Comparative Local Search Approaches to Solve Flexible Job Shop Scheduling Problem

Alham Fikri Aji
University of Edinburgh
s1447728@sms.ed.ac.uk

Abstract

Flexible job shop scheduling is one variant of job shop scheduling problem. Job scheduling problem is considered as NP-hard combinatorics problem. Therefore, approach for finding the optimal result for flexible job shop scheduling is considered ineffective. Local search as heuristic approach for finding near-optimal solution in difficult combinatorics problem is one option to approach flexible job shop scheduling problem. Three different local search techniques by using genetic algorithm, tabu search, and artificial bee colony algorithms are discussed. Those techniques are compared with benchmarked flexible job shop scheduling dataset. In terms of result, genetic algorithm and artificial bee colony outperform in most cases compared to tabu search. In term of time, genetic algorithms provide faster computational time compared to the other algorithms.

1. Introduction

Job Shop Scheduling Problem (JSSP) is combinatoric problem to arrange set of jobs, consist of set of operations to be performed in set of machines, with limitation that each machine can perform at most one operation at any given time. Further more, each operation has predefined operating order [1]. The objective of JSSP is minimizing certain cost function from given valid scheduling. One of the function is to minimize the processing time of all jobs [2].

Flexible JSSP (FJSSP) is one variant of JSSP. FJSSP has some flexibility of choosing the machine to perform certain jobs. In FJSSP, a job may be performed in some given set of machine, rather than fixed predefined machine in JSSP. FJSSP can be defined formally as follows [3]:

- Given set of n jobs $J = \{J_1, ..., J_n\}$
- Each job J_i consists of k_i operations $Oi = \{O_{i1}, O_{i2}, O_{ik_i}\}$
- There are limited set of machines available $M = \{M_1, ..., M_m\}$
- Each operation must be processed in certain machine. We can define $M(O_{ij}) \in M$ as function that returns set of machines used to process $i-$th job's $j-$th operation.

Each machine m has fixed t_{mo} unit of time to complete operation o and it cannot be interrupted.

- Each machine only able to perform at most single task at any given time. Let $start(a)$ is the starting time to process operation a. For any two different operation a and b, where same machine is used, then one should be processed before another. Thus $start(a) \geq start(b) + t_{mb}$ OR $start(b) \geq start(a) + t_{ma}$

- Each operation in Oi must be processed in the predefined order, hence we may define relationship between two operations a and b, $pre(a, b)$ is true if an operation a must be processed before an operation b.

- For any valid scheduling, we define the required time to finish all tasks is $L = \max_{v \in O} = start(v) + t_{mv}$, the time to finish the latest operation with machine m. the objective of FJSSP is to minimize the L.

JSSP and its variant are considered as NP-hard problem [4]. Sotskov and Shakhlevich further explained that JSSP with three set of jobs with three operations and three machines is already classified as NP-Hard [5]. FJSSP involves more complex combinatoric problem, thus solving FJSSP with an exact algorithm tends to be ineffective and heuristic approach to aim close-optimal solution is considered as a better option [6].

Local search is one heuristic approach for difficult combinatorics and optimization problem with good empirical result [7]. Therefore, local search can be used to provide near-optimal solution of FJSSP in effective running time. Three different recent local search approach are explored and compared in term of result and computational time to find out the effectiveness towards FJSSP problem.

The rest parts of this papers are organized as follows. The section 2 of this paper explains recent development of local search algorithms to solve FJSSP. The section 3 of this paper concludes the experimental result achieved in algorithms discussed in section 2. The last section provides summation of discussed algorithms and dissuasion of further works that might be explored.

2. Local Search for Solving FJSSP Problem

Local search explores the solution domain to effectively find near-optimal solution, by moving from solution to another neighbouring solution. Let X as a set of valid solutions based on predefined constraint. In local search, a random starting solution $x_1 \in X$ is chosen. Local search iteratively updates the solution until predefined termination condition occurs. For some iteration n, the solution is updated with neighbouring solution x_{n+1} from current solution x_n. The criterion of selecting the next solution may vary, but the common one is to pick the best solution among the neighbouring solutions [8].

Many local search techniques have been used to solve FJSSP, such as stimulated annealing [9], particle swarm optimization [10] [11], or ant colony optimization[12]. Up until now, local search to solve FJSSP are still being developed by using more efficient and effective

method and implementation. This section discusses three recent and updated implementation to solve FJSSP with different local search approach: genetic algorithm, tabu search, and artificial bee colony.

2.1. Genetic Algorithm Approach

Genetic algorithm is genetics inspired local search approach that treated the solution as a chromosome. At first, the initial population of chromosomes created. For each iteration, the solutions in population were updated by exchanging information through crossover operation or mutation operation. Selection operator is used to limit the size of population by eliminating some unwanted chromosomes. Genetic Algorithm has advantages compared to other local search method in terms of broader variation that can be implemented to find best individual, from the generating initial population phase to dynamic generation phase [6].

Zhang et al. proposed enchanted genetic algorithm approach to solve FJSSP. This approach adapt and modify several previous methods from another genetic algorithm approach and optimize them, to form an effective genetic algorithm implementation [13].

2.1.1. Chromosome Design

Zhang et al. modified the chromosome representation proposed by Ho et al. in [14] as the implementation has inefficient space and time complexity[15]. The representation consists of two parts, machine selection (MS) part and operation sequence (OS) part. Both MS and OS part consist of array of integers with size of the total operations available. In MS part, each integer MS_i represents the $MS_i - th$ available machine used to process the $i - th$ operation. In OS part, each value describe the job number, which represent the job processing order.

Jobs	J_1		J_2
Operations	O_{11}	O_{12}	O_{21}
Set of available machine	$\{M_1, M_3\}$	$\{M_2, M_3, M_4\}$	$\{M_1, M_2, M_4\}$

Table 1: Example of FJSSP

Table 1 shows an example of FJSSP problem. In this problem, there are two jobs available, J_1 and J_2. First job has two operations labelled as O_{11} and O_{12}. The second job has one operation O_{21}. There are four machines available, labelled as M_i for each i from 1 to 4.

MS			OS		
2	2	1	1	2	1
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
M_3	M_3	M_1	O_{11}	O_{21}	O_{12}

Figure 1: Example of chromosome representation

Figure 1 illustrates one valid chromosome representation of instance in table 1. In MS part, the representation means that O_{11} and O_{12} use second available machine and O_{21} uses

the first available machine. The OS part describes the operation order based on the job number as in illustrated above.

2.1.2. Initial Population

The proposed algorithm has two different initial population generations, called Global Selection (GS) and Local Selection (LS). In GS, operation ordering is generated randomly. Then for each operation O_{ij}, a machine will be assigned based on the machine current total processing time, added by the machine time to process the operation O_{ij}. Current total processing time for a machine M_k is sum of processing time of all operations that currently assigned to machine M_k.

In LS, the algorithm works similarly as in GS. The only difference is that current total processing time is counted separately for each job. Hence, to assign operation O_{ij} that belongs to job J_i, local current total processing time of job i is used. Local current total processing time of job i for machine M_k is sum of processing time of all operations in job i that currently assigned to machine M_k.

2.1.3. Selection Operator

The algorithm will randomly choose three individuals and pick the best individual to be selected for the next generation. This method claimed to be able to avoid premature convergence and high influence of very good individuals. Therefore, this method can increase the survivability of good individual.

2.1.4. Crossover Operator

The crossover operator is used to combine two parent chromosomes to form a new child chromosome. The crossover operators are differ in MS and OS part of the chromosome. In MS part, they adopt two different crossover operators: two points crossover [16] and uniform crossover [17]. The crossover operator in OS part adopts preserving order-based crossover (POX) [18].

2.2. Tabu Search Approach

Tabu search iteratively looks up the optimal solution by traversing in a finite set of solution S. For each solution $s \in S$, neighbouring solutions $N(s)$ is defined. In tabu search, special tabu list is defined, to prevent returning to any previously visited solution for several iterations to prevent possible cyclic loop or trapped in local maximum[19]. Li et al. in [20] developed tabu search with multiple approaches for generating initial solution and creating solution neighbourhood representation. The method created by Li et al. can be further explained below:

2.2.1. Solution Representation

Two vectors A_1 and A_2 are defined. A_1 consists of pairs that define the operation to machine matching. The vector A_2 which uses the idea composed by Gen et al. in [21] represents the operation processing order according to the order in the vector sequence. With FJSSP configuration in 1, one vectors example that similar with representation in figure 1 should be $A_1 = (O_{11}, M_3), (O_{12}, M_3), (O_{21}, M_1)$ and $A_2 = O_{12}, O_{21}, O_{11}$.

2.2.2. Initial Solution

The initial solution created separately for both vectors. In A_1 vector, four different methods are used. (1) Random Rule is where each pairing are generated randomly. (2) Operation Minimum Processing Time Rule is where each operation is paired with the machine that has fastest processing time. (3) Global and (4) Local Minimum Processing Time Rule which is completely same with GS and LS methods from genetic algorithm approach.

Vector A_2 generated with four different methods. (1) Random Rule is where the operation order generated randomly. (2) Most Work Remaining Rule orders the jobs with non-increasing remaining work. (3) Most number of Operations Remaining Rule chooses job with higher operations remaining. (4) Shortest Processing Time prioritizes the operations with lowest processing time to be selected first.

2.2.3. Neighbouring Solution

New neighbouring solutions generated with several rules based on machine assignment part and operation scheduling part. In machine assignment, three general rules are defined:

- Random rule: Randomly pick an operation with more than two candidate machines and replace the machine with another.
- Top $k-$Most Work rule: Pick random machine from top-k machine with heaviest workload. Select an operation that using such machine and change it with less busy machine.
- Last Processing rule: Select an operation with latest finish time, and change its machine to another machine that is currently not used in critical operation. Critical operation is considered as any operation that belongs to critical path of FJSSP solution in disjunctive graph form.

Suppose a solution has several critical operations. A Critical block is defined as a maximum sequence of critical operations that share a same machine. The neighbouring solutions of operation scheduling component can be found by modifying the structure of the critical block.

Van Laarhoven et al. in [22] developed first successful neighbourhood structure for JSSP by swapping adjacent pair of critical operations in critical block. Different technique proposed by Dell'Amico and Trubian by moving a critical operation to very end or very first of the block. The author adopted both neighbouring structure to create three different operations. Neighbouring solutions created by swapping an internal critical operation in a block with its head or rear, inserting head or rear operation into the internal location, or inserting an internal operation to very first or very end operation.

For each iteration, the non-tabu solution with smallest total completion time from the generated neighbouring solutions is chosen to replace the current solution. Tabu list is updated by inserting the best neighbour so far and removing the oldest solution.

2.3. Artificial Bee Colony

Artificial bee colony is a new local search approach that mimics honeybee swarm's behaviour in finding a food source [24]. A solution is represented as a food source and the amount of nectar represents how good the solution is. A bee colony has of total N bees that consist of N_s scout bees, N_e employed bees and N_o onlooker bees. Initially, there are N_e solution generated randomly where one solution correspond to one employed bee. For each iteration, each employed bee improved their current solution by predefined method. Then, each onlooker bee will pick one solution from set of current solutions with a certain probability, where better solution has higher probability. Each solution is further improved by onlooker bee depending on how many onlooker bees choose the solution. Finally, each scout bee finds a new solution, thus resulting N_s new solution [25].

Wang et al. proposed novel artificial bee colony approach to solve FJSSP [26]. It claimed to be first artificial bee colony approach to for solving FJSSP. Detailed method of this approach shown below:

2.3.1. Solution Representation

Solution represented as a single vector of pair of machine and operation. The pair order of the vector describes the operation ordering of the solution. This representation considered to be more efficient compared to the representation of tabu search as this representation only requires single vector instead of two. In FJSSP instance in table 1, vector $V = (O_{11}, M_3), (O_{21}, M_1), (O_{12}, M_3)$ is one valid example of the solution.

2.3.2. Solution Initialization

Solution initialization is used in two different parts of artificial bee colony. The first one is for initial population generation and the second one is for scout bee's solution finding phase. There will be N_e solutions generated initially and for N_s solutions generated for each iteration. In each iteration, if there is exist newly generated solution that is better than the employed bees current solution, the new solution will replace the employed bee's solution.

Solutions generated separately between machine assignment solution and operation scheduling solution. Machines are assigned with three rules. (1) Random rule is where assignments generated in complete random. The last two rules are (2) Local and (3) Global Minimum Processing Time Rule, which interestingly same initialization rule as in tabu search approach or genetic algorithm approach. For the operation sequence solution, the author uses three different approaches : (1) Random Rule, (2) Most Time Remaining Rule, and (3) Most number of Operations Remaining Rule. Rule (1) and (3) are the same rules as tabu search approach discussed earlier. Rule (2) sorts the sequence of the jobs in non-increasing order of remaining time[27].

2.3.3. Neighboring Solution

Employed bee globally explores the solution to find new neighbouring solution. The author developed method that similar to how genetic algorithm's way to find neighbouring solutions with crossover and mutation operation. Each employed bee updates the current solution s_i into new solution s_i' by exchanging the information with other employed bee.

Information exchange done by using crossover method. Crossover done by using two points crossover, uniform crossover, and modified preserving order-based crossover (MPOX) [10]. Each solution randomly mutated by changing machine assigned to certain operation with another machine. Onlooker bee further updates the current solution explored by employed bee. This process done by modifying the critical operations that occur in the solution, similar in tabu search approach.

3. Experimental Result

Problem	Jobs (n)	Machine (m)	Total Operations (T_O)
Mk1	10	6	55
Mk2	10	6	58
Mk3	15	8	150
Mk4	15	8	90
Mk5	15	4	106
Mk6	10	15	150
Mk7	20	5	100
Mk8	20	10	225
Mk9	20	10	240
Mk10	20	15	240

Table 2: BRData information [27]

BRData is FJSSP dataset created by Brandimarte[27]. This dataset consists of ten test cases of FJSSP instances with varying number of machines, jobs and operations. Table 2 provides the detailed information of the dataset. All discussed algorithms in this paper used this data set for performance testing, thus their result can be compared directly.

Approach	CPU Power	Halting Criterion	Total runs
Genetic Algorithm	1.8 GHz	100 generations	5
Tabu Search	1.6 GHz	$5 * n * m$ iterations OR T_O iterations without improvement	50
Artificial Bee Colony	2.83-GHz	$n * m$ iterations OR $2 * n$ iterations without improvement	50

Table 3: Experimental setup

Table 3 shows the experimental setup of all discussed algorithms. Zhang et al. genetic algorithm approach is the only approach that uses static halting criterion. Tabu search approach uses limit up to T_O iterations without improvement on besides total iteration halting condition. Artificial Bee Colony proposed by Wang et al. also uses alternate halt condition of $2 * n$ iteration without improvement.

	Genetic Algorithm			Tabu Search			Bee Colony		
Problem	Avg	Best	Time	Avg	Best	Time	Avg	Best	Time
Mk1	40	**40**	1.6	40.30	**40**	2.80	40.00	**40**	3.23
Mk2	26	**26**	2.6	26.50	**26**	19.31	26.50	**26**	35.63
Mk3	204	**204**	1.3	204.00	**204**	0.98	204.00	**204**	1.19
Mk4	60	**60**	6.2	64.88	62	40.82	61.22	**60**	38.94
Mk5	173	173	7.3	172.90	**172**	20.23	172.98	**172**	19.29
Mk6	58	**58**	15.7	67.38	65	27.18	64.48	60	66.61
Mk7	145	144	17.3	142.21	140	35.29	141.42	**139**	131.84
Mk8	523	**523**	2.2	523.00	**523**	4.65	523.00	**523**	2.33
Mk9	307	**307**	30.2	311.29	310	70.38	308.76	**307**	91.21
Mk10	199	**198**	30.6	219.15	214	89.83	212.84	208	237.11

Table 4: Experimental results of proposed methods [26][20][13]

Table 4 shows the results obtained of each discussed algorithm tested with BRData by using configuration in table 3. For each discussed algorithm, best solution, average solution, and running time in second are displayed. Bold number indicates best solution among the algorithms for corresponding data test.

4. Conclusion and Discussion

Local search is able to find sub-optimal solution of FJSSP. Several benchmarked test cases have been used to test and compare the performance of the discussed local search approaches. In general, each algorithm provides similar result. However, different result occurs in some cases, therefore, some approaches still found difficulty to avoid the local maximum.

From all three discussed algorithms, genetic algorithm and artificial bee colony approaches tend to be better from tabu search in term of solution achieved. None of the solutions provided by tabu search is better than both genetic algorithm's and artificial bee colony's solution. genetic algorithm provides better solution in some cases compared to artificial bee colony while artificial bee colony has better solution in some other cases compared to genetic algorithm. In terms of computational time, artificial bee colony tends to be slower than the other two even with better CPU power usage. In the other hand, genetic algorithm provides faster computational time compared with the other two. With good result and faster computational speed, genetic algorithm is more preferable.

The way genetic algorithm explores from solutions to the other solutions is done by exchanging informations from two different parent solutions, by some defined crossover techniques. Therefore, the computational complexity depends on the size of the chromosome. However, in tabu search and artificial bee colony a new solution gained from an effort to improve a current solution. In this study, the improvement done by manipulating operations in critical section. Hence, understanding what is going on with current solution is

required. Another computational operation is needed to transform the solution into a disjunctive graph form as well to detect the critical section. Therefore, this approach requires higher computational complexity. Artificial bee colony approach tends to be even slower as it has three different sub-algorithms implemented in three different bee phases for each iteration.

Zhang et al. stated in their paper that in some cases, their genetic algorithm approach has already reached convergence before 100 generations, and in two test cases convergence obtained from the first generation. Therefore further work by implementing adaptive halting condition might improve the computational speed of genetic algorithm furthermore by reducing the iteration dynamically.

As newly developed local search method, artificial bee colony has shown competitive result with older techniques such as genetic algorithm or tabu search. The artificial bee colony enables hybrid approach that uses different technique implementation for each bee phase. The artificial bee colony approach discussed in this paper interestingly implements genetic algorithm in its employed bee phase, and implements local search based on critical operation similar to tabu search in its onlooker bee phase. Further works can be done by trying different local search approach for each bee phase to find out better combination.

The interesting part about JSSP and its variants is that there is no much information about the optimal solution available, due to its NP-hard problem property. Therefore the way that used to compare the goodness of one method is to compare the result relatively to results from another method. However, we cannot measure how close our solution against the optimal solution. Hence, we do not know whether our developed algorithm is good already and does not require any further improvement, or it is still very far from the actual solution and need more improvement.

One method that might be explored further is by studying results gained from very first approach up until the newest approach. The results can be explored from time to time to see the improvement trend. The trend might provide information of any convergence of the result, that is no noticeable improvement made for several past years. The convergence might be one pointer that indicates the developed approaches have already close to the optimal solution.

References

[1] Jacek Błażewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European journal of operational research*, 93(1):1–33, 1996.

[2] Marshall L Fisher. Optimal solution of scheduling problems using lagrange multipliers: Part i. *Operations Research*, 21(5):1114–1127, 1973.

[3] Anant Singh Jain and Sheik Meeran. Deterministic job-shop scheduling: Past, present and future. *European journal of operational research*, 113(2):390–434, 1999.

[4] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[5] Yu N Sotskov and Natalia V Shakhlevich. Np-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59(3):237–266, 1995.

[6] F Pezzella, G Morganti, and G Ciaschetti. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10):3202–3212, 2008.

[7] Emile HL Aarts and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[8] Marc Pirlot. General local search methods. *European journal of operational research*, 92(3):493–511, 1996.

[9] NM Najid, S Dauzere-Peres, and A Zaidat. A modified simulated annealing method for flexible job shop scheduling problem. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 5, pages 6–pp. IEEE, 2002.

[10] Guohui Zhang, Xinyu Shao, Peigen Li, and Liang Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318, 2009.

[11] Liang Gao, CY Peng, C Zhou, and PG Li. Solving flexible job shop scheduling problem using general particle swarm optimization. In *Proceedings of the 36th CIE conference on computers & industrial engineering*, pages 3018–3027, 2006.

[12] Li-Ning Xing, Ying-Wu Chen, Peng Wang, Qing-Song Zhao, and Jian Xiong. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing*, 10(3):888–896, 2010.

[13] Guohui Zhang, Liang Gao, and Yang Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38(4):3563–3573, 2011.

[14] Nhu Binh Ho, Joc Cing Tay, and Edmund M-K Lai. An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, 179(2):316–333, 2007.

[15] Hongbo Liu, Ajith Abraham, and Crina Grosan. A novel variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems. In *Digital Information Management, 2007. ICDIM'07. 2nd International Conference on*, volume 1, pages 138–145. IEEE, 2007.

[16] Masato Watanabe, Kenichi Ida, and Mitsuo Gen. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Computers & Industrial Engineering*, 48(4):743–752, 2005.

[17] Jie Gao, Linyan Sun, and Mitsuo Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9):2892–2907, 2008.

[18] Kyung-Mi Lee, Takeshi Yamakawa, and Keon-Myung Lee. A genetic algorithm for general machine scheduling problems. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on*, volume 2, pages 60–66. IEEE, 1998.

[19] Parviz Fattahi, Mohammad Saidi Mehrabad, and Fariborz Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3):331–342, 2007.

[20] Jun-Qing Li, Quan-Ke Pan, PN Suganthan, and TJ Chua. A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *The international journal of advanced manufacturing technology*, 52(5-8):683–697, 2011.

[21] Mitsuo Gen, Yasuhiro Tsujimura, and Erika Kubota. Solving job-shop scheduling problems by genetic algorithm. In *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*, volume 2, pages 1577–1582. IEEE, 1994.

[22] Peter JM Van Laarhoven, Emile HL Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992.

[23] Mauro Dell'Amico and Marco Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3):231–252, 1993.

[24] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009.

[25] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[26] Ling Wang, Gang Zhou, Ye Xu, Shengyao Wang, and Min Liu. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 60(1-4):303–315, 2012.

[27] Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.