Production Systems and Information Engineering
Volume 12 (3), pp. 11-19
doi: https://doi.org/10.32968/psaie.2024.3.2

Fitness Landscape Analysis of the Cat Swarm Optimization in the Flow Shop Scheduling Problem

ANITA AGÁRDI
University of Miskolc, Hungary
Institute of Information Technology
agardianita@iit.uni-miskolc.hu

Abstract.
This paper presents a population algorithm, the Cat Swarm Optimization (CSO), and solves a scheduling problem, the Flow Shop Scheduling (FSS), with the algorithm. Cat Swarm Optimization is a population metaheuristic. It was developed for continuous tasks, but Flow Shop Scheduling is a discrete optimization task, so the algorithm had to be discretized. The article presents the test results and also performs analytical analyses using different fitness landscape analysis techniques.

Keywords: Cat Swarm Optimization, fitness landscape, Flow Shop Scheduling Problem

1. Introduction

This article investigates the effectiveness of the Cat Swarm Optimization algorithm. The algorithm is originally suitable for continuous tasks, so the discretization process is also presented. Over the years, many metaheuristics have appeared which are used to solve continuous or discrete optimization tasks. Researchers are constantly improving their efficiency, performing test runs on various discrete or continuous benchmark data. Another more and more common technique these days is to analyze the effectiveness of optimization algorithms not only with test runs but also with fitness landscape techniques, which provide analytical analysis.
In the following, the article presents the literary background, where the Cat Swarm Optimization algorithm is detailed.

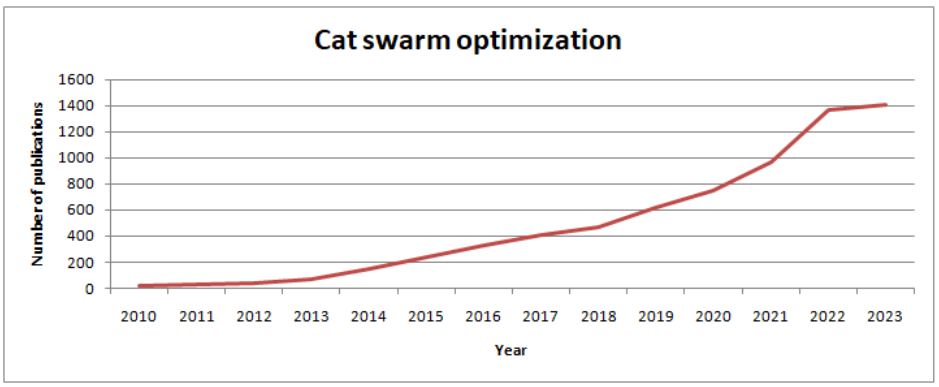

Figure 1. Cat Swarm Optimization keyword result in Google Scholar

The number of Cat Swarm Optimization publications shows a drastic increase from 2010. While only 17 articles were published on the topic in 2010, 238 articles were published in 2015, 749 articles in 2020, and 1410 articles in 2023, according to Google Scholar.
The Cat Swarm Optimization algorithm has been used for many continuous and discrete optimization tasks so far, such as [1]:

- minimization of electricity costs in the payment system
- economic load dispatch
- unit commitment
- reactive power dispatch problem
- classification problems
- optimization of the network structure and learning parameters of the Artificial Neural Network
- electric power consumption prediction
- object-tracking system
- image segmentation
- Traveling Salesman Problem
- Job Shop Scheduling
- Quadratic Assignment Problem
- multi-document summarization problem

Even though it is a relatively new algorithm, researchers have already used it in many topics. The rest of the article is structured as follows: the second chapter introduces Cat Swarm Optimization and the Flow Shop Scheduling Problem. In the third chapter, the discretization of the Cat Swarm Optimization algorithm and the test results are presented. The fourth chapter is the conclusion.

2. Cat Swarm Optimization and Flow Shop Scheduling Problem

2.1. Cat Swarm Optimization [2]

The algorithm is a population search procedure. The developer of the algorithm transformed the natural behavior of cats by introducing two modes: search mode, when the cat is resting, and tracking mode when the cat is hunting.
The CSO algorithm consists of a search mode and a tracking mode, which are combined by a mixing ratio.
The parameters used in the CSO algorithm are:

- SMP: number of cats in search mode
- CDC: the search range of the selected dimension.

- SRD: number of dimensions to be modified.
- SPC: a boolean value indicating whether the cat is currently in searching or hunting mode

1. Searching mode

It shows that the cat is at rest, observing.
The search method is as follows:

1. Creating j copies of the current position of k cats, where $j = SMP$. If the value of SPC is true or $j = SMP - 1$, then the cat is in search mode.
2. Generating the SRD value randomly
3. The probability of each possible solution must be calculated using the equation below, the default probability value for each solution is 1.
4. Performing a mutation, and if a better solution is obtained during the mutation, then replacing the current solution:

$$P_i = \frac{|FS_i - FS_{max}|}{FS_{max} - FS_{min}}$$

2. Tracking mode:
This is the cat's way of hunting, where the cat chases its prey or any moving object. The process description in this mode is as follows:

1. It is necessary to update the speed of each current cat(s) according to the following equation:

$$v_k' = w * v_k + r * c * (x_{best} - x_k)$$

where:
v_k' : The new speed value
w: inertia weight
x_{best}: the best position of the swarm (set of solutions).
v_k: the old speed value (current value).
c: constant.
r: random value in the range $[0, 1]$.

2. Ranking the elements of the population (based on fitness value)
3. The cat k. position needs to be updated as follows:

$$x_k' = x_k + v_k$$

where:
x_k': new position of cat k.
x_k: actual position of cat k.
v_k: velocity of cat k.

The steps of the algorithm are as follows:

- Creation of N cats (creation of the population, usually with randomly generated elements)
- Initialization of the flag, speed, and position for each cat
- Selecting the best cat in the pack (the best element of the population)
- The following steps are repeated until the stop condition:
    - Performing the following steps for individual cats (individual elements of the population).
        - If the cat is in search mode, then the search mode step is applied.
        - If the cat is in hunting mode, then hunting mode is used.
- The algorithm returns with the best solution.

2.2. Flow Shop Scheduling Problem

The Flow Shop Scheduling Problem [3] is a discrete optimization problem. The numbers of machines and jobs are given during the task. The durations required to complete the work are also given for each machine. All work must be done on all machines. The goal is the makespan minimization. The task can also be represented by a permutation, where the elements of the permutation represent the individual jobs, and the sequence represents the order in which they take place on the individual machines.
The rules are also as follows:

- If a machine has started a job, it must also finish it
- If a machine has started a job, it cannot do another job in the meantime, a machine can only do one job at a time
- If the machine cannot yet perform the next job in the sequence, it waits for the next job
- The works start at time 0
- The processing time also includes the setup time in the case of work machines

The above-mentioned version of the Flow Shop Scheduling Problem is the classic task, however, over the years, many versions of the task have been developed, which adapt as best as possible to the needs and production processes of today. Highlighting some of them:

- Flow shop scheduling with heterogeneous workers [4]
- Non-Permutation Flow Shop Scheduling Problem with Time Couplings [5]
- Flow Shop Scheduling Problem with Minimal and Maximal Idle Time [6]
- non-permutation flow-shop scheduling problem [7]
- flow shop scheduling problem with availability constraints [8]
- blocking flow shop scheduling problem [9]
- flow shop scheduling problem with maintenance activities integrated [10]

3. Discrete Cat Swarm Optimization test results

3.1. Discrete Cat Swarm Optimization

The algorithm was created for continuous tasks, so I needed to discretize the algorithm. I will now present the steps for this:

- Initializing the population randomly. Each cat will represent a permutation.
- The following steps must be repeated until the stop condition is not met:
    - Determining for each cat whether they are in searching mode or hunting mode
    - If the given cat is in search mode, then
        - Perform a 2-opt step on the given solution up to the CDC value
    - If the given cat is in hunting mode, then
        - Determination of a new speed value for the given cat, which is calculated using the following formula:

$$v'_k = |w * v_k + rand * c * (x_{best} - x_k)|$$

Where
v'_k : The new speed value
w: inertia weight

$rand$: random number
c: constant parameter value
x_{best}: the best solution so far in the population
x_k:the fitness value of the cat

- Generating v'_k random solutions. If a randomly generated solution is better than the current solution, it is accepted as the current solution

- The algorithm returns with the best solution

3.2. Discrete Cat Swarm Optimization test results

First, the Discrete Cat Swarm Optimization algorithm was tested on the Taillard [11] dataset. Test results were made from Ta001 to Ta020. I also compared the results of CSO with the results of other algorithms, and the relative performances from them are also included in the table.

- HMM-PFA: Hidden Markov Model - Probability Flow Analysis
- HGA: Hybrid Genetic Algorithm
- IIGA: Improved Immune Genetic Algorithm
- DSOMA: Differential Self-Organizing Migrating Algorithm
- HGSA: Hybrid Gravitational Search Algorithm
- IWO: Invasive Weed Optimization

Table 1. Test result comparisons

		Relative performance					
Instance	**CSO**	**HMM-PFA [12] %**	**HGA [12] %**	**IIGA [12] %**	**DSOMA [12] %**	**HGSA [12] %**	**IWO [13] %**
Ta001	1297	114.57	111.72	114.57	105.94	102.08	107.09
Ta002	1366	111.86	106.88	111.86	103.07	105.56	-
Ta003	1145	127.51	121.05	127.51	111.79	95.9	-
Ta004	1362	116.59	111.67	116.59	106.31	107.86	-
Ta005	1283	112.94	109.35	112.94	104.52	100.62	-
Ta006	1244	119.05	114.95	119.05	109.57	111.82	-
Ta007	1260	117.7	115.95	117.7	109.6	103.1	-
Ta008	1273	116.42	112.57	116.42	108.33	101.49	-
Ta009	1279	114.86	109.3	114.86	107.35	102.11	-
Ta010	1171	117.59	113.07	117.59	109.56	105.29	-
Ta011	1684	121.38	116.09	119.42	100.83	101.72	131.06
Ta012	1773	122.17	119.74	122.17	103.38	96.9	-
Ta013	1613	120.27	118.54	120.27	103.91	96.4	-
Ta014	1489	121.63	119.68	121.63	103.83	101.81	-
Ta015	1546	125.03	125.03	125.03	104.59	101.75	-
Ta016	1494	126.64	122.29	126.64	106.43	97.52	-
Ta017	1575	124.63	123.43	124.63	102.98	102.98	-
Ta018	1651	124.59	121.5	124.59	104.85	105.94	-
Ta019	1675	117.79	113.91	117.79	104.3	96.96	-
Ta020	1701	120.58	117.64	120.58	104.76	101.23	-

Based on the test results, CSO gave 1297 fitness result for Ta001, which was 14% better than HMM-PFA and IIGA, 11% better than HGA, 5% better than DSOMA, and 2% better than the IWO. I could only compare the IWO algorithm with a total of 2 data sets. For Ta002, CSO gave 1366 fitness result, which was 11% better than HMM-PFA and IIGA, 6% better than HGA, and 3% better than DSOMA. For

Ta003, CSO gave a fitness value of 1145, which was 27% better than HMM-PFA and IIGA, 21% better than HGA, 11% worse than DSOMA, and 4% worse than the HGSA. The algorithm gave 1362 fitness result for Ta004, which was 16% better than HMM-PFA and IIGA, 11% better than HGA, 6% better than DSOMA, and 7% better than HGSA. From the table, we can see that in only a few cases we got worse results using CSO compared to other algorithms. These Ta003, Ta012, Ta013, Ta016, and Ta019 benchmark data and the HGSA algorithm gave better results on these data.

Table 2. Test result comparisons

Algorithm	**Number of data rows (on which the comparisons were made)**	**Number of better results**
HMM-PFA	20	20
HGA	20	20
IIGA	20	20
DSOMA	20	20
HGSA	20	15
IWO	2	2

Table 2 also shows that almost all test results are better than the other comparison algorithms, except for HGSA, where the Cat Swarm algorithm is better than 15 out of 20 runs. The IWO algorithm was run on 2 benchmark data only.

3.3. Cat Swarm Optimization(CSO) algorithm fitness landscape results

This subsection presents the results of the search space for the CSO algorithm. When analyzing the search space, I examined the results of the iterations given by the algorithm. I used the following search space techniques [14]:

- Fitness values
- Average of fitness
- distances
- Average of Hamming
- distances
- Average of basic swap
- sequence distances
- Fitness distances of the best solution
- Hamming distances of the best solution
- Basic swap sequence distances of the best solution
- Cost density
- Fitness distance of filtered global optima
- Hamming distance of filtered global optima
- Basic swap sequence distance of filtered global optima

Table 3. Cat Swarm Optimization fitness landscape results

		Ta001	
	Distance	**LB**	**UB**
Fitness values		1300	1325
Average fitness distances	**Fitness**	2.27	22.73
Average of Hamming distances	**Hamming**	1.9	17.46
Average of basic swap sequence distances	**BSS**	1.63	16.17
Fitness distances of the best solution	**Fitness**	2.27	22.73
Hamming distances of the best solution	**Hamming**	1.9	17.46

Basic swap sequence distances of the best solution	**BSS**	1.63	16.17
Cost density		3.0	90.0
Fitness distance of filtered global optima	**Fitness**	2.27	22.73
Hamming distance of filtered global optima	**Hamming**	1.9	17.46
Basic swap sequence distance of filtered global optima	**BSS**	1.63	16.17

		Ta002	
	Distance	**LB**	**LB**
Fitness values		1367	1367
Average fitness distances	**Fitness**	0.06	0.06
Average of Hamming distances	**Hamming**	0.96	0.96
Average of basic swap sequence distances	**BSS**	0.72	0.72
Fitness distances of the best solution	**Fitness**	0.06	0.06
Hamming distances of the best solution	**Hamming**	0.96	0.96
Basic swap sequence distances of the best solution	**BSS**	0.72	0.72
Cost density		6.0	6.0
Fitness distance of filtered global optima	**Fitness**	0.06	0.06
Hamming distance of filtered global optima	**Hamming**	0.96	0.96
Basic swap sequence distance of filtered global optima	**BSS**	0.72	0.72

		Ta003	
	Distance	**LB**	**LB**
Fitness values		1160	1160
Average fitness distances	**Fitness**	3.29	3.29
Average of Hamming distances	**Hamming**	7.03	7.03
Average of basic swap sequence distances	**BSS**	5.92	5.92
Fitness distances of the best solution	**Fitness**	3.29	3.29
Hamming distances of the best solution	**Hamming**	7.03	7.03
Basic swap sequence distances of the best solution	**BSS**	5.92	5.92
Cost density		3.0	3.0
Fitness distance of filtered global optima	**Fitness**	3.29	3.29
Hamming distance of filtered global optima	**Hamming**	7.03	7.03
Basic swap sequence distance of filtered global optima	**BSS**	5.92	5.92

		Ta004	
	Distance	**LB**	**LB**
Fitness values		1375	1375
Average fitness distances	**Fitness**	0.95	0.95
Average of Hamming distances	**Hamming**	0.35	0.35
Average of basic swap sequence distances	**BSS**	0.29	0.29
Fitness distances of the best solution	**Fitness**	48.05	48.05
Hamming distances of the best solution	**Hamming**	0.35	0.35
Basic swap sequence distances of the best solution	**BSS**	0.29	0.29
Cost density		1.0	1.0
Fitness distance of filtered global optima	**Fitness**	0.95	0.95
Hamming distance of filtered global optima	**Hamming**	0.35	0.35
Basic swap sequence distance of filtered global optima	**BSS**	0.29	0.29

		Ta005	
	Distance	**LB**	**LB**
Fitness values		1296	1296
Average fitness distances	**Fitness**	3.44	3.44
Average of Hamming distances	**Hamming**	9.26	9.26
Average of basic swap sequence distances	**BSS**	7.59	7.59
Fitness distances of the best solution	**Fitness**	3.44	3.44
Hamming distances of the best solution	**Hamming**	9.26	9.26
Basic swap sequence distances of the best solution	**BSS**	7.59	7.59

Cost density		1.0	1.0
Fitness distance of filtered global optima	**Fitness**	3.44	3.44
Hamming distance of filtered global optima	**Hamming**	9.26	9.26
Basic swap sequence distance of filtered global optima	**BSS**	7.59	7.59

I also ran the Cat Swarm Optimization results from Ta001 to Ta005. For Ta001, I got fitness scores between 1325 and 1300, and fitness distance averages between 22 and 2. For Ta002 they are between 1368 and 1367, there is no big improvement here because the average fitness distances are also between 0 and 1. For Ta003, the fitness values are between 1423 and 1375, and the average distances are between 21 and 3. For Ta004, the algorithm resulted in fitness values between 1423 and 1375. And the average fitness values were between 48 and 1. The individual iterations brought significant improvement to this data set as well. For the Ta005 dataset, the iteration yielded fitness values between 1324 and 1296. Here, the average fitness values are between 20 and 3, which is also a good improvement. In terms of Hamming and Basic Swap Sequence distances, the algorithm is characterized by a small lower bound and medium average upper bound.

4. Conclusion and future research direction

This article presents the Cat Swarm Optimization (CSO) algorithm, which was originally developed for continuous optimization tasks, so this algorithm requires a discretization process. In the article, the solution of the Flow Shop Scheduling (FSS) task was shown with the CSO algorithm. The test results and their comparison with six other algorithms were presented, of which the CSO algorithm gave better results in all cases for five algorithms, and for one algorithm it gave better results in 15 cases out of 20 data. The CSO algorithm was analyzed analytically in the article, using different fitness landscape analysis techniques, such as fitness distance, basis swap sequence distance, and Hamming distance. Based on the results, it can be said that the Discrete Cat Swarm Optimization algorithm effectively solved the Flow Shop Scheduling (FSS) problem. Future research areas can be the implementation and analysis of other metaheuristic algorithms with fitness landscape techniques, and comparison of the running results.

Acknowledgment.
„Supported by the ÚNKP-23-4-II. New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.”

References

[1] Ahmed, A. M., Rashid, T. A., & Saeed, S. A. M. (2020). Cat swarm optimization algorithm: a survey and performance evaluation. Computational intelligence and neuroscience, 2020. https://doi.org/10.1155/2020/4854895

[2] Bouzidi, A., & Riffi, M. E. (2014, October). Cat swarm optimization to solve job shop scheduling problem. In 2014 Third IEEE International Colloquium in Information Science and Technology (CIST) (pp. 202-205). IEEE. https://doi.org/10.1109/CIST.2014.7016619

[3] Kulcsár, G., & Erdélyi, F. (2007). A new approach to solve multi-objective scheduling and rescheduling tasks. International Journal of Computational Intelligence Research, 3(4), 343-351.

[4] Benavides, A. J., Ritt, M., & Miralles, C. (2014). Flow shop scheduling with heterogeneous workers. European Journal of Operational Research, 237(2), 713-720. https://doi.org/10.1016/j.ejor.2014.02.012

[5] Idzikowski, R., Rudy, J., & Gnatowski, A. (2021). Solving Non-Permutation Flow Shop Scheduling Problem with Time Couplings. Applied Sciences, 11(10), 4425. https://doi.org/10.3390/app11104425

[6] Rudy, J. (2021). Parallel Makespan Calculation for Flow Shop Scheduling Problem with Minimal and Maximal Idle Time. Applied Sciences, 11(17), 8204. https://doi.org/10.3390/app11178204

[7] Rossit, D. A., Tohmé, F., & Frutos, M. (2018). The non-permutation flow-shop scheduling problem: a literature review. Omega, 77, 143-153. https://doi.org/10.1016/j.omega.2017.05.010

[8] Aggoune, R. (2004). Minimizing the makespan for the flow shop scheduling problem with availability constraints. European Journal of Operational Research, 153(3), 534-543. https://doi.org/10.1016/S0377-2217(03)00261-3

[9] Miyata, H. H., & Nagano, M. S. (2019). The blocking flow shop scheduling problem: A comprehensive and conceptual review. Expert Systems with Applications, 137, 130-156. https://doi.org/10.1016/j.eswa.2019.06.069

[10] Branda, A., Castellano, D., Guizzi, G., & Popolo, V. (2021). Metaheuristics for the flow shop scheduling problem with maintenance activities integrated. Computers & Industrial Engineering, 151, 106989. https://doi.org/10.1016/j.cie.2020.106989

[11] E. Taillard, "Benchmarks for basic scheduling problems", EJOR 64(2):278-285, 1993. https://doi.org/10.1016/0377-2217(93)90182-M

[12] Wei, H., Li, S., Jiang, H., Hu, J., & Hu, J. (2018). Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion. Applied Sciences, 8(12), 2621. https://doi.org/10.3390/app8122621

[13] Zhou, Y., Chen, H., & Zhou, G. (2014). Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem. Neurocomputing, 137, 285-292. https://doi.org/10.1016/j.neucom.2013.05.063

[14] Agárdi, A., Kovács, L., & Bányai, T. (2021). The fitness landscape analysis of the ant colony system algorithm in solving a vehicle routing problem. ACADEMIC JOURNAL OF MANUFACTURING ENGINEERING, 19(2).