Bouzidi, Abdelhamid; Riffi, Mohammed Essaid; Barkatou, Mohammed

Article

Cat swarm optimization for solving the open shop scheduling problem

Journal of Industrial Engineering International

Provided in Cooperation with:
Islamic Azad University (IAU), Tehran

Suggested Citation: Bouzidi, Abdelhamid; Riffi, Mohammed Essaid; Barkatou, Mohammed (2019) : Cat swarm optimization for solving the open shop scheduling problem, Journal of Industrial Engineering International, ISSN 2251-712X, Springer, Heidelberg, Vol. 15, Iss. 2, pp. 367-378, https://doi.org/10.1007/s40092-018-0297-z

Journal of Industrial Engineering International (2019) 15:367–378
https://doi.org/10.1007/s40092-018-0297-z

ORIGINAL RESEARCH

Cat swarm optimization for solving the open shop scheduling problem

Abdelhamid Bouzidi[1,2] · **Mohammed Essaid Riffi**[2] · **Mohammed Barkatou**[3]

Received: 27 February 2018 / Accepted: 22 October 2018 / Published online: 14 November 2018

Abstract
This paper aims to prove the efficiency of an adapted computationally intelligence-based behavior of cats called the cat swarm optimization algorithm, that solves the open shop scheduling problem, classified as NP-hard which its importance appears in several industrial and manufacturing applications. The cat swarm optimization algorithm was applied to solve some benchmark instances from the literature. The computational results, and the comparison of the relative percentage deviation of the proposed metaheuristic with other's existing in the literature, show that the cat swarm optimization algorithm yields good results in reasonable execution time.

Keywords Scheduling problem · Swarm optimization · Behavior of cats · Computationally intelligence · Discrete

Introduction

To solve real optimization problems such as in industrial and manufacturing applications, the problem should be formulated as a theoretical problem. In 1974, Gonzalez and Sahni (1976) had introduced one of the most known complex combinatorial problem called the open shop scheduling problem (OSSP). There are several real-world applications of the OSSP, such as system-on-a-chip testing (Iyengar and Chakrabarty 2002), the area of satellite-switched time-division multiple access (Dell'Amico and Martello 1996), routing packets (Suel 1995), the scheduling and wavelength assignment problem in optical networks that are based on the wavelength-division-multiplexing technology (Bampis and Rouskas 2002), routing in optical transpose interconnect system (Lucas 2010), in routing in heterogeneous networks to model communications schedules (Bhat et al. 2000).

The OSSP problem is classified as NP-hard (Gonzalez and Sahni 1976), that is why some researchers had tried to solve it by introducing some methods, such as exact methods, polynomial time algorithm proposed by Gonzalez and Sahni (1976), and the branch and bound developed by Brucker et al. (1997). In general, the exact methods can attain some local solutions and rarely a global solution. The metaheuristic has proven its efficiency to reach the global solution of some problems such as the OSSP. Some metaheuristics are used to solve the OSSP problem, such as simulated annealing (Liaw 1999a) and Tabu search algorithm proposed by Liaw (1999b), genetic algorithm proposed by Prins (2000), extended genetic algorithm proposed by Rahmani Hosseinabadi et al. (2018), hybrid ant colony optimization proposed by Blum (2005), bee colony optimization proposed by Huang and Lin (2011), particle swarm optimization proposed by Sha and Hsu (2008).

This paper presents a new approach for solving the open shop scheduling problem, which is called the cat swarm optimization. In order to prove that the proposed method is efficient, the result obtained by its application to solve some benchmarks instances is compared with those existing in the literature. The rest of the paper is organized as follows; Section two is a description and formulation of the open shop scheduling problem, with an example. Section three presents the cat swarm optimization algorithm, its used parameters, and its process. Section four describes the proposed adaptation of cat swarm optimization to solve the open shop scheduling problem.

✉ Abdelhamid Bouzidi
mr.abdelhamid.bouzidi@gmail.com

Mohammed Essaid Riffi
said@riffi.fr

Mohammed Barkatou
barkatou.m@ucd.ac.ma

[1] Lab. LISGA, Centre El Jadida, Groupe ISGA, EL Jadida, Morocco

[2] Lab, LAROSERI, Department of Computer Science, Faculty of Sciences, Chouaib Doukkali University, EL Jadida, Morocco

[3] Lab, Innovation in Science, Technology and Modeling (ISTM), Faculty of Sciences, Chouaib Doukkali University, EL Jadida, Morocco

Section five shows the results obtained by the application of the adapted cat swarm optimization to solve some benchmark instances and a discussion. Finally, a conclusion is presented.

Open shop scheduling problem

Presentation of the problem

The OSSP (Gonzalez and Sahni 1976) involves a collection of m machines $M = \{M_1, \ldots, M_m\}$ and a collection of n jobssss $\{J_1, \ldots, J_n\}$. Each job J_i ($i \in [1, n]$.) consists of m operations $J_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$, and each operation o_{ij} from b Ji executed on a dedicated machine $M_j (j \in [1, m])$ must be processed in a determined process time p_{ij}.

The OSSP consists of n jobs, J, should be processed at most m machines, M, each job I consists of $n*m$ operations $O = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$, each operation $o_{ij} = (m_{ij}, t_{ij})$ from job J_i should be executed in machine $m_{ij} \in M$ in a determinate execution time t_{ij}. One performance measure which is considered to be minimized is the total execution time of all process called makespan.

Problematic assumptions

- All operations should be processed.
- Each machine can process at most one operation at a time in determining operation process time.
- The operations in the same job cannot process simultaneously.
- Two operations of the same job cannot be processed at the same time.

Formulation of the problem

Let's consider n jobs $J = \{J_1, \ldots, J_n\}$ and m machines $\{M_1, \ldots, M_m\}$, and each job consists of $n \times m$ operations $O = \{o_{11}, o_{12}, \ldots, o_{nm}\}$, each operation t_{ij} from the job I should be executed in machine j at a determinate execution time. The solution is a schedule presented by a sequence of $n*m$ operations, numbered from 1 to $n \times m$. To calculate the makespan, the data is presented with a matrix of information Minfo that has $m*n$ four columns and four rows. The *Minfo* matrix presenting the information of each operation is described in Fig. 1. Let's use:

O_i: The name (number) of operation $\boldsymbol{i}$ in schedule ($i \in [1, (n * m)]$.). J_{o_i}: The job of selected operation o_i. M_{o_i}: The machine name where the operation o_i should be processed. T_{o_i}: The processing time of operation o_i.

Example

Let's consider the following: 3*3 OSSP, where $n = 3$, $m = 3$, $J = \{J_1, J_2, J_3\}$, $M = \{1,2,3\}$, and for every J_i in J, $Ji = \{(m_{ik}, t_{ik})\}$ for $k \in [1, 3]$,

$$\begin{pmatrix} O_1 & O_2 & O_3 & O_4 & O_5 & O_6 & O_7 & O_8 & O_9 \\ J_{o_1} & J_{o_2} & J_{o_3} & J_{o_4} & J_{o_5} & J_{o_6} & J_{o_7} & J_{o_8} & J_{o_9} \\ M_{o_1} & M_{o_2} & M_{o_3} & M_{o_4} & M_{o_5} & M_{o_6} & M_{o_7} & M_{o_8} & M_{o_9} \\ T_{o_1} & T_{o_2} & T_{o_3} & T_{o_4} & T_{o_5} & T_{o_6} & T_{o_7} & T_{o_8} & T_{o_9} \end{pmatrix}$$

Fig. 1 Informations matrix

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 3 & 1 & 2 & 1 & 3 & 2 & 1 & 2 & 3 \\ 2 & 3 & 5 & 5 & 7 & 1 & 4 & 5 & 1 \end{pmatrix}$$

Fig. 2 The schedule information matrix

$J1 = \{(3, 2), (1, 3), (2, 5)\}$
$J2 = \{(1, 5), (3, 7), (2, 1)\}$
$J3 = \{(1, 4), (2, 5), (3, 1)\}$

Let's choose a random solution: $x = \{7, 5, 3, 2, 9, 1, 6, 4, 8\}$.

The representation of information matrix is shown in Fig. 2:

To present the solution, the Gantt chart is used in Fig. 3 by respecting all OSSP problem assumptions and the total makespan of the solution x is:

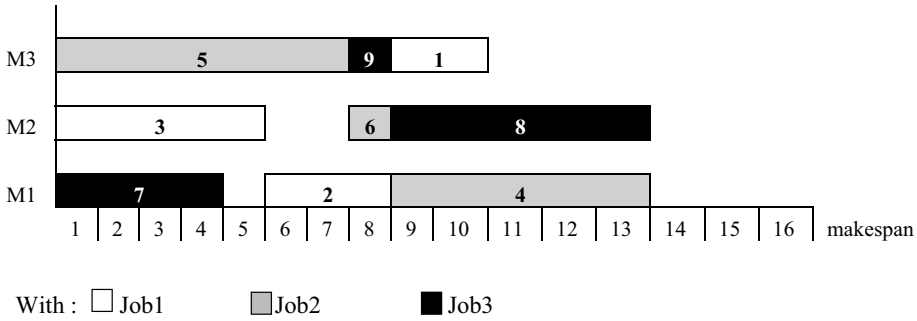
Makespan $(x) = 13$.

Cat swarm optimization algorithm

Cat swarm optimization (CSO) algorithm is a computational intelligent algorithm, inspired from the behavior of cats. The CSO algorithm was introduced by Chu and Tsai (2006). This algorithm is divided into two modes which are the seeking mode and the tracing mode. The seeking mode presents the rest mode in real life of a cat, where it spends the majority lifetime and the tracing mode, when the cat hunts a prey or any moving object. Every cat is characterized by its own position, its velocity, and the flag to identify whether the cat is in the seeking mode or the tracing mode.

The CSO algorithm proposed by Chu and Tsai (2006) was improved by some researchers to ameliorate its efficiency, as using average-inertia weight suggested by Orouskhani et al. (2011), introducing an adaptive parameter control by Wang et al. (2015), parallel cat swarm optimization by PW Tsai et al. (2008), solving combinatorial optimization problem by Bouzidi and Riffi (2013), solving the clustering problem improved by Razzaq et al. (2016), enhanced parallel cat swarm optimization based on the Taguchi method by Tsai et al. (2012). It was also extended to solve multi-objective problems in 2012 by Pradhan and Panda (2012). These improvements were applied to solve some difficult application problems, such as IIR system identification by Panda et al. (2011b), optimizing least-significant-bit

Fig. 3 Gant chart



substitution by Wang et al.(2012), optimal placement of multiple UPFC's in voltage stability enhancement under contingency by Kumar and Kalavathi (2014), direct and inverse modeling of plants by Panda et al. (2011a), single bitmap block truncation coding of color images by Cui et al. (2013), linear antenna array synthesis by Pappula and Ghosh (2014), improved metaheuristic techniques for simultaneous capacitor and DG allocation in radial distribution networks by Kawtar et al. (2015).

This paper aims to give an improved CSO algorithm to solve the OSSP problem, and to prove its efficiency, it was applied to solve some benchmark problems.

CSO to OSSP

To solve the OSSP problem by the CSO, its operators and operations (elementary and global) were enhanced. The improvements are described as follows:

Cat's parameters

In CSO algorithm, the solution presents the global best solution (Gbest) found by the cats in the swarm; for each cat, the position presents the solution that should be a schedule in OSSP problem. The velocity is used to move from a position to another; in OSSP, a novel solution is obtained by applying a set of swaps to the solution, and thus, the set of swaps is presented by the velocity, and the flag is used to know the cat mode. To sum up, the used operators of each cat in a swarm are:

Mode	Parameter	Role
SM and TM	Position	The schedule solution presented by a sequence of all operations
	Flag	The cat mode (seeking or tracing mode)
TM	Velocity	Set of couple permutation (i,j) that will be applied to a position, where i and j are a range of two velocities in the selected position
SM	SMP	Number copies of cats in the SM
	CDC	Percent length of the mutation
	SRD	First rang in the selected solution vector
	SPC	A Boolean value

Cat's process

The metaheuristic is known by its intelligent combination of two principal concepts which are exploring and exploiting. For the CSO metaheuristic, in each mode, we find the concept of exploration and exploitation. The two modes are combined intelligently with the mixture ratio (MR). The proposal CSO process respects the definition proposed by Chu and Tsai (2006), but some improvements are provided to solve the open shop scheduling problem, and these improved modes are described as follows:

Seeking mode

The seeking mode (SM) presents the cat at rest and also as being alert-looking around its environment for its next move. The position is presented by a vector of schedule, and for that the four parameters of this mode were adapted, and the new role of each one is:

The seeking mode steps can be described as follows:

1. Put j copies of the present position of the cat k, with $j = \text{SMP}$. If the value of SPC is true or $j = \text{SMP-1}$, and retain the cat as one of the candidates.
2. Generate a random value of SRD
3. If the fitness (FS) is not equal, calculate the probability of each candidate by Eq. (a), and the default probability value of each candidate is 1.
4. Perform mutation and replace the current position.

$$\text{Pi} = \frac{|\text{FS}_i - \text{FS}_{\text{min}}|}{\text{FS}_{\text{max}} - \text{FS}_{\text{min}}} \quad \text{where } 0 < i < j \tag{a}$$

where FS_i is the fitness of cat_i, FS_{max} is the maximum fitness in the swarm and the FS_{min} is the minimal fitness in the swarm.

Example

Let's consider in the SM, that the position (solution) $x = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. The size is $n = 12$,SMP = 5,SPC = True and the first value of SRD = 2.

Step 1: The program makes four copies of the selected cat position and considering the selected position as a candidate, because SPC = True.

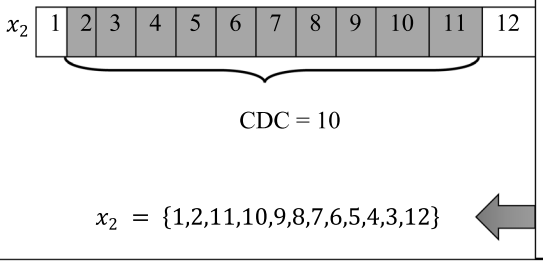
Step 2: For each copy, according to the CDC, randomly increase or decrease the current SRD percentage value, and replace the old values *b* the application of a swap between the SRD position and the second position that is (CDC + SRD) if the sum is less than 12 (the total size of the problem), else the second position is (CDC + SRD) − (12 + 1).

➢ *The first candidate :*

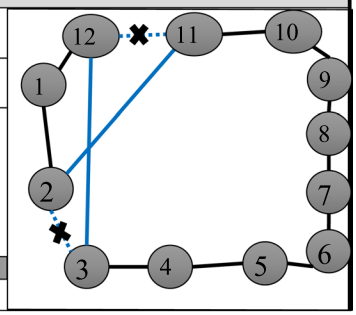
$x_1 = \{1,2,3,4,5,6,7,8,9,10,11,12\}$ is the cat position that has been taken in account since SPC=TRUE.

➢ *The second candidate :*

with SRD=2

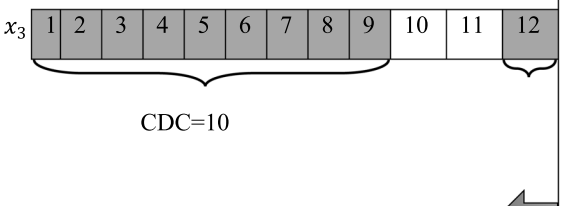


$x_2 = \{1,2,11,10,9,8,7,6,5,4,3,12\}$

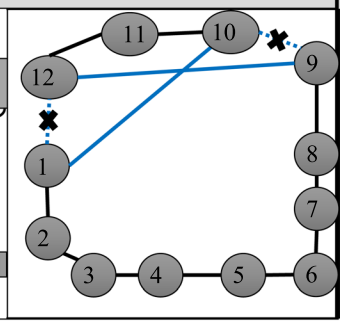


➢ *The third candidate :*

with SRD=12, et CDC = 10

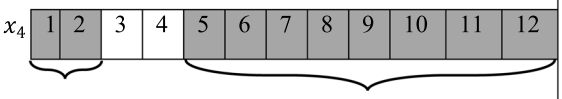


$x_3 = \{1,2,3,4,5,6,7,8,9,12,11,10\}$

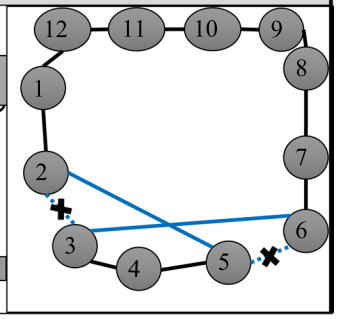


➢ *The fourth candidate:*

with SRD=5

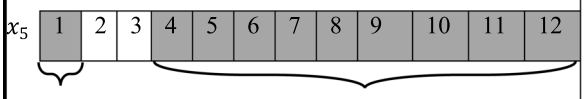


$x_4 = \{1,2,5,4,3,6,7,8,9,10,11,12\}$

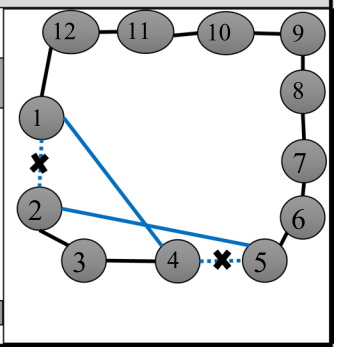


➢ *The fifth candidate:*

with SRD=4



$x_5 = \{1,4,3,2,5,6,7,8,9,10,11,12\}$



The tracing mode

The mode tracing (TM) presents the cat at the quick movement, according to its own velocity, while chasing a pray or any moving object. This section is devoted to describing the parameters, the process and elementary operations used in this mode

Tracing mode parameters

The used parameters in this mode are:

X_{best}	The best solution/position of the cat who has the best fitness value
V_k	The old speed value (current value)
V'_k	The new value velocity of the cat$_k$
w	The inertia weighted parameter
c	A constant
r	A random value in the range [0, 1]
$\boldsymbol{X'_k}$	The new value of the position of the cat$_k$
X_k	The actual position of the cat$_k$
V_k	The velocity of the cat$_k$

(a) Tracing mode process:

The process of TM is given as:

(1) Update the velocities of each cat$_k$ According to the equation:

(2) check if the velocities are of the highest order.

$$V'_k = w * V_k + r * c * \left(X_{best} - X_k\right) \quad \text{(a)}$$

(3) update the position of the cat$_k$ according to equation:

$$X'_k = X_k + V_k \quad \text{(b)}$$

Elementary operations:

The elementary operations (addition, subtraction and multiplication) used in tracing mode to solve the OSSP are not the same as those defined by Chu and Tsai (2006) to solve continuous optimization problem. The used operation is like the elementary operations defined to PSO algorithm, by Clerc (2004), to solve the combinatorial optimization problem. These operations will be performed on the velocity and the position of each cat in the TM mode.

Let's put x and x' two positions (schedules), and a velocity v represents all permutations to perform.

Let's put: $X = \{1, 2, 3, 4, 5\}$ and $v = \{(2, 5),(3, 1),(4, 3)\}$

Opposite of velocity:

Let's put $|v|$ presents the size (number of permutations couple (i, j)) of the velocity $v = \{(i_k, j_k)[k: 0 \rightarrow |v|]\}$.

The opposite is defined by: $\neg v = \{(i_k, j_k)[k:|v| \rightarrow 0]\}$,with: $v + \neg v = \emptyset$.

Addition:

This is done between a position **x** and velocity v, in order to have a new position x'.

$$x + v = x'$$

Adding operation which translates the movement represents the set of permutations to be applied to the position x to getting a new position x'.

Example

$$x + v = \{4, 5, 1, 3, 2\}$$

Subtraction (position–position):

This operation is performed between two positions to get a velocity.

$$x' - x = v$$

The subtraction is the opposite of the addition operation ($x' - x = \text{v} \Leftrightarrow x + v = x'$). In this case, by two positions x and x', the result is all permutations performed on x, to obtain x'. These pairs of permutations are the velocity v.

Example

Let's $x' = x + v = \{4, 5, 1, 3, 2\} \Rightarrow x' - x = v$

Multiplication:

This operation is performed between a real r and velocity $v = (i_k, j_k)[k:0 \rightarrow |v|]$; the result is a velocity. The different possible cases, according to the real r, are:

- If $r = 0$: $r \times v = 0$
- If $((r > 0)$ and $(\text{r} \leq 1))$: Then $r \times v = (i_k j_k)[k: 0 \rightarrow (c \times |v|)]$
- If $(r > 1)$: then separate. Decimal and integer part, $r = n + x$. Where n is the integer part of r, and x corresponds to decimal parts. And returns to each party to the previous cases.
- If $(r < 0)$: $r \times v = (-r) \times \neg v$. Now $(-r) > 0$, and you will consider one of the previous case.

Example

Let's put $v = \{(2, 5), (3, 1), (4, 3)\}$and $r = 0, 4 \Rightarrow r \times v = \{(2, 5)\}$.

The complete mode process:

The flowchart represents the description of the complete CSO process as shown in Fig. 4.

Computational results and discussion

This section presents the obtained results and the discussion of the proposal of CSO to solve some benchmark instances proposed by Taillard (1993) and Guéret and Prins (1999). This proposed adaptation is coded in Visual C++, which runs on an Ultrabook characteristic's 2.1 GHz 2.59 Ghz Intel Core i7 PC with 8G of RAM. Each instance runs for 10 times in the maximum time of 1 h.

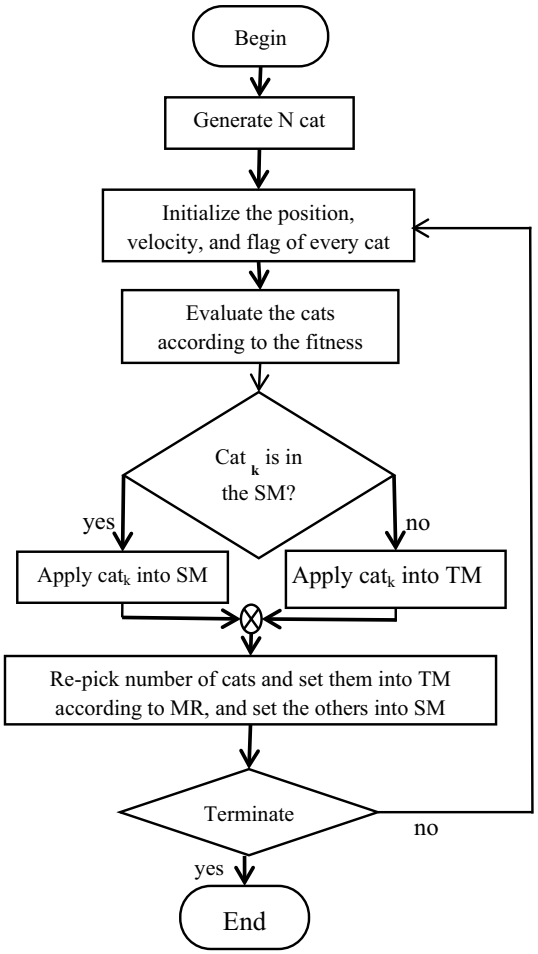


Fig. 4 CSO process flowchart

Parameter tuning

The used parameters values in CSO process are the SMP, CDC used in the seeking mode, the parameters w, r and c used in tracing mode, and the MR used in the general process.

For respecting the real life of cats, this paper dose not have to change the SMP, CDC, MR, c values and also the range of the variation of the random value r.

About the inertia weighted parameter w, this paper had used a fix parameter values such that it was consider in the proposal of CSO to solve combinatorial problem (Bouzidi and Riffi 2014b), that values were analyzed and discussed by the application to solve the TSP problem (Bouzidi and Riffi 2013). After that it was applied to solve other combinatorial problems by using this parameters values, such as the QAP (Riffi and Bouzidi 2014), JSSP (Bouzidi and Riffi 2014a) and FSSP (Bouzidi and Riffi 2015).

To resume, the used parameters values are shown in Table 1.

Evaluation of the proposed algorithm

This section presents two tables that show the collected results by the application of the adapted CSO algorithm to the benchmark ***instance***s of Taillard (1993) in Table 2, and Guéret and Prins (1999) in Table 3. For each instance, the number of job J and machines M is defined ($J \times M$), the best-known fitness solution (BKS) is found by the existing methods to solve the OSSP to the selected problem instance, the best obtained solution (BS) is found by applying CSO algorithm got in ten runs of the program, and the BS lower than the BKS is marked by * after the instance name. To prove the efficiency of the CSO, the relative percentage deviation (RPD) is calculated. And the average execution time (Aver_T) is taken into seconds.

The *RPD* is calculated by:

$$\mathrm{RPD} = \frac{\mathrm{BS} - \mathrm{BKS}}{\mathrm{BKS}}$$

Table 2 shows the obtained results by the application of proposed CSO to solve the Taillard (1993) benchmark instances.

Table 2 shows that the application of CSO to solve all proposed *Taillard* instances problems has a lower-to-negligible RPD value, and each instance problem solution is obtained in a reasonable execution time.

Table 3 shows the obtained results by the application of CSO to solve the OSSP benchmark instances proposed by Guéret and Prins (1999).

Also the CSO can find the optimal solution of many benchmark instance problems of *Guéret and Prins* that have the lower values of ***RPD***, and the solutions are obtained in a reasonable execution time. Also, when the application takes more time running, the CSO algorithm gives better solutions to all selected benchmark instances. Table 4 proves that CSO application with more time (more than 900 s) can find the best optimal solutions. Table 4 shows some benchmark instance, its size (number of jobs J and number of machines M), the best-known solution BKS for each one, the best solution (BS) found by the application of CSO method, the number of jobs J and the number of machines M; the RPD1 presents the relative percentage deviation by executing the application for a maximal time 900 s, and the RPD2 presents the relative percentage deviation by executing the application more than 900 s, and

Table 1 Used parameter's values in CSO

Parameter	*SMP*	*CDC*	*MR*	*w*	*r*	*c*
Value	5	0.8	0.3	0.7	[0,1]	2.05

Table 2 Result obtained by applying CSO to solve OSSP instances proposed by Taillard

Instance	$J\times M$	BKS	BS.	WS	RPD	Aver_T (s)
tail41	4×4	193	***193***	*193*	0.00	0.001
tail42	4×4	236	***236***	*236*	0.00	0.001
tail43	4×4	271	***271***	*271*	0.00	0.001
tail44	4×4	250	***250***	*250*	0.00	0.156
tail45	4×4	295	***295***	*295*	0.00	0.125
tail46	4×4	189	***189***	*189*	0.00	0.003
tail47	4×4	201	***201***	*201*	0.00	0.109
tail48	4×4	217	***217***	*217*	0.00	0.359
tail49	4×4	261	***261***	*261*	0.00	0.219
tail410	4×4	217	***217***	*217*	0.00	0.156
Average RPD:					**0.00**	
tail51	5×5	300	***300***	*300*	0.00	0.610
tail52	5×5	262	***262***	*262*	0.00	0.735
tail53	5×5	323	***323***	*323*	0.00	2.235
tail54	5×5	310	***310***	*310*	0.00	8.906
tail55	5×5	326	***326***	*326*	0.00	1.344
tail56	5×5	312	***312***	*312*	0.00	13.813
tail57	5×5	303	***303***	*303*	0.00	14.688
tail58	5×5	300	***300***	*300*	0.00	35.236
tail59	5×5	353	***353***	*353*	0.00	4.032
tail510	5×5	326	***326***	*326*	0.00	2.453
Average RPD:					**0.00**	
tail71	7×7	435	***435***	435	0.00	24.375
tail72	7×7	443	***443***	444	0.00	26.065
tail73	7×7	468	***468***	*477*	0.00	51.658
tail74	7×7	463	***463***	464	0.00	227.313
tail75	7×7	416	***416***	416	0.00	169.751
tail76	7×7	451	***452***	458	0.22	640.466
tail77	7×7	422	***426***	432	0.95	349.208
tail78	7×7	424	***424***	424	0.00	21.567
tail79	7×7	458	***458***	458	0.00	83.953
tail710	7×7	398	***398***	398	0.00	112.39
Average RPD:					**0.12**	
tail101	10×10	637	***645***	650	1.26	160.762
tail102	10×10	588	***588***	588	0.00	474.782
tail103	10×10	598	***599***	606	0.17	68.567
tail104	10×10	577	***577***	577	0.00	41.04
tail105	10×10	640	***640***	645	0.00	679.018
tail106	10×10	538	***538***	538	0.00	126.936
*tail107**	10×10	616	***616***	619	0.00	22.877
tail108	10×10	595	***595***	601	0.00	375.292
tail109	10×10	595	***595***	599	0.00	263.801
tail1010	10×10	596	***596***	600	0.00	112.65
Average RPD:					**0.14**	
tail151	15×15	937	***937***	937	0.00	250.323
tail152	15×15	918	***920***	926	0.22	501.669
tail153	15×15	871	***871***	871	0.00	498.766
tail154	15×15	934	***934***	934	0.00	112.994
tail155	15×15	946	***952***	959	0.63	834.63
tail156	15×15	933	***933***	933	0.00	149.578
tail157	15×15	891	***891***	898	0.00	867.539

Table 2 (continued)

Instance	$J \times M$	BKS	BS.	WS	RPD	Aver_T (s)
tail158	15×15	893	***893***	893	0.00	176.073
tail159	15×15	899	***913***	931	1.56	658.615
tail1510	15×15	902	***902***	910	0.00	891.272
Average RPD:					**0.24**	
tail201	20×20	1155	***1166***	1170	0.95	457.839
tail202	20×20	1241	***1260***	1267	1.53	618.409
tail203	20×20	1257	***1257***	1257	0.00	340.266
tail204	20×20	1248	***1253***	1259	0.40	816.467
tail205	20×20	1256	***1256***	1256	0.00	432.391
tail206	20×20	1204	***1204***	1204	0.00	806.064
tail207	20×20	1294	***1310***	1317	1.24	859.499
tail208	20×20	1177	***1210***	1234	2.80	695.772
tail209	20×20	1289	***1289***	1289	0.00	207.641
tail2010	20×20	1241	***1241***	1241	0.00	489.458
Average RPD:					**0.69**	

Best results are shown in bolditalics

Table 3 Result obtained by applying CSO to solve OSSP instances of Guéret and Prins

Instance	J×M	BKS	BS.	WS	%RPD	Aver_T (s)
gp03-01	3×3	1168	*1168*	*1168*	0.00	0.001
gp03-02	3×3	1170	*1170*	*1170*	0.00	0.001
gp03-03	3×3	1168	*1168*	*1168*	0.00	0.001
gp03-04	3×3	1166	*1166*	*1166*	0.00	0.001
gp03-05	3×3	1170	*1170*	*1170*	0.00	0.001
gp03-06	3×3	1169	*1169*	*1169*	0.00	0.001
gp03-07	3×3	1165	*1165*	*1165*	0.00	0.001
gp03-08	3×3	1167	*1167*	*1167*	0.00	0.001
gp03-09	3×3	1162	*1162*	*1162*	0.00	0.001
gp03-10	3×3	1165	*1165*	*1165*	0.00	0.001
Average RPD:					**0.00**	
gp04-01	4×4	1281	***1281***	*1281*	0.00	0.002
gp04-02	4×4	1270	***1270***	*1270*	0.00	0.002
gp04-03	4×4	1288	***1288***	*1288*	0.00	0.001
gp04-04	4×4	1261	***1261***	*1261*	0.00	0.003
gp04-05	4×4	1289	***1289***	*1289*	0.00	0.003
gp04-06	4×4	1269	***1269***	*1269*	0.00	0.001
gp04-07	4×4	1267	***1267***	*1267*	0.00	0.007
gp04-08	4×4	1259	***1259***	*1259*	0.00	0.002
gp04-09	4×4	1280	***1280***	*1280*	0.00	0.005
gp04-10	4×4	1263	***1263***	*1263*	0.00	0.001
Average RPD:					**0.00**	
gp05-01	5×5	1245	***1245***	*1245*	0.00	0.008
gp05-02	5×5	1247	***1247***	*1247*	0.00	0.003
gp05-03	5×5	1265	***1265***	*1265*	0.00	0.006
gp05-04	5×5	1258	***1258***	*1258*	0.00	0.009
gp05-05	5×5	1280	***1280***	*1280*	0.00	0.005
gp05-06	5×5	1269	***1269***	*1269*	0.00	0.009
gp05-07	5×5	1267	***1267***	*1269*	0.00	0.002

Table 3 (continued)

Instance	J×M	BKS	BS.	WS	%RPD	Aver_T (s)
gp05-08	5×5	1287	***1287***	*1287*	0.00	0.002
gp05-09	5×5	1262	***1262***	*1262*	0.00	0.003
gp05-10	5×5	1254	***1254***	*1254*	0.00	0.422
Average RPD:					**0.00**	
gp06-01	6×6	1264	***1264***	*1264*	0.00	0.281
gp06-02	6×6	1285	***1285***	*1285*	0.00	1.014
gp06-03	6×6	1255	***1255***	*1255*	0.00	0.484
gp06-04	6×6	1275	***1275***	*1275*	0.00	0.390
gp06-05	6×6	1299	***1299***	*1299*	0.00	0.187
gp06-06	6×6	1284	***1284***	*1284*	0.00	0.250
gp06-07	6×6	1290	***1290***	*1290*	0.00	0.140
gp06-08	6×6	1265	***1265***	*1265*	0.00	0.506
gp06-09	6×6	1243	***1243***	*1243*	0.00	0.203
gp06-10	6×6	1254	***1254***	*1254*	0.00	0.234
Average RPD:					**0.00**	
gp07-01	7×7	1159	***1159***	*1159*	0.00	0.406
gp07-02	7×7	1185	***1185***	*1185*	0.00	0.375
gp07-03	7×7	1237	***1237***	*1237*	0.00	0.391
gp07-04	7×7	1167	***1167***	*1167*	0.00	5.756
gp07-05	7×7	1157	***1157***	*1157*	0.00	0.797
gp07-06	7×7	1193	***1193***	*1193*	0.00	0.688
gp07-07	7×7	1185	***1185***	*1185*	0.00	1.030
gp07-08	7×7	1180	***1180***	*1180*	0.00	12.813
gp07-09	7×7	1220	***1220***	*1220*	0.00	0.437
gp07-10	7×7	1270	***1270***	*1270*	0.00	0.250
Average RPD:					**0.00**	
gp08-01	8×8	1130	***1130***	*1130*	0.00	27.373
gp08-02	8×8	1135	***1135***	*1135*	0.00	3.646
gp08-03	8×8	1110	***1110***	*1110*	0.00	76.871
gp08-04	8×8	1153	***1153***	*1153*	0.00	6.652
gp08-05	8×8	1218	***1218***	*1218*	0.00	5.288
gp08-06	8×8	1115	***1115***	*1115*	0.00	198.506
gp08-07	8×8	1126	***1126***	*1126*	0.00	214.562
gp08-08	8×8	1148	***1148***	*1148*	0.00	13.500
gp08-09	8×8	1114	***1114***	*1114*	0.00	240.140
gp08-10	8×8	1161	***1161***	*1161*	0.00	50.338
Average RPD:					**0.00**	
gp09-01	9×9	1129	***1129***	*1129*	0.00	209.037
gp09-02	9×9	1110	***1110***	*1112*	0.00	72.290
*gp09-03**	9×9	1116	***1115***	*1116*	0.00	9.028
gp09-04	9×9	1130	***1130***	*1130*	0.00	57.012
gp09-05	9×9	1180	***1180***	*1180*	0.00	3.723
gp09-06	9×9	1093	***1193***	*1093*	0.00	122.432
*gp09-07**	9×9	1091	***1090***	*1090*	0.00	89.763
*gp09-08**	9×9	1106	***1105***	*1105*	0.00	200.877
gp09-09	9×9	1123	***1123***	*1139*	0.00	84.270
gp09-10	9×9	1112	***1120***	*1121*	0.72	741.101
Average RPD:					**0.07**	
gp10-01	10×10	1093	***1111***	*1121*	1.65	273.601
gp10-02	10×10	1097	***1097***	*1097*	0.00	84.298

Table 3 (continued)

Instance	J×M	BKS	BS.	WS	%RPD	Aver_T (s)
gp10-03	10×10	1081	***1110***	*1151*	2.68	356.432
gp10-04	10×10	1083	***1090***	*1109*	0.65	428.704
gp10-05	10×10	1073	***1090***	*1108*	1.58	567.397
gp10-06	10×10	1071	***1100***	*1120*	2.71	600.557
gp10-07	10×10	1080	***1084***	*1117*	0.37	475.311
gp10-08	10×10	1095	***1117***	*1119*	2.01	465.301
gp10-09	10×10	1115	***1122***	*1127*	0.63	285.534
gp10-10	10×10	1092	***1113***	*1123*	1.92	262.229
Average RPD:					**1.42**	

Best results are shown in bolditalics

Table 4 Results in more than 900 s of CPU time

Instance	*J*×*M*	BKS	BS.	RPD2	RPD1	Aver_T (s)
in152	15×15	918	918	0.00	0.76	5391.31
in155	15×15	946	946	0.00	0.85	1763.04
in201	20×20	1155	1155	0.00	0.95	2089.60
in204	20×20	1248	1248	0.00	0.40	1248.00
in207	20×20	1294	1294	0.00	0.53	5050.18
gp10-08	10×10	1095	1097	0.18	2.01	1486.45

until it obtains the best solution, and the average time execution in ten runs (Aver_T) to seconds.

Comparison with other metaheuristics and discussion

This part aims to compare the average RPD of seven methods, including the one obtained by applying CSO to OSSP. Those other methods are two of hybrid variable neighborhood search methods (VNS), which are VNS-based curtailed fashion VNS (CLS) and VNS-based greedy local search VNS (GLS), two-phase solution method (TPSM), genetic algorithm (GA), the genetic algorithm incorporating the mutation (MGA) (Naderi and Zandieh 2014) and the electromagnetism algorithm (EA). All these methods run in a PC with 2.0 GHz Intel Core 2 Duo and 2 GB of RAM memory.

The average RPD obtained by each metaheuristic for the different problem size of two known benchmark instances (Taillard 1993; Guéret and Prins 1999) is presented as follows:

In order to do the comparative study, the best way used is the graphical representation that provides a visual display to more assess the collected average RPD results to solve the differents benchmark instances by the metaheuristics in study. For this reason, the results in Table 5 were translated into two graphs; Fig. 5 represents the variation of RPD of the seven methods for the different size of ***Taillard*** benchmark instances problems; Fig. 6 is like the first but of the benchmark instances are of Guèret and Prins.

Figure 5 shows that the CSO algorithm had the lower RPD for the different problem size instances, which means that it has more efficiency than the others.

Again, Fig. 6 shows that the CSO algorithm had the lower RPD for the different problem sizes, which means that the CSO is more efficient than other methods.

Conclusion

To conclude, this paper presented the efficiency of the cat swarm optimization algorithm to solve the theoretical problem, open shop scheduling problem, by its ability to find the best-known solution for some benchmark instances and to find the new best solutions. The comparison of the results of the CSO method to some recent existing methods to solve OSSP problem has also proved the efficiency of the CSO algorithm to solve the OSSP

Table 5 The average RPD obtained of each of the methods against two well-known benchmarks

Benchmark instances size	Algorithms						
	CSO	VNS (CLS)	VNS (GLS)	GA	MGA	TPSM	EA
Taillard							
4×4	0.00	0.00	2.26	3.53	0.00	3.89	0.42
5×5	0.00	1.30	3.51	4.49	2.48	5.54	2.20
7×7	0.55	1.67	3.17	4.76	3.11	6.08	1.68
10×10	0.16	2.66	4.46	3.68	0.13	6.35	2.10
15×15	0.67	–	–	–	–	–	2.03
20×20	1.22	4.08	5.64	3.46	1.66	8.46	1.72
Guèret and Prins							
3×3	0.00	0.00	2.97	2.41	0.00	3.21	0.75
4×4	0.00	0.00	3.98	3.05	0.00	4.80	0.89
5×5	0.00	0.48	3.90	4.50	1.89	5.06	1.47
6×6	0.00	1.56	3.49	4.93	2.34	5.18	1.99
7×7	0.00	2.15	3.10	5.33	2.49	4.73	2.08
8×8	0.00	1.26	3.77	4.91	1.53	5.05	1.36
9×9	0.06	2.32	4.54	4.83	1.06	5.10	1.64
10×10	1.94	3.05	5.69	5.43	2.80	6.82	2.74

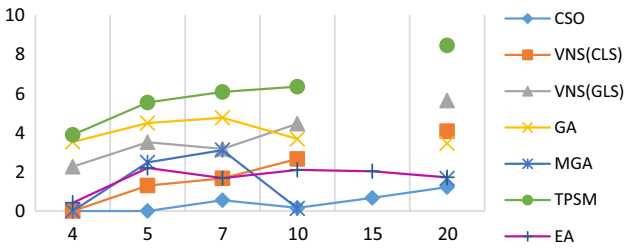


Fig. 5 Mean plots for metaheuristic versus the problem size of Taillard benchmark instances

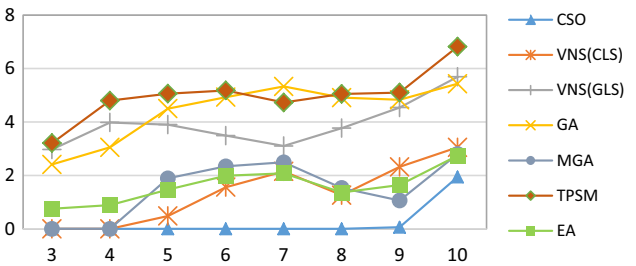


Fig. 6 Mean plots for metaheuristic versus the problem size of Guèret and Prins benchmark instances

problem. That is why, the aim is to extend the proposed metaheuristic to be applied to various real applications based on OSSP.

Compliance with ethical standards

Conflict of interest There is no conflict of interest with this research as known by the authors.

References

Bampis E, Rouskas GN (2002) The scheduling and wavelength assignment problem in optical WDM networks. J Lightwave Technol 20(5):782–789

Bhat PB, Prasanna VK, Raghavendra CS (2000) Block-cyclic redistribution over heterogeneous networks. Cluster Comput 3(1):25–34

Blum C (2005) Beam-ACO-Hybridizing ant colony optimization with beam search: an application to open shop scheduling. Comput Oper Res 32(6):1565–1591

Bouzidi A, Riffi ME (2013) Discrete cat swarm optimization to resolve the traveling salesman problem. Int J Adv Res Comput Sci Softw Eng 3(9):13–18

Bouzidi A, Riffi ME (2014) Cat swarm optimization to solve job shop scheduling problem. In: Information Science and Technology (CIST), 2014 third IEEE international colloquium in information, pp 202–205

Bouzidi A, Riffi ME (2014). Discrete cat swarm optimization algorithm applied to combinatorial optimization problems. 5th Workshop on Codes, cryptography and communication systems (WCCCS), pp. 30–34

Bouzidi A, Riffi ME (2015) Cat swarm optimization to solve flow shop scheduling problem. J Theor Appl Inf Technol 72(2):239–243

Brucker P, Hurink J, Jurisch B, Wöstmann B (1997) A branch and bound algorithm for the open-shop problem. Discrete Appl Math 76(1–3):43–59

Chu S-C, Tsai P-W (2006) Cat swarm optimization. In: Pacific Rim international conference on artificial intelligence, pp 854–858

Clerc M (2004) Discrete particle swarm optimization, illustrated by the traveling salesman problem. New optimization techniques in engineering, pp 219–239

Cui S-Y, Wang Z-H, Tsai P-W, Chang C-C, Yue S (2013) Single bitmap block truncation coding of color images using cat swarm optimization. In: Recent advances in information hiding and applications, pp. 119–138

Dell'Amico M, Martello S (1996) Open shop, satellite communication and a theorem by Egerváry (1931). Oper Res Lett 18(5):207–211

Gonzalez T, Sahni S (1976) Open shop scheduling to minimize finish time. J ACM 23(4):665–679

Guéret C, Prins C (1999) A new lower bound for the open-shop problem. Ann Oper Res 92:165–183

Huang Y-M, Lin J-C (2011) A new bee colony optimization algorithm with idle-time-based filtering scheme for open shop-scheduling problems. Expert Syst Appl 38(5):5438–5447

Iyengar V, Chakrabarty K (2002) System-on-a-chip test scheduling with precedence relationships, preemption, and power constraints. IEEE Trans Comput Aided Des Integr Circuits Syst 21(9):1088–1094

Kanwar N, Gupta N, Niazi K, Swarnkar A (2015) Improved meta-heuristic techniques for simultaneous capacitor and DG allocation in radial distribution networks. Int J Electr Power Energy Syst 73:653–664

Kumar GN, Kalavathi MS (2014) Cat swarm optimization for optimal placement of multiple UPFC's in voltage stability enhancement under contingency. Int J Electr Power Energy Syst 57:97–104

Liaw C-F (1999a) A tabu search algorithm for the open shop scheduling problem. Comput Oper Res 26(2):109–126

Liaw C-F (1999b) Applying simulated annealing to the open shop scheduling problem. IIE Trans 31(5):457–465

Lucas KT (2010) Parallel enumeration sort on OTIS-hypercube. In: International conference on contemporary computing, pp 21–31

Naderi B, Zandieh M (2014) Modeling and scheduling no-wait open shop problems. Int J Prod Econ 158:256–266

Orouskhani M, Mansouri M, Teshnehlab M (2011) Average-inertia weighted cat swarm optimization. In: International conference in swarm intelligence, 321–328

Panda G, Pradhan PM, Majhi B (2011a) Direct and inverse modeling of plants using cat swarm optimization. Handb Swarm Intell 8:469–485

Panda G, Pradhan PM, Majhi B (2011b) IIR system identification using cat swarm optimization. Expert Syst Appl 38(10):12671–12683

Pappula L, Ghosh D (2014) Linear antenna array synthesis using cat swarm optimization. AEU-Int J Electron Commun 68(6):540–549

Pradhan PM, Panda G (2012) Solving multiobjective problems using cat swarm optimization. Expert Syst Appl 39(3):2956–2964

Prins C (2000) Competitive genetic algorithms for the open-shop scheduling problem. Math Methods Oper Res 52(3):389–411

Rahmani Hosseinabadi AA, Vahidi J, Saemi B, Sangaiah AK, Elhoseny M (2018) Extended genetic algorithm for solving open-shop scheduling problem. Soft Comput. https://doi.org/10.1007/s00500-018-3177-y

Razzaq S, Maqbool F, Hussain A (2016) Modified cat swarm optimization for clustering. In: International conference on brain inspired cognitive systems, pp 161–170

Riffi ME, Bouzidi A (2014) Discrete cat swarm optimization for solving the quadratic assignment problem. Int J Soft Comput Softw Eng 4(6):85–92

Sha D, Hsu C-Y (2008) A new particle swarm optimization for the open shop scheduling problem. Comput Oper Res 35(10):3243–3261

Suel T (1995) Permutation routing and sorting on meshes with row and column buses. Parallel Process Lett 5(1):63–80

Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285

Tsai P-W, Pan J-S, Chen S-M, Liao B-Y, Hao S-P (2008) Parallel cat swarm optimization. 2008 international conference on machine learning and cybernetics, vol 6, pp 3328–3333

Tsai P-W, Pan J-S, Chen S-M, Liao B-Y (2012) Enhanced parallel cat swarm optimization based on the Taguchi method. Expert Syst Appl 39(7):6309–6319

Wang J (2015) A new cat swarm optimization with adaptive parameter control. In: Sun H, Yang CY, Lin CW, Pan JS, Snasel V, Abraham A (eds) Genetic and evolutionary computing. Advances in intelligent systems and computing, vol 329. Springer, Cham, pp 69–78

Wang Z-H, Chang C-C, Li M-C (2012) Optimizing least-significant-bit substitution using cat swarm optimization strategy. Inf Sci 192:98–108