Multi-Agent Model based on Tabu Search for the Permutation Flow Shop Scheduling Problem

Olfa Belkahla Driss, Hafewa Bargaoui
Stratégies d'Optimisation et Informatique intelligentE (SOIE)
High Institute of Management, University of Tunis
41, Street of Liberty Bouchoucha-City CP-2000-Bardo Tunis, Tunisia
Higher Business School of Tunis, University of Manouba

KEYWORD

Scheduling
Permutation flow shop
Tabu Search
Multi-agent system

ABSTRACT

The objective of this work is to present a distributed approach for the problem of finding a minimum makespan in the permutation flow shop scheduling problem. The problem is strongly NP-hard and its resolution optimally within a reasonable time is impossible. For these reasons we opted for a Multi-agent architecture based on cooperative behaviour allied with the Tabu Search meta-heuristic. The proposed model is composed of two classes of agents: Supervisor agent, responsible for generating the initial solution and containing the Tabu Search core, and Scheduler agents which are responsible for the satisfaction of the constraints under their jurisdiction and the evaluation of all the neighborhood solutions generated by the Supervisor agent. The proposed approach has been tested on different benchmarks data sets and results demonstrate that it reaches high-quality solutions.

1 Introduction

The scheduling theory is a branch of operations research and it represents a very profetic research area. By definition, a scheduling problem consists in allocation number of jobs to machines taking into consideration a set of constraints. One of the most popular cases of machine scheduling we cite the permutation flow shop which has been the subject of a significant amount of literature in the combinatorial optimization and scheduling theory. It concerns a set N = {1,…, n} of n independent jobs (tasks) has to be processed on a set of M = {1,…,m} machines in the same order. The processing time for job i on machine j is denoted $t_{i,j}$. In the permutation flow shop problem job passing in the sequence is forbidden; that is, the processing sequence on the first machine is maintained throughout the remaining machines. The objective is to find a permutation that minimizes one or more criterion. The studied problem is expressed as Fm|prmu|Cmax following the notation introduced by [Graham and al, 1979] where the objective is to find a permutation that minimizes the makespan (the completion time of the last job on the last machine). This problem is known to be NP-hard in the strong sense for m ≥3 [Garey and Johnson, 1979].

The article is structured as follow: the next section, presents an overview of the literature about the permutation flow shop scheduling problem with makespan minimization. Section 3, describes in details the background of Tabu Search. Section 4 highlights the proposed multi agent model named Multi-Agent model for Flow Shop Problem (M.A.F.S.P). Section 5, contains the adaptation of the different elements of the Tabu Search. In section 6, we present the global dynamic of M.A.F.S.P. Section 7 discusses experimental results on the performance of the proposed approach in treating the permutation flow shop problem minimizing the makespan. Finally, section 8

gives our conclusions and remarks with some future directions.

2 Relate work

Since the pioneering algorithm of [Johnson, 1954] to solve the two machines problem optimally, many exact methods have been developed in order to obtain the optimal solution. Noteworthy [Tseng and al, 2004] presented an empirical analysis of four competing mixed-integer linear programming (MILP) models of to solve the regular permutation flow shop problem with makespan criterion. [Ladhari and Haouari, 2005] proposed a high performing approach based on a parallelization of the well known branch and bound algorithm deployed in a grid of computer. However, due to the difficulty and the huge computation time, of exactly resolving the problem has led researchers to focus their energies in the development of heuristic procedures which provide high quality solution in a short time even though the size of the problem increases. The heuristic methods can be classified either as constructive or improvement approaches. Constructive heuristic builds a feasible solution starting from scratch, step by step. Among them the CDS heuristic [Campbell and al, 1970] as an extension of the Johnson algorithm for m>2 and the algorithm developed by [Dannenbring, 1977] known as RA which can be presented as variant of CDS procedure. Also, [Nawaz and al, 1983] presented the NEH heuristic which is considered as the best method among simple constructive heuristics for flow shop scheduling. On the other hand, the improvement heuristic starts with a feasible solution which they aim to improve by exploring its neighborhood. Methods of this type include [Dong and al, 2008] which is a variant of NEH heuristic. Meta-heuristics for the PFSP appeared much later to generate good results. Some of the most recent are the ant colony optimization presented by [Ying and Liao, 2004], the iterated greedy methods by [Ruiz and Stützle, 2007], the Tabu Search proposed by [Ekşioğlu and al, 2008] and the recent bee colony developed by [Liu and Liu, 2013].

3 The Tabu Search approach

The Tabu Search (TS) originally proposed by [Glover, 1986] is an iterative meta-heuristic approach based on the principle of the local search and that proved to be successful in solving hard optimization problem. Tabu Search works by starting from an initial solution as current solution and then moves successively among neighboring by applying a move mechanism in the hope of finding an improved solution until some stopping criterion has been satisfied. At each iteration, the process consists in selecting the most appropriate neighborhood which may not be an improving solution. Then the search is repeated starting from the best found permutation as a new current solution. In order to avoid being trapped in suboptimal regions, the Tabu Search uses a memory structure which operates by storing the selected move in a data structure called Tabu List for certain number of iterations named Tabu List size. This mechanism excludes moves which would bring the search back where it was at some previous iteration.
In this simplest form, Tabu Search requires the following ingredients:

- Initial solution
- Neighborhood definition
- Move strategy
- Tabu List
- An integrated diversification scheme
- Stopping rules

The following section describes in details the proposed Multi-Agent model.

4 The Multi-Agent model

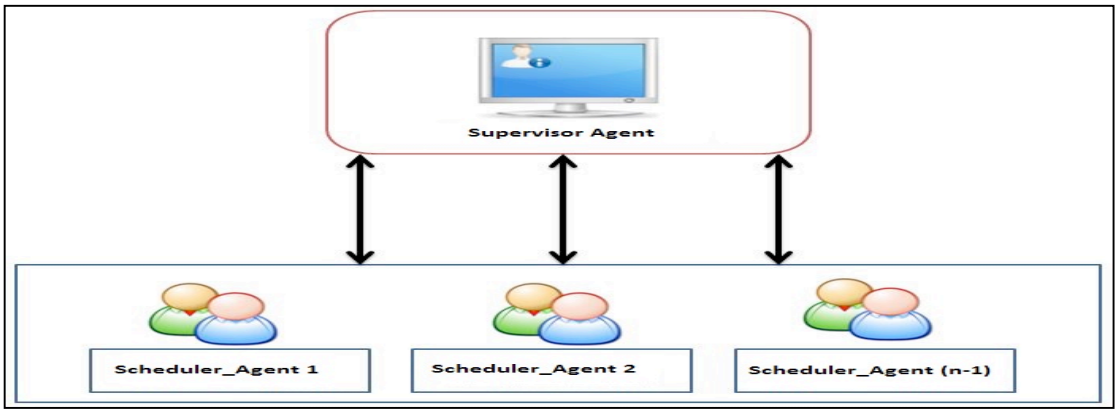

Fig. 1. M.A.F.S.P architecture

Multi-Agent system is one of the most interesting programming paradigms to facilitate the development of distributed and decentralized architectures. In a Multi-Agent system, each agent can communicate, collaborate, coordinate and negociate with other agents in order to achieve the objectives for which it was designed. Our model called Multi-Agent model for Flow Shop Problem (M.A.F.S.P) as illustrated in Fig.1 consists of two types of reactive agents (i.e agent based on Stimulus/Reaction): One Supervisor agent and (n-1) Scheduler agents agents with n refers the number of jobs. We use (n-1) Scheduler agents because for each current solution of the TS algorithm the Supervisor agent generates (n-1) neighboring solutions. Each agent has acquaintances (the agents that it knows and with which it can communicate) and its own behavior. The global optimization process of our model is based on Tabu Search approach in order to minimize the makespan. In the remainder of this section, we explain in detail the different types of agents.
Before starting the distributed solving process the Supervisor agent generate an initial solution and then try to ameliorate it iteratively by applying the global optimization process based on TS approach in order to minimize the makespan. The different types of agents are explained in detail below.

4.1 Supervisor agent

Supervisor agent contains the core of Tabu Search; it's unsatisfied as long as the maximal number of iteration is not reached. It aims to launch the program and create as many Scheduler agents as neighboring solutions (i.e. number of jobs-1). Furthermore, it provides all the necessary information needed for each Scheduler agent namely the execution time of each job on all machines and the current solution. It also aims to detect that the problem has been resolved and output the corresponding scheduling solution and its makespan value.
As the core of the global optimization, i.e. the Tabu Search, of M.A.F.S.P is located at the Supervisor agent, the knowledge of the latter are composed mainly of the Tabu Search parameters. Static knowledge of the Supervisor agent is composed of:

- The execution time of each job on all machines
- The initial solution S0 from which begins the optimization process.
- The used stopping criterion: Max_Iterations, i.e. the maximum number of iterations allowed by the Tabu Search algorithm.
- The size of the Tabu List
- The used diversification criterion: threshold _diversification, i.e. the number of iterations between two successive improvements.

Its dynamic knowledge consists of:

- The current solution and its makespan
- Neighbor solutions and their makespans,
- The best found schedule and its makespan
- The tabu list elements
- The performed number of iterations
- The number of iterations after the last improvement

The Supervisor agent is satisfied when the stopping criterion is reached; in this case it provides the found-solution to the user.

4.2 Scheduler agent

The Scheduler agent plays an important role in our model. Indeed, it has been shown that the time needed to evaluate the value of the makespans of the neighborhood of the current solution is almost the entire calculation time [Taillard, 1990]. For this reason, we assign for each Scheduler agent the task of evaluating and calculating the cost of one neighbor; according to this, we estimate, reduce the total time required to search for the best neighbor.
The Scheduler agent behaviors are divided into two types:

- Local plan which describes the behavior of an isolated agent, i.e. calculating the cost of the scheduling and the satisfaction of constraints under their responsibility, and it does not require interaction with other agents,
- Communication protocol that describes the behavior of the Scheduler agent interacting with the Supervisor agent and requires sending and receiving messages.

In M.A.F.S.P, the evaluation of the whole neighborhood solutions is achieved by cooperation among the Scheduler agents. In fact, each Scheduler agent receives a message from the Supervisor agent that contains the current solution, the job ji to be moved as well as its new position. The Scheduler agent, responsible for scheduling calculations, put the job j_i in its new position and calculates the cost of the new solution. Then, it sends a message to the Supervisor agent containing the new scheduling and its cost. The latter, after receiving all the neighboring solutions calculated by the Scheduler agents, chooses the best non tabu neighbor from the neighborhood of the current solution to start the next iteration.
The knowledge of the Scheduler agent are mainly composed of the current solution provided by the Supervisor agent, the execution time of each job on all machines, the job to move and its new position.

5 Elements of Tabu Search approach

In what follows, we will give the adaptation of the different parameters of the Tabu Search for the permutation flow shop scheduling.

4.2 Initial solution

It has been shown that the effectiveness of the approach based on the principle of local search depends heavily on the quality of the initial solution [Jain and al, 2000]. Based on the review of flow shop heuristics by [Ruiz and Maroto, 2005], the method due to Nawaz, Enscore and Ham (NEH) [Nawaz and al, 1983] is regarded as one of the best constructive heuristic for the permutation flow shop problem. The NEH heuristic can be described by the following four steps:
1. Sort the n jobs by decreasing total processing time on the m machines
2. Take the first two jobs and schedule them in order to minimize the partial makespan as if there were only these two jobs
3. For the k-th job, k = 3 to n do
4. Insert the k-th job into the place, among k possible ones, which minimizes the partial makespan
The complexity of NEH heuristic is $O(n^3m)$ which can lead to considerable computation times for large instances. However, it is possible to reduce the complexity of NEH to $O(n^2m)$ by

using the implementation of [Taillard, 1990] called NEHT, by means of three matrices, named e (heads), q (tails) and f (heads plus processing times).
In M.A.F.S.P, we use [Taillard, 1990] algorithm since it is the best polynomial heuristic in practice and it is rather implemented straightforwardly. At the initial stage, the Supervisor agent generates an initial scheduling then searches through its neighborhood, a permutation with the lowest makespan by applying TS mechanism.

5.2. Neighborhood definition

The function adopted to generate the neighborhood of a current solution is the key factor for the success of an approach based on Tabu Search. A move changes the location of some jobs in the basic sequence. In the literature, we can meet many types of moves. Among the most used ones we can cite:

Swapping move: given a permutation π of n jobs; j and k ∈{1, …, n}, two consecutive positions in π, the permutation that results of such a move is obtained by swapping the two jobs in position j and k. The size of the neighborhood is (n-1). Experiments show that such a neighborhood yields local minima and requires high calculation time.

Exchange move: given a permutation π of n jobs; j and k ∈{1, …, n}, two positions in π, the permutation that results of such a move is obtained by interchanging the jobs in positions j and k. The size of the neighborhood is n(n - 1)/2, these moves find good quality of schedules but the neighborhood exploration is in O (n^3 m) which leads to a high computational time .

Insertion move: given a permutation π of n jobs; j and k ∈{1, …, n}, two positions in π, the permutation that results of such move is obtained by inserting the job in position i in position j. The size of this neighborhood is (n-1)2 but it can be evaluated in O(n^2m), using the insertion algorithm described in the NEHT heuristics.
For all neighborhood types the positions j and k can be specified randomly or by applying some predefined criterion.

Experiments show that the insertion move is regarded as the best neighborhood structure for the permutation flow shop scheduling problem so we choose this type of neighborhood because of its efficiency in terms of the quality of solution and running time.
In order to remove a job from a sequence to reinsert it elsewhere efficiently we use the extension of Taillard's implementation presented in [Deroussi and al, 2010] which can be described as follows:
1. Compute the earliest completion time $e_{i,j}$ of the i-th job on the j-th machine; the starting time of the first job on the first machine is 0.

$e_{0,j} = 0$; $e_{i,0} = 0$;
$e_{i,j} = \max(e_{i,j-1}, e_{i-1,j}) + t_{i,j}$;
$i = 1,\dots, k$, $j = 1,\dots,m$

2. Compute the tail $q_{i,j}$, i.e. the duration between the starting time of the i-th job on the j-th machine and the end of the operations.

$q_{k+1,j} = 0$; $q_{i,m+1} = 0$;
$q_{i,j} =, \max(q_{i,j+1}, q_{i+1,j})+t_{i,j}$
$i = k, \dots, 1$; $j = m, \dots, 1$

3. The value of the partial makespan M'_i when removing the job at the i-th position is:

$M'_i = \max_j(e_{i-1,j}+q_{i+1,j})$
$i = 1,\dots, k$; $j = 1, \dots,m$

While the best insertion position in the case of job insertion is the position p that minimizes the makespan, in case of job remove [Deroussi and al, 2010] proposed a relative criterion which consists in maximizing the gain of makespan relatively to the sum of processing times of the removed job. The relative criterion is described as follows: remove the job at the p-th position, such that $M_p = \max_{i=1..k}\left(\frac{M - M'_i}{\sum_{j=1..m} t_{i,j}}\right)$ where M is the makespan of the permutation before removing the job. In M.A.F.S.P once the initial permutation is determined, it is considered as a current solution. The Supervisor agent selects the less well-inserted job and sends a message to the (n-1) Scheduler agents each of which contains the current solution, the job to move and its new position (remove the job at position p1 and insert it at position p_2 such as
($p_2 \notin p_1$), see Fig. 2. Then all the Scheduler agents have the responsibility to evaluate the cost of each neighbor in a parallel way.

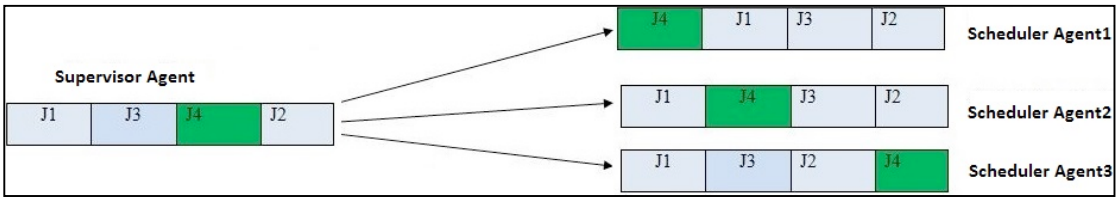

Fig. 2. The proposed neighborhood structure

5.3 Move strategy

Move strategy determines the manner in which the neighbors are examined. There are three well known ways to choose a move leading the next step:

1) Examine all possible neighbors and choose the best not tabu move
2) Examine a fixed number of neighbors that are not tabu
3) Examine the neighbors and make a move as soon as a better solution is found

In M.A.F.S.P, first, each Scheduler agent calculate the makespan of each neighborhood solution then the Supervisor agent selects the non-tabu permutation that has the smallest makespan among all possible neighbors as a new current solution for the next iteration.
Once the calculation of makespan of all possible neighbor solutions is done, they will be sent to the Supervisor agent. The latter chooses the non-tabu permutation with the smallest cost to start a new iteration and inserts its makespan in the Tabu List.
When a new best solution cannot be found after a predetermined threshold, the search switches to diversification phase which will be explained in Section 5.5.

5.4. Tabu List

The Tabu List is one of the mechanisms that attempted to avoid local optimality. A variety of elements can be stored in this list such as: makespans that was already obtained, a job and its position or a pair of jobs with their positions. In M.A.F.S.P, we propose to apply a Tabu List defined as a finite list T with a fixed size experimentally determined which stores makespans of permutations already visited. The FIFO (First In First Out) rule is adopted; once the list is full, the oldest element is removed and a new one is added. In our implementation we used a Tabu List of size 7.

5.5. Diversification scheme

Despite the effectiveness of the Tabu Search method in solving permutation flow shop scheduling, some limitations have been detected. Indeed, the main inconvenience is summed up in the absence of an effective diversification technique that encourages the search process to examine unvisited regions, as the best solutions at the local level are not necessarily good solutions globally. In order to provide a wide exploration of the search space and to get out of a local optimum, NEHT heuristic is hybridized into TS to find a new solution. In M.A.F.S.P, when no improvement in makespan is obtained for a certain value named Counter_Diversification, it means that best-so-far solution is not replaced by neighborhood solutions and that Tabu Search is entrapped in local optimum. Once the variable counter_diversification exceeds a predetermined number of iterations, called Threshold_Diversification, the algorithm jumps back to the best permutation obtained so far and selects randomly two integers L_1 and L_2 ($0< L_1$, $L_2 \leq n$ and $L1 < L_2$) representing the start and end point of the sub-sequence of jobs to reinsert by applying the insertion phase of the NEHT heuristic until a complete permutation of n jobs is obtained as described in Fig. 3.

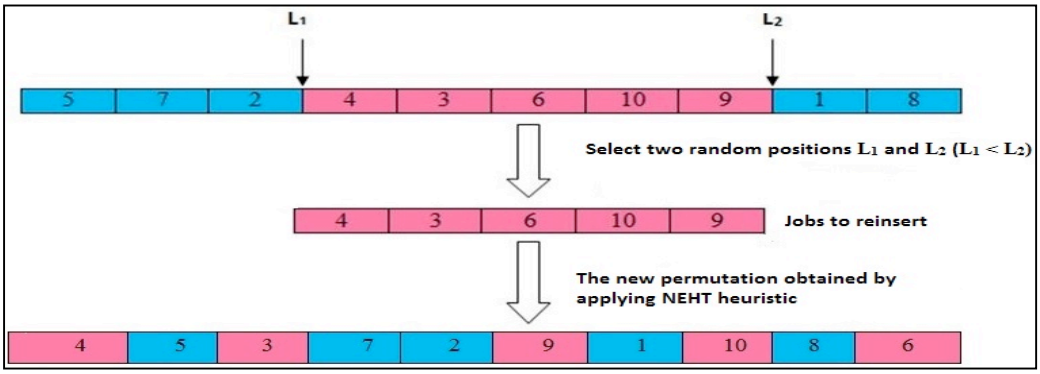

Fig. 3. Example for the application of diversification scheme

In M.A.F.S.P, the Supervisor agent applies the diversification phase on the current solution. The new obtained scheduling serves as a new current solution of Tabu Search in order to get rid of the local optimum.

5.6. Stopping rules

In M.A.F.S.P, the stopping rule that ends the search consists of setting a limit on the number of iterations called max_iteration which will be experimentally determined.
In M.A.F.S.P, when the maximum number of iterations is reached the Supervisor stops the algorithm and displays the results.

6 Global dynamic

Before starting the distributed solving process, the Supervisor agent knowing the problem to solve, applys the NEHT heuristic to generate an initial solution and then try to ameliorate it by applying iteratively the Tabu Search approach.
Once the initial permutation is determined, it is considered as a current solution. The Supervisor agent selects the less well-inserted job and sends a message to the (n-1) Scheduler agents each of which contains the current solution, the job to move and its new position(remove the job at position p_1 and insert it at position p_2 such as $p_2 \notin p_1$). Then all the Scheduler agents have the responsibility to evaluate the cost of each neighbor in a parallel way. Once the calculation of makespan of all possible neighbor solutions is done, they will be sent to the Supervisor agent. The latter chooses the non-tabu permutation with the smallest cost to start a new iteration and inserts its makespan in the Tabu List.
When a new best solution cannot be found after a predetermined threshold "Threshold_Diversification", the search switches to diversification phase. In M.A.F.S.P, the Supervisor agent applies the diversification phase on the current solution. The new obtained scheduling serves as a new current solution of Tabu Search in order to get rid of the local optimum.
The above process continues until the stopping rule is satisfied, at this stage the Supervisor agent kills all the Scheduler agents, and displays to the user the best found permutation and its makespan.

7 Experimental evaluation

The M.A.F.S.P was implemented in the JADE platform and all tests were conducted on a Core2Duo 2.20 GHz with 3.0 GB. To test the effectiveness and the performance of the proposed M.A.F.S.F for the permutation flow shop scheduling, we provide a comprehensive experimental evaluations and comparisons of the proposed algorithm with 3 other powerful methods: NEHT heuristic [Nawaz and al, 1983], 3XTS Tabu Search [Ekşioğlu and al, 2008] and HDABC bee colony [Liu and Liu, 2013].

Two tests were carried out. For the first test we used Taillard's instances [Taillard, 1993] which combine 20, 50, 100 and 200 jobs with 5, 10 and 20 machines. For the second test, computational simulations are carried out 21 different instances rec01, rec03 to rec41 designed by Reeves and Yamada [Reeves and Yamada, 1998].

The results of comparison between NEHT, 3XTS and M.A.F.S.P are given in table1. Results are only for 23 instances of Taillard's benchmarks because 3XTS comparison is based on these 23 instances. The asterisk (*) indicates that optimal solution was found.

Problem	size (N × M)	NEHT	3XTS	M.A.F.S.P
Ta001	20*05	1286	1278	**1278***
Ta002		1365	1359	**1359***
Ta003		1159	1081	**1081***
Ta004		1325	1293	**1293***
Ta005		1305	1235	**1235***
Ta011	20*10	1680	1582	**1582***
Ta012		1729	1659	**1659***
Ta013		1557	1496	**1496***
Ta014		1439	1377	**1377***
Ta015		1502	1419	**1419***
Ta021	20*20	2410	2297	**2297***
Ta022		2150	2103	**2099***
Ta023		2411	2330	2328
Ta024		2262	2229	**2223***
Ta025		2397	2291	**2291***
Ta031	50*5	2733	2724	**2724***
Ta041	50*10	3135	3025	**3025**
Ta051	50*20	4082	3893	3895
Ta061	100*5	5519	5493	**5493***
Ta071	100*10	5846	5770	**5770***
Ta081	100*20	6541	6300	**6283**
Ta091	200*10	10 942	10869	10872
Ta101	200*20	11 594	11251	11 331

Table. 1. Results for NEHT, 3XTS and M.A.F.S.P

Analyzing the performance of our algorithm, we have observed that results obtained by M.A.F.S.P are better than those obtained by 3XTS approach in 17% of instances while 3XTS approach outperforms our approach in 13% of instances. We also note that M.A.F.S.P and 3XTS provide the same results in 70% of instances. Table1 shows also that the proposed approach can easily reach the optimal solution for 17 instances out of 23.

In order to show the effective of M.A.F.S.P, we carry out a simulation to compare it with another recent algorithm called HDABC [Liu and Liu, 2013]. The experimental results are listed in Table2.

nstances	Problem size (N × M)	C*	HDABC	M.A.F.S.P
Rec01	20*05	1247	1247	**1247***
Rec03		1109	1109	**1109***
Rec05		1242	1242	**1242***
Rec07	20*10	1566	1566	**1566***
Rec09		1537	1537	**1537***
Rec11		1431	1431	**1431***
Rec13	20*15	1930	1932	**1930***
Rec15		1950	1963	**1950***
Rec17		1902	1917	**1902***
Rec19	30*10	2093	2101	**2099**
Rec21		2017	2046	**2019**
Rec23		2011	2020	**2018**
Rec25	30*15	2513	2542	**2522**
Rec27		2373	2392	**2379**
Rec29		2287	2310	**2289**
Rec31	50*10	3045	3101	**3053**
Rec33		3114	3126	**3114***
Rec35		3277	3277	**3277***
Rec37	75*20	4951	5104	**5014**
Rec39		5087	5180	**5134**
Rec41		4960	5106	**5031**

Table. 2. Results for HDABC and M.A.F.S.P

From Table2, it can be concluded that the performance of M.A.F.S.P is better than HDABC. Tests demonstrate that M.A.F.S.P surpassed HDABC in 67% of instances in terms of makespan. Also, we can see that that the proposed approach provided the optimal solution for 11 instances out of 21.

8 Conclusions and future research

In this paper we have proposed a multi-agent model based on Tabu Search meta-heuristic, called Multi_Agent model for Flow Shop Problem (M.A.F.S.P), to solve the permutation flow shop scheduling problem with makespan criterion. M.A.F.S.P consists of Supervisor agent and Scheduler agents in interaction, trying to find the best possible permutation.
Our model was tested against another Tabu Search and a bee colony algorithm. The results are very promising and show that the M.A.F.S.P is competitive with other successful methods.

Future topics along this line of research may include:

- Extending the idea to combining M.A.F.S.P with other meta-heuristics such as simulated annealing or genetic algorithm to improve the solution and explore the search space.
- M.A.F.S.P could be easily modified to application in different problems for flow shop, for example: no-wait flow shop.
- Investigate the applicability of the proposed approach to the multi-objective permutation flow shop scheduling problem.

9 References

[Graham and al, 1979] Graham, R.L., E. Lawler, J.K. Lenstra, A. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Annals of Discrete Mathematics 5, pp. 236-87 (1979)

[Garey and Johnson, 1979]. Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: Freeman; (1979)

[Johnson, 1954] Johnson, S.M. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, pp. 61-68. (1954)

[Tseng and al, 2004] Tseng F.T., Stafford Jr. E.F., Gupta J.N.D. An empirical analysis of integer programming formulations for the permutation flowshop. OMEGA, The International Journal of Management Science 32 (4):285–93, 2004.

[Ladhari and Haouari, 2005] A computational study of the permutation flow shop problem based on a tight lower bound. Computers & Operations Research 32, pp. 1831–47 (2005)

[Campbell and al, 1970] Campbell H.G., Dudek R.A., Smith M.L. A heuristic algorithm for the n job, m machine sequencing problem. Management Science 16 (10), pp. B630-B637 (1970)

[Dannenbring, 1977] Dannenbring D.G. An evaluation of flow shop sequencing heuristics. Management Science 23(1), pp. 1174-82 (1977)

[Nawaz and al, 1983] Nawaz M, Enscore Jr. EE, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA: The International Journal of Management Science 11 (1): 91–95 (1983)

[Dong and al, 2008] X. Dong, H. Huang, P. Chen. An improved NEH-based heuristic for the permutation flow shop problem. Computers & Operations Research. 35(12), pp. 3962-68 (2008)

[Ying and Liao, 2004] Ying K.C., Liao C.J. An ant colony system for permutation flow-shop sequencing. Computers & Operations Research 31:791–801, 2004

[Ruiz and Stützle, 2007] Ruiz R., Stützle T. A simple and effective iterated greedy algorithm for the

permutation flow shop scheduling problem. European Journal of Operational Research 177, pp. 2033-49 (2007)

[Ekşioğlu and al, 2008] Ekşioğlu B, Ekşioğlu S.D, Jain P. A tabu search algorithm for the flow shop scheduling problem with changing neighborhoods. Computers & Industrial Engineering, Vol 54(1) pp. 1-11(2008)

[Liu and Liu, 2013] Y.F. Liu, S.Y. Liu. A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. Applied Soft Computing 13.13(3), pp. 1459-1463(2013)

[Glover, 1986] Glover F. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13, pp. 533-49 (1986)

[Taillard, 1990] E. Taillard : Some efficient heuristic methods for the flowshop sequencing problem. European Journal of Operational Research, Vol. 47, pp. 65-74. (1990)

[Jain and al, 2000] A. Jain, B. Rangaswamy et S. Meeran : Job shop neighborhoods and move evaluation strategies, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, 2000.

[Ruiz and Maroto, 2005]

[Deroussi and al, 2010] Deroussi L, Gourgand M, Norre S. Une adaptation efficace des mouvements de Lin et Kernighan pour le flow-shop de permutation. 8éme Conférence Internationale de MOdélisation et SIMulation: MOSIM, Hammamet, Tunisie (2010)

[Taillard, 1993 Taillard E. Benchmarks for basic scheduling problems. European Journal of Operational Research 64, pp. 278-85, (1993)

[Reeves and Yamada, 1998]. Reeves C, Yamada T. Genetic algorithms, path relinking and the flowshop sequencing problem. Evolutionary Computation 6, pp. 45–60, (1998)