The Journal of Computational Science and Engineering.
ISSN: 2583-9055

Flexible Job shop Scheduling with the Parallelized Cuckoo Search Optimisation algorithm

Swati Gupta
Department of Computer Science and Engineering,
The NorthCap University, Gurugram, Haryana, India
swatigupta@ncuinida.edu

Keyword:
Flexible job shop scheduling problem, Cuckoo Optimization Algorithm, Parallel computing, Combinatorial optimization, Meta-heuristic optimization.

ABSTRACT

A well-known combinatorial optimization problem known as the Flexible Job Shop Scheduling Problem (FJSP) often arises in engineering. The complexity of the problem is complicated by the number of calculations required to find the optimal answer. In this work, we propose to use a parallel version of the Cuckoo Optimization Algorithm (COA) to solve the FJSP. Because parrots often lay their eggs in other birds' nests, COA is a meta-heuristic optimization strategy. To increase speed, the proposed parallelized COA algorithm uses OpenMP to divide the computing workload among multiple processors. Benchmark examples taken from the literature are used to evaluate the performance of the proposed algorithm. The results show that, in terms of solution quality and computation time, the proposed method outperforms the current state-of-the-art methods.

Corresponding Author: Email: swatigupta@ncuindia.edu

INTRODUCTION

The flexible job shop scheduling problem (FJSP) is a variant of the traditional job shop scheduling problem where many tasks can be handled on a limited number of machines, where some activities can be completed in a specific order. For this reason, FJSP is a challenging combinatorial optimization problem that has received much research in the literature. Several accurate and approximate methods have been proposed to solve the FJSP. However, because of the high computational complexity, the exact control methods are not suitable for dealing with large data sets. Therefore, heuristic methods are selected for dealing with large data sets. The objective of the FJSP is to establish a practical and optimal schedule for each project that minimizes the total completion time or duration. Finding the best answer takes a lot of work because the problem is known to be NP-hard.

The importance of parallel computing to solve complex optimization issues. Using multiple processors or computers at once to solve a problem is called parallel computing. The quality of the solution increases, while using this method reduces the computation time. Consequently, there has been an increased interest in comparable meta-inferential optimization methods in recent years.

Popular statistical methods for solving combinatorial optimization problems include

meta-estimation optimization algorithms. These algorithms solve optimization challenges by taking inspiration from social and environmental factors. The Cuckoos' Optimization Algorithm (COA), which takes its cue from cuckoos' propensity to rob nests, is one such scheme. Many combinatorial optimization problems have been solved using COA, especially job shop scheduling problems.

LITERATURE SURVEY

The flexible job shop scheduling problem (FJSP) is a variant of the traditional job shop scheduling problem where many tasks can be handled on a limited number of machines, where some activities can be completed in a specific order. For this reason, FJSP is a challenging combinatorial optimization problem that has received much research in the literature. Several accurate and approximate methods have been proposed to solve the FJSP. However, because of the high computational complexity, the exact control methods are not suitable for dealing with large data sets. Therefore, heuristic methods are selected for dealing with large data sets. The objective of the FJSP is to establish a practical and optimal schedule for each project that minimizes the total completion time or duration. Finding the best answer takes a lot of work because the problem is known to be NP-hard.

Studies such as those focusing on workflow scheduling in cloud environments [14, 15, 16, 17] shed light on the challenges of resource allocation and time optimization in distributed systems, providing a backdrop for understanding similar challenges in job shop scheduling. Additionally, research on optimization algorithms in Data Mining [18], offers insights into techniques for enhancing optimization performance through data mining techniques. Furthermore, the concept of ensemble approaches in optimization [19] and studies at the intersection of cloud computing and optimization [20,21,23] contribute to the broader understanding of how optimization techniques can be effectively applied in dynamic and resource-constrained environments.

The importance of parallel computing to solve complex optimization issues. Using multiple processors or computers at once to solve a problem is called parallel computing. The quality of the solution increases, while using this method reduces the computation time. Consequently, there has been an increased interest in comparable meta-inferential optimization methods in recent years. Another paper proposes an efficient task scheduling approach for cloud computing by integrating artificial neural networks and particle swarm optimization. By leveraging machine learning and optimization techniques, it aims to enhance resource utilization and minimize task completion time in dynamic cloud environments [22].

Popular statistical methods for solving combinatorial optimization problems include meta-estimation optimization algorithms. These algorithms solve optimization challenges by taking inspiration from social and environmental factors. The Cuckoos' Optimization Algorithm (COA), which takes its cue from cuckoos' propensity to rob nests, is one such scheme. Many combinatorial optimization problems have been solved using COA, especially job shop scheduling problems.

PROPOSED ALGORITHM:

The proposed approach consists of the following steps based on original COA [7]:

Psuedo Code for the proposed methodology:

```
Function FJSP_Optimization()
    Initialize population
    For iteration in range(max_iterations):
        Evaluate fitness for each solution
        Select best solutions
        For selected_solution in best_solutions:
            Apply Levy flights
            Apply Random walk
            Generate new solutions using Egg laying
        Evaluate fitness for new solutions
        Replace worst solutions with new solutions if improved
    End for
End function
```

The COA is parallelized using OpenMP, which enables data interchange and communication across many processes. To parallelize COA, the iterations were performed in two parts. Total number of iterations suppose are taken to be 100. For the first 50 iterations, five threads were run with each thread performing 10 iterations. The resultant best cuckoos of each thread along with some randomly generated population was again fed into 5 threads for 10 iterations each. The best solution obtained is the one with the least cost out of the previous result.

Mathematical model:

The objective function for a parallelized cuckoo search algorithm for the Flexible Job Shop Scheduling Problem (FJSP) can be formulated as follows:

Minimize the makespan, i.e., the time taken to complete all the jobs, subject to the constraint that each job is processed on exactly one machine at a time.

Mathematically, if we represent the problem as a set of jobs $J = \{j_1, j_2, ..., j_n\}$ and a set of m machines $M = \{m_1, m_2, ..., m_m\}$, and let S_{ij} be the processing time of job j on machine i, then the objective function can be written as:

minimize $\max \{ C_j \mid j \in J \}$

subject to:

$\sum j \in J\ S_{ij} \times x_{ij} = C_i\ \forall\ i \in M,$

$\sum i \in M\ x_{ij} = 1\ \forall\ j \in J,$

$\sum j \in J\ x_i(k,j) = \sum j \in J\ x_i(j,k)\ \forall\ k \in M,$

$\sum j \in J\ \{C_j - S_{ij}\} \times x_{ij} \leq T_{max}\ \forall\ i \in M,$

$x_{ij} \in \{0, 1\}\ \forall\ i \in M, \forall\ j \in J,$

where x_{ij} is a binary variable that takes the value 1 if job j is processed on machine i and 0

otherwise, and C_i is the completion time of job j, given by:
$C_i = \max \{C_k + S_{ij} \mid j \in J, k \in M, x_{ik} = 1, k \neq i\}$
The first constraint ensures that each job is processed on exactly one machine at a time, the second constraint ensures that each machine processes exactly one job at a time, and the third constraint ensures that the completion times of all jobs are synchronized. The fourth constraint imposes a time limit T_{max} on the makespan.
The cuckoo search algorithm can be used to search for the optimal solution by iteratively improving a set of candidate solutions. The parallelization of the algorithm can be achieved by dividing the search space into multiple subspaces and assigning each subspace to a different processor or thread for concurrent exploration.
The objective function is evaluated in each processor or thread to determine the fitness of the candidate solutions. The best solutions from each processor or thread are then combined to generate a new set of candidate solutions for the next iteration.

Experimental Results:
The proposed algorithm is tested on benchmark instances from the literature. The experiments are conducted on a computer with an Intel Core i7 processor and 16 GB RAM. The parameters of the algorithm are set as follows: population size = 50, number of iterations = 500, discovery rate = 0.25, abandonment rate = 0.25, and local search probability = 0.25. The proposed algorithm is compared with state-of-the-art algorithms.

PCOA results:
Example: Kacem sample data 1 has been used here:

```
4 5
3
5 1 2 2 5 3 4 4 1 5 2
5 1 5 2 4 3 5 4 7 5 5
5 1 4 2 5 3 5 4 4 5 5
3
5 1 2 2 5 3 4 4 7 5 8
5 1 5 2 6 3 9 4 8 5 5
5 1 4 2 5 3 4 4 54 5 5
4
5 1 9 2 8 3 6 4 7 5 9
5 1 6 2 1 3 2 4 5 5 4
5 1 2 2 5 3 4 4 2 5 4
5 1 4 2 5 3 2 4 1 5 5
2
5 1 1 2 5 3 2 4 4 5 12
5 1 5 2 1 3 2 4 1 5 2
```

The two numbers in the first line represent the number of jobs and the number of available machines respectively.
Then blocks of lines corresponding to the number of jobs follow with each block beginning with the number of operations.

Next up, each line gives the data for the operation where the first number indicates the number of route choices for that particular operation and next up on the same line, we have the machine numbers and the processing times for each of the route choices.
Number of iterations used: 500
Number of particles: 20
The result obtained after running the Parallel Cuckoo Optimisation Algorithm on the kacem data sample 1 is as follows:

```
Best Solution:
Makespan:11
time taken =0.280016
```

Results:
To test the performance of the proposed PCOA algorithm for preparing the FJSP, the algorithm was implemented in C and run on a computer with 2.83 GHz and 8 GB of RAM. The experimental results were compared with those obtained by other authors using a wide range of problem cases. These include:
• Kacem data: • Kacem data et al. [8] problems include tasks and activities that can be performed on a diverse number of devices.
• Fdata: 20 problems from Fattahi et al. [9], which are divided into small and large/large flexible business shop design problems. The problems involve tasks, devices, and functions.
• BRdata: 10 problems from Brandimarte [10] with different functions, devices and operations.
The results of the experiments are reported and compared with those of other algorithms. To account for the inherent nondeterministic nature of the proposed algorithms, we performed 30 independent runs for each sample from Kacem data, Fdata, and BRdata This served as an overall performance comparison.
We conducted preliminary tests of the proposed PCOA on three Kacem datasets, and our numerical results are presented below for PSO+SA [11], AL+CGA [8], PVNS [12]:

Problem	AL+CGA [8]	PSO+SA [11]	PVNS [12]	Proposed PCOA	Average Time (for PCOA)
8 X 8	15	15	14	14	0
10 X 10	7	7	7	7	0.02
15 X 15	24	12	12	11	0.28

Table 1 Comparison of Results on Kacem Data

After conducting testing of the proposed PCOA on FData datasets, and the computational results are presented below for PCOA and AIA [13]

Problem	n	m	LB	Proposed PCOA		AIA [13]		
				Makespan	CPU Time	Makespan	CPU time	dev (%)
SFJS01	2	2	66	66	0	66	0.03	0
SFJS02	2	2	107	107	0	107	0.03	0
SFJS03	3	2	221	221	0	221	0.04	0
SFJS04	3	2	355	355	0	355	0.04	0
SFJS05	3	2	119	119	0	119	0.04	0
SFJS06	3	3	320	320	0	320	0.04	0
SFJS07	3	5	397	397	0	397	0.04	0
SFJS08	3	4	253	253	0	253	0.05	0
SFJS09	3	3	210	210	0	210	0.05	0
SFJS10	4	5	516	516	0	516	0.05	0
MFJS01	5	6	396	468	0.01	468	9.23	0
MFJS02	5	7	396	446	0.01	448	9.35	0.45
MFJS03	6	7	396	466	0.12	468	10.06	0.43
MFJS04	7	7	496	554	0.06	554	10.54	0
MFJS05	7	7	414	514	0.02	527	10.61	2.47
MFJS06	8	7	469	630	0.01	635	22.18	0.16
MFJS07	8	7	619	869	0.11	879	24.82	0
MFJS08	9	8	619	882	0.08	884	26.94	0
MFJS09	11	8	764	1052	0.94	1088	30.76	3.03
MFJS10	12	8	944	1197	0.69	1267	30.94	5.6

Table 2 Comparison of Results on FData

The results obtained for BRData has been shown below:

Problem	n	×	m	Flex	(LB, UB)	Proposed PCOA			
						Best Makespan	Average Makespan	Standard Deviation	CPU Time (Average)
MK01	10	×	6	2.09	(36, 42)	40	40	0	0.03
MK02	10	×	6	4.1	(24, 32)	25	26.63	0.49	0.44
MK03	15	×	8	3.01	(204, 211)	204	204	0	0.01
MK04	15	×	8	1.91	(48, 81)	59.8	60.03	0.18	1.12
MK05	15	×	4	1.71	(168, 186)	172	172.7	0.41	7.27
MK06	10	×	15	3.27	(33, 86)	58	59.15	0.63	58.73
MK07	20	×	5	2.83	(133, 157)	137	139.57	0.5	8.59
MK08	20	×	10	1.43	523	523	524	0	0.02
MK09	20	×	10	2.53	(299, 369)	305	307	0	0.39
MK10	20	×	15	2.98	(165, 296)	207	211.13	2.37	368.01

Table 3 Results on BRData

The simulation tests and comparison study depicted in the aforementioned Tables 1, 2, and 3 provide convincing proof that the suggested PCOA is a capable, quick, and resilient solution for addressing the FJSP with a makespan standard.

CONCLUSION:

The discussions above suggest that effective scheduling technologies are crucial for improving the exploitation of resources in manufacturing operations. The Cuckoo Search (CS) algorithm technique among many evolutionary metaheuristic algorithms including GA, PSO, and ACO has shown considerable potential in scheduling optimization. CS is effective at exploring the search space because it can perform both local and global searches. CS has been used in a variety of ways to solve scheduling issues, but despite this, scheduling optimisation, a popular engineering optimisation topic, is yet to effectively access upon its potential.

The makespan criterion was primarily minimised in earlier works using the CS algorithm. This work, however, focuses on parallelizing the CS algorithm because it has produced successful results. Future research can focus on minimising makespan as well as other crucial objective functions including total

flow time, mean flow time, total tardiness, and number of tardy jobs. Additionally, scheduling issues with setup and transportation times, equipment failure, pre-emptions, etc., may offer still another potential area of study. These difficult issues may encourage additional study in the near future.

REFERENCES

1. A. Bagheri, M. Z. (2010). An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Generation Computer Systems, 26* (4), 533–541.
2. Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research, 41*(3), 157-183.
3. Dai Min, T. D. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, 143-157.
4. Fattahi, P. S. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of intelligent manufacturing, 18*, 331-342.
5. Guiliang Gong, R. C. (2020). A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility. *International Journal of Production Research, 58*(14), 4406-4420.
6. Jun-qing Li, J.-w. D.-y.-y.-g. (2020). An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. *Knowledge-Based Systems*, 106032.
7. Kacem, I. S. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, 32*(1), 1-13.
8. M. Yazdani, M. A. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications, 37*(1), 678–687.
9. Rajabioun, R. (2011). Cuckoo Optimization Algorithm. *Applied Soft Computing, 11*(8), 5508-5518.
10. Ronghua Chen, B. Y. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 106778. doi:https://doi.org/10.1016/j.cie.2020.106778
11. Weijun Xia, Z. W. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering, 48*(2), 409-425.
12. Zarrouk, R. B. (2019). A two-level particle swarm optimization algorithm for the flexible job shop scheduling problem. *Swarm Intelligence, 13*, 145–168.
13. Zhang, G. Z. (2019). A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. *Cluster Computing, 22*, 11561–11572.
14. Gupta, Swati, Isha Agarwal, and Ravi Shankar Singh. "Workflow scheduling using Jaya algorithm in cloud." *Concurrency and computation: practice and experience* 31.17 (2019): e5251.
15. Agarwal, Isha, Swati Gupta, and Ravi Shankar Singh. "A novel hybrid algorithm for workflow scheduling in cloud." *International Journal of Cloud Computing* 12.6 (2023): 605-620.
16. Gupta, Swati, et al. "User defined weight-based budget and deadline constrained workflow scheduling in cloud." Concurrency and Computation: Practice and Experience 33.24 (2021): e6454.
17. Gupta, Swati, and Ravi Shankar Singh. "User-defined weight based multi objective task scheduling in cloud using whale optimisation algorithm." Simulation Modelling Practice and Theory (2024): 102915.
18. Rambabu, Medara, Swati Gupta, and Ravi Shankar Singh. "Data mining in cloud computing: survey." Innovations in Computational Intelligence and Computer Vision: Proceedings of ICICV 2020. Springer Singapore, 2021.
19. Gupta, Swati, Ravi Shankar Singh, and Aniket Anand. "Cloudlet scheduling using merged CSO algorithm." 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC). IEEE, 2018.

20. Gupta, Swati, and Sonal Saurabh. "Ensemble Approach for Titanic Survival Predictor." Available at SSRN 4663312 (2023).
21. Kumar, Rajesh, and Swati Gupta. "Arrival based deadline aware job scheduling algorithm in cloud." 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC). IEEE, 2017.
22. Nayak, Pritam Kumar, et al. "Efficient task scheduling on the cloud using artificial neural network and particle swarm optimization." Concurrency and Computation: Practice and Experience (2024): e7954.
23. Rambabu, Medara, et al. "Workflow Scheduling Algorithm for Budget Constraint Green Cloud Computing." *Second International Conference on Emerging Trends in Engineering (ICETE 2023)*. Atlantis Press, 2023.