Production Systems and Information Engineering
Volume 12 (1), pp. 41–48.
https://doi.org/10.32968/psaie.2024.1.4

EFFICIENCY ANALYSIS OF THE SINE-COSINE ALGORITHM IN SOLVING THE FLOW SHOP SCHEDULING PROBLEM

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

Abstract: This article presents the efficiency of the Sine-Cosine Algorithm (SCA) and the analysis of its search space. The Sine-Cosine Algorithm is originally a continuous optimization algorithm, which was discretized in this article because the article solves the Flow Shop Scheduling (FSS) Problem. The article also presents the continuous Sine-Cosine Algorithm and its discretized modification. The article presents the Flow Shop Scheduling problem and then the test results.

Keywords: *Sine-Cosine Algorithm, fitness landscape, Flow Shop Scheduling*

1. Introduction

This article presents a relatively new optimization algorithm, the Sine-Cosine Algorithm (SCA), and solves a discrete optimization problem with this algorithm. The discrete optimization task is Flow Shop Scheduling (FSP).

The Sine-Cosine Algorithm was first published by Seyedali Mirjalili in 2016 in the article "Sine Cosine Algorithm for Constrained Optimization Problems" [1].

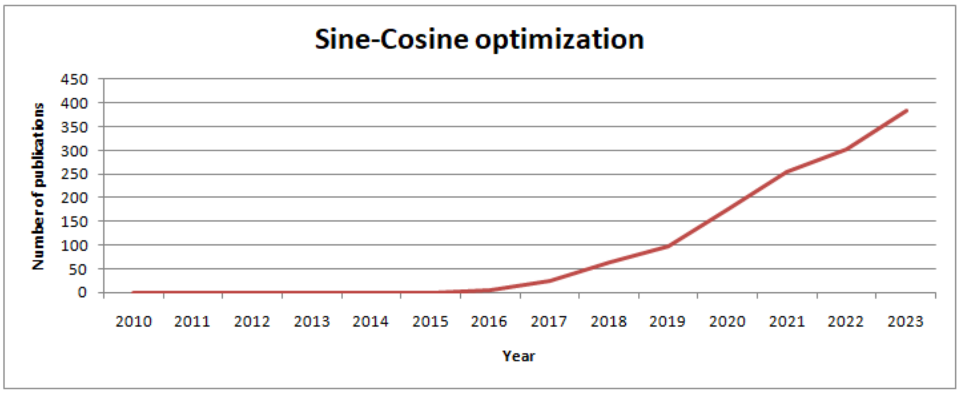

Figure 1. Sine-Cosine Optimization algorithm keywords in Google Scholar

It can be seen that since its publication year (2016), more and more researchers have published the algorithm in their articles.

5 articles were published in 2016, 177 in 2020, and 385 in 2023.

Several versions of the algorithm were developed, which were adapted to continuous and discrete tasks, and the researchers published numerous improvements, tunings, and hybridizations. Some of these algorithms are highlighted below [2]:

- Adaptive Sine-Cosine Algorithm
- Fuzzy Sine-Cosine Algorithm
- Mutation Sine-Cosine Algorithm
- Orthogonal-based Sine-Cosine Algorithm
- Learning Sine-Cosine Algorithm
- Binary Sine-Cosine Algorithm
- Chaotic Sine-Cosine Algorithm
- Levy Flight Sine-Cosine Algorithm
- Opposition-based Sine-Cosine Algorithm
- Learning Sine-Cosine Algorithm
- Improved Sine-Cosine Algorithm

Some application area of the Sine-Cosine Algorithm:

- Optimal power flow solution in power systems [3]
- optimization for automatic voltage regulator system [4]
- image segmentation [5]
- engineering design problems [6]
- text categorization [7]
- object tracking [8]
- feature selection method for data mining tasks [9]

The article is further structured as follows: the second chapter describes the Flow Shop Scheduling Problem and the Sine-Cosine Algorithm. The third chapter contains the test results. The fourth chapter is the conclusions and future research directions.

2. Flow Shop Scheduling and Sine-Cosine Algorithm

2.1. Flow Shop Scheduling

Flow Shop Scheduling [10] is a discrete production scheduling problem. According to the classical problem, n tasks and m machines are given. All tasks must be completed on all machines. The task is to determine the sequence of jobs so that the makespan is minimal. According to the task, if a machine has started a job, it must also finish it, and only then can another job follow. Production starts from time 0. Over the years, many versions of the task have developed to meet the largest and most complex tasks of industrial needs. I would like to highlight some of these versions:

- hybrid flow shop scheduling problem [10]
- distributed permutation flow shop scheduling problem with worker flexibility [11]
- no-wait flexible flow shop scheduling problem [12]

- hybrid flow shop scheduling problem with limited human resource constraint [13]
- flow shop scheduling problem with maintenance activities integrated [14]
- flow shop scheduling problem with sequence-independent setup time [15]
- flow shop scheduling problem with missing operations [16]
- flow shop scheduling problem with total completion time minimization [17]

2.2. Sine-Cosine Algorithm [18]

Sine-cosine optimization was first developed for continuous tasks. It is an optimization method that maintains a population of solutions performs operations on the populations. It uses sine and cosine mathematical functions to make small or large changes on the population.

The algorithm works with the following update equation:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * \sin(r_2) * \left|r_3 P_i^t - X_i^t\right| & r_4 < 0.5 \\ X_i^t + r_1 * \cos(r_2) * \left|r_3 P_i^t - X_i^t\right| & r_4 \geq 0.5 \end{cases}$$

Parameters of type r of the algorithm:r_1, r_2, r_3, r_4. X_i^t means the position of solution i in the iteration t.

r_1 can be calculated using the following formula:

$$r_1 = a - \frac{a * t}{T}$$

where T is the maximum number of iterations and a is a constant, the parameter of the algorithm.

$$r_2 = 2 * \pi * rand$$
$$r_3 = b * rand \quad b > 1$$

where in the equations b is a constant.

The r_4 is a random number in the interval $[0,1]$.

As the formulas above indicate, there are four basic parameters in SCA:r_1, r_2, r_3, r_4. Az r_1 parameter indicates the particular region within which solutions are searched. The r_2 parameter determines how far the movement should be towards or away from the target. The r_3 parameter gives a random weight to the solution that is important ($r_3 > 1$) or not important ($r_3 < 1$). The r_4 parameter equally prefers the sine and cosine function according to the equation, so it is random whether the sine or cosine function is currently applied.

The steps of the algorithm for a continuous problem are as follows:

- Initialization of the input parameters
- Initialization of population elements (most often with randomly generated solutions) and evaluation.
- The following steps are repeated until the termination condition is not met:
    - Determination of the r_1, r_2, r_3 parameters, and $r_4 = rand(0,1)$

- Calculation of the following elements of the population using the formula, where P_i^t the best position of individual i in iteration t:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * \sin(r_2) * \left|r_3 P_i^t - X_i^t\right| & r_4 < 0.5 \\ X_i^t + r_1 * \cos(r_2) * \left|r_3 P_i^t - X_i^t\right| & r_4 \geq 0.5 \end{cases}$$

- The algorithm returns with the best element of the population

The above algorithm is suitable for solving continuous tasks, so the algorithm had to be converted to discrete problems so that it could be used for specific production scheduling.

2.3. Discrete Sine-Cosine Algorithm

The discussed Flow Shop Scheduling problem is a discrete production scheduling task, so the discretization of the Sine-Cosine Algorithm was necessary. The steps of the algorithm for a discrete problem are as follows:

- Initialization of the input parameters
- Initialization of population elements (most often with randomly generated solutions) and evaluation of that. Here, we assign each X_i^t to the populations, which is a continuous number, this is also randomly generated.
- The following steps are repeated until the termination condition is not met:
  - Calculation of the r_1, r_2, r_3 parameters, $r_4 = rand(0,1)$
  - Calculation of the following elements of the population using the following formula:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * \sin(r_2) * \left|r_3 P_i^t - X_i^t\right| & r_4 < 0.5 \\ X_i^t + r_1 * \cos(r_2) * \left|r_3 P_i^t - X_i^t\right| & r_4 \geq 0.5 \end{cases}$$

  - For the given solution, we perform the 2-opt step X_i^{t+1}-times.
  - If the best solution obtained with this way is better than the given element of the population, then the given element of the population is replaced by the best result of the 2-opt steps
- The algorithm returns with the best element of the population

3. Test results

This section contains the results for the Taillard [19] benchmark dataset, and also compares the effectiveness of the Sine-Cosine Algorithm with other algorithms, published and implemented by other researchers. The test data is from Ta001 to Ta030.

In the case of IWO, we can only find test results in a few cases. For Ta001, SCA gave a fitness value of 1426, which is 4% better than the HMM-PFA algorithm, 1% better than HGA, 4% better than IIGA, 3% worse than DSOMA, 7% worse than HGSA and 2% worse than IWO. For Ta002, SCA gave a fitness value of 1454, which is 5% better than HMM-PFA, 0.4% better than HGA, 5% better than IIGA, and 3% worse than DSOMA, and 0.8% worse than HGSA. For Ta003, the algorithm gave 1334 fitness value. This value is 9% better than HMM-PFA, 3% better than HGA, 9% better than IIGA, 4% worse than DSOMA, 17% worse than HGSA. For Ta004, SCA gave a result of 1535, which is 3% better than HMMPFA, 0.9% worse than HGA, 3% better than IIGA, 5% worse than DSOMA, 4% worse than HGSA.

Table 1. Test results

		Relative performance					
Instance	**SCA**	**HMM-PFA [20] %**	**HGA [20] %**	**IIGA [20]%**	**DSOMA [20] %**	**HGSA [20] %**	**IWO [21] %**
Ta001	1426	104.21	101.61	104.21	96.35	92.85	97.41
Ta002	1454	105.09	100.41	105.09	96.84	99.17	–
Ta003	1334	109.45	103.9	109.45	95.95	82.31	–
Ta004	1535	103.45	99.09	103.45	94.33	95.7	–
Ta005	1457	99.45	96.29	99.45	92.04	88.61	–
Ta006	1463	101.23	97.74	101.23	93.16	95.08	–
Ta007	1469	100.95	99.46	100.95	94.01	88.43	–
Ta008	1413	104.88	101.42	104.88	97.59	91.44	–
Ta009	1479	99.32	94.52	99.32	92.83	88.3	–
Ta010	1386	99.35	95.53	99.35	92.57	88.96	–
Ta011	1960	104.29	99.74	102.6	86.63	87.4	112.60
Ta012	1947	111.25	109.04	111.25	94.14	88.24	–
Ta013	1757	110.42	108.82	110.42	95.39	88.5	–
Ta014	1540	117.6	115.71	117.6	100.39	98.44	–
Ta015	1673	115.54	115.54	115.54	96.65	94.02	–
Ta016	1675	112.96	109.07	112.96	94.93	86.99	–
Ta017	1775	110.59	109.52	110.59	91.38	91.38	–
Ta018	1827	112.59	109.8	112.59	94.75	95.73	–
Ta019	1856	106.3	102.8	106.3	94.13	87.5	–
Ta020	1871	109.62	106.95	109.62	95.24	92.04	–
Ta021	2625	113.26	110.93	113.26	92.8	88.8	122.90
Ta022	2500	114.08	111.2	103.28	89.36	91.2	–
Ta023	2635	114.35	110.89	114.35	94.08	94.12	–
Ta024	2644	113.5	112.22	113.5	88.8	89.33	–
Ta025	2692	111.55	109.7	111.55	90.45	93.13	–
Ta026	2518	119.06	115.49	118.67	94.64	94.32	–
Ta027	2548	119.78	116.56	119.78	93.8	91.88	–
Ta028	2500	113.56	110.52	113.56	93.12	91.16	–
Ta029	2717	110.75	109.39	110.75	86.97	88.7	–
Ta030	2663	111.87	109.61	111.87	87.23	90.16	–

Table 2. Test results comparisons

Algorithm	**Number of data rows (on which the comparisons were made)**	**Number of better results**
HMM-PFA	30	27
MA	–	–
IG-RIS	–	–
HGA	30	23
IIGA	30	27
DSOMA	30	1
HGSA	30	0
IWO	3	2

The table above summarizes how many test datasets the SCA algorithm could be compared with, and how many of these test datasets the SCA algorithm was better. In the case of the HMM-PFA algorithm, the SCA gave better results for 27 data sets, in the case of HGA for 23 data sets, and in the case of IIGA for 27 data sets, while for the DSOMA algorithm, it was only better for 1 data set, while for IWO this number was 2 out of 3 test runs. Overall, it can be said that the SCA algorithm gave better results than most comparison algorithms in most of the benchmark test datasets, so the algorithm can be effectively used for the Flow Shop Scheduling Problem.

4. Conclusions and future work

The article presented the Flow Shop Scheduling problem, which was solved by the Sine-Cosine Algorithm. The Sine-Cosine Algorithm was originally developed for a continuous optimization task, so it was necessary to discretize the algorithm. The discretized algorithm was also presented in the article. The article then presented test results for the Taillard benchmark data set in the article. The results of SCA were compared with the results of six other optimization algorithms, based on which it can be said that SCA gave better results than most algorithms in the majority of cases. The future research direction is the discretization of other metaheuristics and the comparison of the results with the Taillard benchmark dataset.

Acknowledgement

„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."

ÚNKP Új Nemzeti Kiválósági Program

Ministry of Culture and Innovation

References

[1] Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, 96, pp. 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

[2] Abualigah, L., & Diabat, A. (2021). Advances in sine cosine algorithm: a comprehensive survey. *Artificial Intelligence Review*, 54 (4), pp. 2567–2608. https://doi.org/10.1007/s10462-020-09909-3

[3] Attia, A. F., El Sehiemy, R. A., & Hasanien, H. M. (2018). Optimal power flow solution in power systems using a novel Sine-Cosine algorithm. *International Journal of Electrical Power & Energy Systems*, 99, pp. 331–343. https://doi.org/10.1016/j.ijepes.2018.01.024

[4] Hekimoğlu, B. (2019). Sine-cosine algorithm-based optimization for automatic voltage regulator system. *Transactions of the Institute of Measurement and Control*, 41 (6), pp. 1761–1771. https://doi.org/10.1177/0142331218811453

[5] Khrissi, L., El Akkad, N., Satori, H., & Satori, K. (2022). Clustering method and sine cosine algorithm for image segmentation. *Evolutionary Intelligence*, pp. 1–14. https://doi.org/10.1007/s12065-020-00544-z

[6] Rizk-Allah, R. M. (2018). Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. *Journal of Computational Design and Engineering*, 5 (2), pp. 249–273. https://doi.org/10.1016/j.jcde.2017.08.002

[7] Belazzoug, M., Touahria, M., Nouioua, F., & Brahimi, M. (2020). An improved sine cosine algorithm to select features for text categorization. *Journal of King Saud University-Computer and Information Sciences*, 32 (4), pp. 454–464. https://doi.org/10.1016/j.jksuci.2019.07.003

[8] Nenavath, H., & Jatoth, R. K. (2018). Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking. *Applied Soft Computing*, 62, pp. 1019–1043. https://doi.org/10.1016/j.asoc.2017.09.039

[9] Abualigah, L., & Dulaimi, A. J. (2021). A novel feature selection method for data mining tasks using hybrid sine cosine algorithm and genetic algorithm. *Cluster Computing*, 24, pp. 2161–2176. https://doi.org/10.1007/s10586-021-03254-y

[10] Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European journal of operational research*, 205 (1), pp. 1–18. https://doi.org/10.1016/j.ejor.2009.09.024

[11] Mraihi, T., Driss, O. B., & El-Haouzi, H. B. (2023). Distributed permutation flow shop scheduling problem with worker flexibility: Review, trends and model proposition. *Expert Systems with Applications*, p. 121947. https://doi.org/10.1016/j.eswa.2023.121947

[12] Shao, W., Shao, Z., & Pi, D. (2021). Effective constructive heuristics for distributed no-wait flexible flow shop scheduling problem. *Computers & Operations Research*, 136, p. 105482. https://doi.org/10.1016/j.cor.2021.105482

[13] Costa, A., Fernandez-Viagas, V., & Framiñan, J. M. (2020). Solving the hybrid flow shop scheduling problem with limited human resource constraint. *Computers & Industrial Engineering*, 146, p. 106545. https://doi.org/10.1016/j.cie.2020.106545

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

[15] Belabid, J., Aqil, S., & Allali, K. (2020). Solving permutation flow shop scheduling problem with sequence-independent setup time. *Journal of Applied Mathematics*, pp. 1–11. https://doi.org/10.1155/2020/7132469

[16] Rossit, D. A., Toncovich, A. A., Rossit, D. G., & Nesmachnow, S. (2021). Solving a flow shop scheduling problem with missing operations in an Industry 4.0 production environment. *Journal of Project Management*, Vol. 6, Issue 1, pp. 33–44.

[17] Brum, A., Ruiz, R., & Ritt, M. (2022). Automatic generation of iterated greedy algorithms for the non-permutation flow shop scheduling problem with total completion time minimization. *Computers & Industrial Engineering*, 163, p. 107843. https://doi.org/10.1016/j.cie.2021.107843

[18] Demiral, M. F. (2020). A parameters analysis of sine cosine algorithm on travelling salesman problem. *El-Cezeri*, 7 (2), pp. 526–535. https://doi.org/10.31202/ecjse.662864

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

[20] 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), p. 2621. https://doi.org/10.3390/app8122621

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