 + S046 + V046
C017	tai5_5_7 + S017 + V017	C047	tai15_15_7 + S047 + V047
C018	tai5_5_8 + S018 + V018	C048	tai15_15_8 + S048 + V048
C019	tai5_5_9 + S019 + V019	C049	tai15_15_9 + S049 + V049
C020	tai5_5_10 + S020 + V020	C050	tai15_15_10 + S050 + V050
C021	tai7_7_1 + S021 + V021	C051	tai20_20_1 + S051 + V051
C022	tai7_7_2 + S022 + V022	C052	tai20_20_2 + S052 + V052
C023	tai7_7_3 + S023 + V023	C053	tai20_20_3 + S053 + V053
C024	tai7_7_4 + S024 + V024	C054	tai20_20_4 + S054 + V054
C025	tai7_7_5 + S025 + V025	C055	tai20_20_5 + S055 + V055
C026	tai7_7_6 + S026 + V026	C056	tai20_20_6 + S056 + V056
C027	tai7_7_7 + S027 + V027	C057	tai20_20_7 + S057 + V057
C028	tai7_7_8 + S028 + V028	C058	tai20_20_8 + S058 + V058
C029	tai7_7_9 + S029 + V029	C059	tai20_20_9 + S059 + V059
C030	tai7_7_10 + S030 + V030	C060	tai20_20_10 + S060 + V060

4.2 Experimental environment

The implementation is programed in Matlab and run on an Intel i7 12700H computer(2.70GHz CPU) with 16.00 GB RAM.

4.3 Experimental results

The goal of OSS is to minimise the makepan. ISSA is compared with simulated annealing algorithm (SA), genetic algorithm (GA) and PSO, and the comparison of the algorithms for OSS is shown in Table 5. Where the deviation is defined as $deviation = \frac{ISSA-BKS}{BKS} \times 100\%$. It can be concluded that 20 of the 60 results are obtained by ISSA.

Table 5 Comparison of the algorithms for OSS

Instance	*BKS*	*SA*	*GA*	*PSO*	*ISSA*	*Deviation* (%)
tai4_4_1	193	193	193	193	193	0.00
tai4_4_2	236	236	236	236	236	0.00
tai4_4_3	271	271	271	271	271	0.00
tai4_4_4	250	250	250	250	250	0.00
tai4_4_5	295	295	295	295	295	0.00
tai4_4_6	189	189	189	189	189	0.00
tai4_4_7	201	201	201	201	201	0.00
tai4_4_8	217	217	217	217	217	0.00
tai4_4_9	261	261	261	261	261	0.00
tai4_4_10	217	217	217	217	217	0.00
tai5_5_1	300	300	300	300	300	0.00
tai5_5_2	262	262	262	262	262	0.00
tai5_5_3	323	323	323	323	323	0.00
tai5_5_4	310	310	310	310	314	1.29
tai5_5_5	326	326	326	326	326	0.00
tai5_5_6	312	312	312	312	312	0.00
tai5_5_7	303	303	303	303	303	0.00
tai5_5_8	300	300	300	300	303	1.00
tai5_5_9	353	353	353	353	355	0.57
tai5_5_10	326	326	326	326	326	0.00
tai7_7_1	435	435	435	435	438	0.69
tai7_7_2	443	445	443	443	452	2.03
tai7_7_3	468	468	468	468	475	1.50
tai7_7_4	463	463	463	463	471	1.73
tai7_7_5	416	416	416	416	417	0.24
tai7_7_6	451	456	451	451	464	2.88
tai7_7_7	422	428	422	422	428	1.42
tai7_7_8	424	424	424	424	427	0.71
tai7_7_9	458	458	458	458	459	0.22
tai7_7_10	398	398	398	398	400	0.50
tai10_10_1	637	645	637	637	653	2.51
tai10_10_2	588	588	588	588	593	0.85
tai10_10_3	598	602	598	598	608	1.67
tai10_10_4	577	577	577	577	578	0.17
tai10_10_5	640	640	640	640	647	1.09
tai10_10_6	538	538	538	538	538	0.00
tai10_10_7	616	616	616	616	619	0.49
tai10_10_8	595	595	595	595	600	0.84

Table 5 Comparison of the algorithms for OSS (continued)

Instance	*BKS*	*SA*	*GA*	*PSO*	*ISSA*	*Deviation* (%)
tai10_10_9	595	596	595	595	598	0.50
tai10_10_10	596	596	596	596	603	1.17
tai15_15_1	937	937	937	937	940	0.32
tai15_15_2	918	920	918	918	930	1.31
tai15_15_3	871	871	871	871	876	0.57
tai15_15_4	934	934	934	934	934	0.00
tai15_15_5	946	946	946	946	962	1.69
tai15_15_6	933	933	933	933	944	1.18
tai15_15_7	891	891	891	891	895	0.45
tai15_15_8	893	893	893	893	893	0.00
tai15_15_9	899	905	899	899	927	3.11
tai15_15_10	902	902	902	902	921	2.11
tai20_20_1	1155	1155	1155	1155	1214	5.02
tai20_20_2	1241	1253	1241	1241	1317	5.00
tai20_20_3	1257	1257	1257	1257	1294	2.94
tai20_20_4	1248	1248	1248	1248	1304	4.49
tai20_20_5	1256	1256	1256	1256	1308	3.42
tai20_20_6	1204	1204	1204	1204	1266	4.32
tai20_20_7	1294	1294	1294	1294	1340	3.55
tai20_20_8	1169	1189	1177	1169	1278	9.32
tai20_20_9	1289	1289	1289	1289	1333	3.34
tai20_20_10	1241	1241	1241	1241	1298	3.63

The experimental results of the depot storage part are shown in Table 6. No intelligent algorithms are used in this part. Here, $uc = 10$, $T_i = 1$.

Table 6 Experimental results of depot storage

Instance	*Cost*	*Instance*	*Cost*
S001	5400	S031	15800
S002	5400	S032	16000
S003	6300	S033	17500
S004	7400	S034	16600
S005	6800	S035	14600
S006	7500	S036	12800
S007	6900	S037	16700
S008	7400	S038	13500
S009	6100	S039	20300
S010	6500	S040	17200

Table 6 Experimental results of depot storage (continued)

Instance	*Cost*	*Instance*	*Cost*
S011	8500	S041	20300
S012	7200	S042	25000
S013	5800	S043	25900
S014	7500	S044	20800
S015	6900	S045	23600
S016	8300	S046	24100
S017	5800	S047	19900
S018	6300	S048	22400
S019	7100	S049	24000
S020	6600	S050	28700
S021	12200	S051	41100
S022	10100	S052	29300
S023	12200	S053	31000
S024	9600	S054	34900
S025	11100	S055	39900
S026	11600	S056	32200
S027	11300	S057	34800
S028	9900	S058	29300
S029	9500	S059	35200
S030	11400	S060	33300

VRPTW aims to minimise the number of vehicles needed (NV) and the total distance (TD). ISSA is compared with SA, GA and PSO and the comparison of the algorithms for VRPTW is shown in Table 7. It can be concluded that among 60 results, 58 better results are obtained by ISSA on NV, and 41 better results are on TD.

Optimal solution for V059 can be shown as follows.

Optimal Solution:

Number of Vehicles: 4, Total Distances: 401.7346,

Route1: $0 \rightarrow 8 \rightarrow 10 \rightarrow 9 \rightarrow 0$

Route2: $0 \rightarrow 7 \rightarrow 17 \rightarrow 19 \rightarrow 20 \rightarrow 18 \rightarrow 16 \rightarrow 0$

Route3: $0 \rightarrow 5 \rightarrow 14 \rightarrow 15 \rightarrow 13 \rightarrow 6 \rightarrow 0$

Route4: $0 \rightarrow 3 \rightarrow 2 \rightarrow 11 \rightarrow 12 \rightarrow 4 \rightarrow 1 \rightarrow 0$

Optimal delivery route solved by ISSA for V059 is shown as Figure 4.

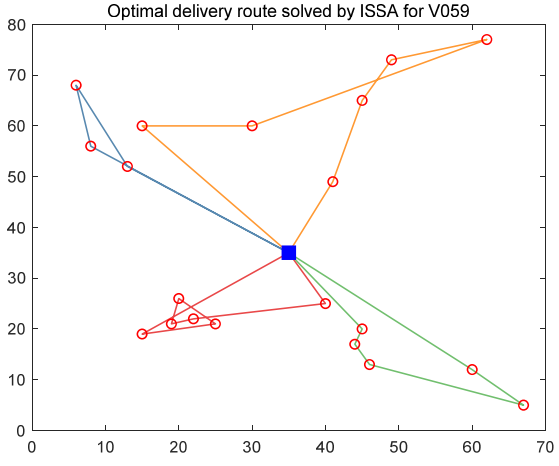
Table 7 Comparison of the algorithms for VRPTW

Instance	*SA*		*GA*		*PSO*		*ISSA*	
	NV	*TD*	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*
V001	2	115.38	2	115.38	2	115.38	1	124.17
V002	1	93.02	1	93.02	1	93.02	1	93.02
V003	1	140.91	1	140.91	1	140.91	1	140.91
V004	1	145.05	1	145.05	1	145.05	1	145.05
V005	1	109.25	1	109.25	1	109.25	1	109.25
V006	1	105.16	1	105.16	1	105.16	1	105.16
V007	1	140.56	1	140.56	1	140.56	1	140.56
V008	1	62.40	1	62.40	1	62.40	1	62.40
V009	1	74.66	1	74.66	1	74.66	1	74.66
V010	1	109.88	1	109.88	1	109.88	1	109.88
V011	1	145.51	1	145.51	1	145.51	1	145.51
V012	1	81.66	1	81.66	1	81.66	1	81.66
V013	1	123.59	1	123.59	1	123.59	1	123.59
V014	1	180.07	1	180.75	1	180.07	1	180.07
V015	2	139.13	2	139.13	2	139.13	2	139.13
V016	2	192.25	2	192.25	2	192.25	2	192.25
V017	2	187.29	2	187.29	2	187.29	2	187.29
V018	1	112.30	1	112.30	1	112.30	1	112.30
V019	1	117.60	1	117.60	1	117.60	1	117.60
V020	1	130.47	1	130.47	1	130.47	1	130.47
V021	2	203.05	2	203.05	2	203.05	2	203.05
V022	2	141.06	2	141.06	2	141.06	2	141.06
V023	2	202.67	2	202.67	2	202.67	2	202.67
V024	2	205.48	2	205.48	2	205.48	1	220.99
V025	2	175.58	2	175.58	2	175.58	2	176.42
V026	2	163.79	2	163.79	2	163.79	2	163.79
V027	2	176.12	2	176.12	2	176.12	2	176.12
V028	2	196.27	2	196.27	2	196.27	1	199.72
V029	2	145.51	2	145.51	2	145.51	2	145.51
V030	2	142.99	2	142.99	2	142.99	2	142.99
V031	3	325.98	3	325.98	3	325.98	2	326.08
V032	3	287.31	2	281.09	2	281.09	2	281.09
V033	2	244.05	2	244.05	2	244.05	2	244.05
V034	2	176.67	2	176.67	2	176.67	2	176.67
V035	2	172.36	2	168.19	2	168.19	2	168.19
V036	2	145.93	2	145.93	2	145.93	2	145.93
V037	2	186.71	2	186.71	2	186.71	2	186.71

Table 7 Comparison of the algorithms for VRPTW (continued)

Instance	*SA*		*GA*		*PSO*		*ISSA*	
	NV	*TD*	*NV*	*TD*	*NV*	*TD*	*NV*	*TD*
V038	2	198.82	2	198.39	2	198.39	2	198.39
V039	3	276.77	3	276.77	3	276.77	3	276.77
V040	3	220.43	3	220.43	3	220.43	2	227.45
V041	4	304.41	4	304.41	4	304.41	4	304.41
V042	4	339.69	3	337.91	3	337.91	3	359.88
V043	3	257.76	3	257.76	3	257.76	3	266.70
V044	3	292.62	3	285.81	3	292.18	3	311.25
V045	3	336.49	3	336.49	3	336.49	3	336.49
V046	4	383.63	3	378.11	3	378.11	3	395.88
V047	3	250.98	3	250.73	3	250.73	3	250.98
V048	3	301.88	3	301.88	3	301.88	3	301.88
V049	3	284.42	3	282.65	3	282.65	3	282.65
V050	3	306.79	3	304.29	3	304.29	3	304.29
V051	5	455.48	4	445.18	4	445.18	5	462.94
V052	4	379.94	4	379.94	4	379.94	5	400.66
V053	4	286.45	4	285.83	4	285.83	4	286.45
V054	4	372.12	4	372.12	4	384.85	4	389.54
V055	5	531.27	5	518.18	5	518.18	5	535.07
V056	4	302.06	4	298.34	4	301.52	4	306.78
V057	4	330.65	4	330.65	4	330.65	4	333.12
V058	3	316.26	3	316.26	3	328.92	3	335.59
V059	5	411.91	4	401.73	4	401.73	4	401.73
V060	4	310.19	4	300.13	4	300.13	4	300.13

Figure 4 Optimal delivery route solved by ISSA for V059 (see online version for colours)



Optimal solution for V060 can be shown as follows.

Optimal Solution:

Number of Vehicles: 4, Total Distances: 300.1295,

Route1: $0 \to 3 \to 4 \to 12 \to 5 \to 15 \to 11 \to 0$

Route2: $0 \to 18 \to 20 \to 17 \to 19 \to 9 \to 8 \to 10 \to 0$

Route3: $0 \to 13 \to 6 \to 7 \to 2 \to 0$

Route4: $0 \to 14 \to 1 \to 16 \to 0$

Optimal delivery route solved by ISSA for V060 is shown as Figure 5.

Figure 5 Optimal delivery route solved by ISSA for V060 (see online version for colours)

The comparison of the algorithms for CSDMLTW problem is shown in Table 8. Here, $c^{'} = 100$, $b_k^{'} = 300$. It can be concluded that 19 results obtained by ISSA are better in 60 experiments.

Table 8 Comparison of the algorithms for CSDMLTW problem

Problem	*SA*	*GA*	*PSO*	*ISSA*
C001	59314	59314	59314	61951
C002	56906	56906	56906	56906
C003	75673	75673	75673	75673
C004	75915	75915	75915	75915
C005	69075	69075	69075	69075
C006	57948	57948	57948	57948
C007	69168	69168	69168	69168
C008	47820	47820	47820	47820
C009	54598	54598	54598	54598

Table 8 Comparison of the algorithms for CSDMLTW problem (continued)

Problem	*SA*	*GA*	*PSO*	*ISSA*
C010	61164	61164	61164	61164
C011	82153	82153	82153	82153
C012	57898	57898	57898	57898
C013	75177	75177	75177	75177
C014	92521	92725	92521	92921
C015	81239	81239	81239	81239
C016	97175	97175	97175	97175
C017	92287	92287	92287	92287
C018	69990	69990	69990	70290
C019	77680	77680	77680	77880
C020	78341	78341	78341	78341
C021	116615	116615	116615	116915
C022	96918	96718	96718	97618
C023	119801	119801	119801	120501
C024	117544	117544	117544	122997
C025	105374	105374	105374	105726
C026	106337	105837	105837	107137
C027	106936	106336	106336	106936
C028	111181	111181	111181	112516
C029	98953	98953	98953	99053
C030	94097	94097	94097	94297
C031	178094	177294	177294	178924
C032	160993	159127	159127	159627
C033	150915	150515	150515	151515
C034	127301	127301	127301	127401
C035	130308	129057	129057	129757
C036	110379	110379	110379	110379
C037	134313	134313	134313	134613
C038	132646	132517	132517	133017
C039	162931	162831	162831	163131
C040	142929	142929	142929	145735
C041	205323	205323	205323	205623
C042	218907	218173	218173	225964
C043	190328	190328	190328	193510
C044	201986	199943	201854	207575
C045	219147	219147	219147	220747
C046	232489	230833	230833	237264
C047	184294	184219	184219	184694

Table 8 Comparison of the algorithms for CSDMLTW problem (continued)

Problem	*SA*	*GA*	*PSO*	*ISSA*
C048	202264	202264	202264	202264
C049	199826	198695	198695	201495
C050	210937	210187	210187	212087
C051	293244	290154	290154	301182
C052	268582	267382	267382	279798
C053	242635	242449	242449	246335
C054	271336	271336	275155	282162
C055	324881	320954	320954	330321
C056	243218	242102	243056	249834
C057	263395	263395	263395	268736
C058	243078	241878	244876	257777
C059	287673	284619	284619	288919
C060	250457	247439	247439	251939

5 Conclusion

In this paper, ISSA is used to solve CSDMLMD. Comparing ISSA with SA, GA and PSO, relatively good results can be obtained. The next step is to mix salp swarm algorithm with other intelligent algorithms to solve this problem.

Acknowledgements

This work is partially supported by the Science and Technology Program of Guangdong Province under grant Nos. 2016A050502060 and 2020B1010010005, the Science and Technology Program of Guangzhou under grant Nos. 202206010011 and 2023B03J1339.

References

Abdelmaguid, T.F. (2020) 'Scatter search with path relinking for multiprocessor open shop scheduling', *Computers and Industrial Engineering*, Vol. 141, p.106292.

Ahmadian, M.M., Khatami, M., Salehipour, A. and Cheng, T.C.E. (2021) 'Four decades of research on the open-shop scheduling problem to minimize the makespan', *European Journal of Operational Research*, Vol. 295, pp.399–426, doi: https://doi.org/10.1016/j.ejor.2021.03.026

Cai, Y., Qi, Y., Cai, H., Huang, H. and Chen, H. (2019) 'Chaotic discrete bat algorithm for capacitated vehicle routing problem', *International Journal of Autonomous and Adaptive Communications Systems*, Vol. 12, No. 2, pp.91–108.

Ewees, A.A., Al-qaness, M.A.A. and Elaziz, M.A. (2021) 'Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with setup times', *Applied Mathematical Modelling*, Vol. 94, pp.285–305.

Liaw, C.F. (1999) 'Applying simulated annealing to the open shop scheduling problem', *IIE Transactions*, Vol. 31, No. 5, pp.457–465.

Liaw, C.F. (2000) 'A hybrid genetic algorithm for the open shop scheduling problem', *European Journal of Operational Research*, Vol. 124, No. 1, pp.28–42.

Liu, R., Tao, Y. and Xie, X. (2019) 'An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits', *Computers and Operations Research*, Vol. 101, pp.250–262.

Marinakis, Y., Marinaki, M. and Migdalas, A. (2019) 'A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows', *Information Sciences*, Vol. 481, pp.311–329.

Ozolins, A. (2019) 'Dynamic programming approach for solving the open shop problem', *Central European Journal of Operations Research*, Vol. 29, pp.291–306.

Rahmani Hosseinabadi, A.A., Vahidi, J., Saemi, B., Sangaiah, A.K. and Elhoseny, M. (2019) 'Extended genetic algorithm for solving open-shop scheduling problem', *Soft Computing*, Vol. 23, No. 13, pp.5099–5116.

Sheikhalishahi, M., Eskandari, N., Mashayekhi, A. and Azadeh, A. (2019) 'Multi-objective open shop scheduling by considering human error and preventive maintenance', *Applied Mathematical Modelling*, Vol. 67, pp.573–587.

Zhang, H., Zhang, Q., Ma, L., Zhang, Z. and Liu, Y. (2019) 'A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows', *Information Sciences*, Vol. 490, pp.166–190.