STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.08

A TABU SEARCH APPROACH FOR PERMUTATION FLOW SHOP SCHEDULING

CRISTINA ELENA DODU AND MIRCEA ANCĂU

Abstract. The adaptive distance between the neighbourhood's makespans influences the local search to explore the non-investigated areas of the solutions space. A Tabu Search with the intensive concentric exploration over non-explored areas is proposed as an alternative solution to the simplest Tabu Search with the random shifting of two jobs indexes operation for Permutation Flow Shop Problem (PFSP) with the makespan minimization criteria.

1. INTRODUCTION

The Permutation Flow Shop Problem(PFSP) is a production problem where a set of n jobs have to be processed on the same order on m machines. Every job has a running time or processing time on each machine, no machine processes more than one job at a time, the preemption of jobs is not allowed and it is considered that machines never breakdown during the scheduling process. The goal is to find the right sequence among the possible $n!$ sequence to minimize the production time - the time at which the last job is completed on machine m, called the makespan. If there are 2 machines, then the problem can be solved in $O(nlog_n)$ time by Johnson's algorithm[15]-the most classical algorithm in the scheduling area. Minimum of three active machines condition in the PFSP environment's setup causes a migration of the problem-solving approach (Garey[15]) to the "NP-complete problem" standards because the optimization algorithms with polynomial time have not yet been found. It had a positive spillover effect, a plenitude of heuristic and meta-heuristics has concentrated on offering a viable global solution. The heuristic's life-cycle is defined by the self-governing steps, independent of each

Received by the editors: 24 February 2020.

2010 *Mathematics Subject Classification.* 68T20, 68P10.

1998 *CR Categories and Descriptors.* I.2.8 [**Computing methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search*; D.2.8 [**Software engineering**]: Metrics – *Complexity measures*.

Key words and phrases. Metaheuristics, PFSP, Permutation Flow Shop Problem, Scheduling Flow shop with concentric exporation, Local search, Tabu Search.

other: index's establishing, solution's building and solution's improvement. Palmer[5], Campbell, Dudek, Smith[11], Gupta[12], Dannenbring[4] obtained widely known and esteemed results for the constructive algorithms and Nawaz, Enscore and Ham[14] for the insertion heuristic algorithms known as NEH algorithm. Heuristics do not have the knowledge of self-sustaining to alter the direction of the search approach when a local optimum is seeable and meta-heuristics are designed to fix this impediment. Meta-heuristics may accept a temporary deterioration of the solution which allows them to explore more thoroughly the solution space and thus to get a hopefully better solution (that sometimes will coincide with the global optimum). From the meta-heuristics typologies, the single-point approach transforms the current solution by analyzing its neighbourhood.

In section 2 were introduced the formal definition of PFSP and the Tabu Search methodology. Section 3 explaines in detail **TSA** algorithm -the proposed methodology and the results obtained on the Taillard's benchmark sets[7] are presented in the next section. In section 5 is analyzed the **TSA**' performance on solving PPSP and finally, section 6 takes up the conclusions.

2. METHODOLOGY REVIEW

The problem is formally defined in the following: n, jobs $j_1, j_2, ..., j_n$ have to be processed on a series of m machines: $m_1, m_2, ..., m_m$ and the processing order of the jobs on the machines is the same for every machine. For each job j on each machine i the processing time that is defined before the beginning of the process. A complete list of these assumptions is detailed by Baker[2]:

- All jobs are independent and available for processing at time 0.
- All machines are continuously available.
- Each machine can process at most one job at a time and each job can be processed only on one machine at a time.
- The processing of a given job at a machine cannot be interrupted once started, i.e, no preemption is allowed.
- Setup times are sequence independent and are included in the processing times or are otherwise ignored.
- All jobs are independent and available for processing at time 0.
- An infinite in-process storage buffer is assumed. If a given job needs an unavailable machine then it joins a queue of unlimited size waiting for that machine. The objective is to find the sequence of n jobs, which achieves the minimal makespan when all jobs are processed on the m machines. The total number of feasible solutions to this problem is derived from the possible job's permutations on machines.

Let be a jobs processing permutation $J = (j_1, j_2, ..., j_n)$, where j_k denotes the job which is in position k of J. Let $p(i, j)$ be the processing time ($j = 1, ..n$ and $i = 1..m$). The completion time $c(j, i)$ is calculated by the formula:

$$c(j, i) = \max(c(j - 1, i), c(j, i - 1) + p(i, j)) \tag{1}$$

where $c(0, i) = 0$ and $c(j, 0) = 0$

The makespan or the completion time is the difference between the time of completion of the last job and the starting time of the first job:

$$Cmax(J) = c(m, n) \tag{2}$$

Let β denote the set of all job permutations, J. The makespan minimization criterion or $Cmax$ means finding J^*, the optimal sequence of jobs that will minimize the completion time:

$$Cmax(J^*) = \min(J, J \in \beta) \tag{3}$$

For more than two machines, PFSP is one of the well-known NP-hard combinatorial optimization problems. The mainly used metaheuristics are based on an improvement of an initial solution by research in its neighbourhood by one of the disruption procedures of the current solution.

Tabu Search(TS)as a single-point meta-heuristic emissary localizes the best candidate from the neighbourhood(NH) of a proposed solution. TS avoids to re-visit the old solutions. memorizing in the Tabu List (TL) as is described by the Tabu Search methodology (Glover[9]]). The neighbourhood's population is feeded from random exchange on the current solution's attributes.

Taillard[7] spotlighted the starting solution of TS approach selecting the improved heuristic proposed by Nawaz, Enscore and Ham[14] - known as NEH and shifted a single position to obtain a new neighbor. Nowicki and Smutnicki[6] concluded that the interrelated units of jobs legitimatizes specific insertions more proficients than other combinations. Reeves et al.[3] have been monitored the vicinity of the global optimum solution because not far distance are concentrated the local optimum solutions and spotted the effect as "big valley phenomenon" (when going along trajectory between two local optimal solutions, it is possible to find a new optimum local or even an optimum global). Ochoa and Veerapen[10] decompose "big valley phenomenon" into sub-valleys or funnels and identified a possible issue in discovering high-quality solutions if the global optimum is not positioned in the largest valley. Drezner[18] proposed the concentric tabu search for the Quadratic Assignment Problem and suggested different rules for the scanning around the center solution.

3. PROPOSED METHODOLOGY

First, Tabu Search approach (**TSA**) analyzes the nearest solutions and after that it searches in other non-explored areas that are registered into the buffer zone. **TSA** starts with the solution provided by the algorithm NEH[14] and randomly interchanges two jobs indexes in ordet to alter the current solution (SL) and the result becomes a neighbor if its makespan is filtered by the restrictions of the adaptive distance ($ADIST$).

The adaptive distance is increased only when the empty list of candidates in the neighbourhood forces the algorithm to uses the best candidate from the buffered list. The adaptive distance together with the selection in the buffer list conduct the exploration far distance from the last current solution. The following restrictions are applied on each candidate:

$$Cmax(SL) \neq Cmax(X) \tag{4}$$

$$Cmax(X) \leq Cmax(SL) + ADIST \tag{5}$$

TSA proposed a buffer zone that acts as a waiting list for the non-explored regions because here are collected all the "invisible" candidates to the neighbourhood. Considering that all-knowing entities prepared into an individual iteration are "unseeable" then the neighbourhood is empty and the buffer zone's repository provides the next current solution. Once the search for non-explored areas is started from the best solution from the buffer zone, the buffer zone is ready to collect other non-explored solutions from scratch and $ADIST$ is increased with a small value.

3.1. Algorithm TSA.

$k \leftarrow 1, s \leftarrow NEH, SL \leftarrow s, SLO \leftarrow s,$
$F \leftarrow Cmax(SL), ADIST \leftarrow 0, maxIterNum \leftarrow 5000, maxNum \leftarrow 100$
while $k \leq maxIterNum$ **do**
  $add(TL, SL)$
  $tBuffer \leftarrow empty$
  $num \leftarrow 0$
  repeat
    $G \leftarrow generateRandom$
    if $G \in TL$ **then**

```
      repeat
        G ← generateRandom
        num ← num + 1
      until [G ∉ TL]or[num = maxNum]
    end if
    if [Cmax(G) ≠ Cmax(SL)]and[Cmax(G) ≤ Cmax(SL) + ADIST]
    then
      add(NH(SL), G)
    else
      add(tBuffer, G)
    end if
  until num ≤ maxNum
  ordersByCmax(NH(SL))
  ordersByCmax(tBuffer)
  if NH(SL) = empty then
    BT ← extractFirst(tBuffer)
    if BT ∈ TL then
      repeat
        remove(tBuffer, BT)
        BT ← extractFirst(tBuffer)
      until BT ∈ TL
      SL ← BT
      ADIST ← ADIST + 1
    end if
  else
    BS ← extractFirst(NH(SL))
    SL ← BS
  end if
  if Cmax(SL) < F then
    SL ← SLO
    ADIST ← 0
  end if
  k ← k + 1
end while
```

Notations:

- s – the initial solution is the solution obtained by Nawaz, Enscore and Ham [14] for the insertion heuristic algorithm known as NEH
- k – the iteration number, $maxIterNum$ - the number of iterations
- $maxNum$ - the number of the neighbors
- SL - the current solution in the kth iteration

- $NH(SL)$ - the neighbourhood of SL
- BS - the best-evaluated entity from $NH(SL)$ obtained based on the following formula
- SLO - the optimal solution
- $F - Cmax(SLO)$
- G - neighbor and $G \in NH(SL)$. The distance between each neighbor and the current solutions restricted by the $ADIST$
- TL – the tabu list
- $tBuffer$ - the buffer zone
- TL – the tabu list
- BT- the best-evaluated entity from $tBuffer$:

$$Cmax(BT) = \min\{Cmax(T), T \in tBuffer\} \tag{6}$$

- $generateRandom$ generates randomly an entity by interchanging two jobs positions
- $add(list, entity)$ adds entity to the list
- $ordersByCmax(list)$ orders the list descending by Cmax value calculated for each element
- $extractFirst(list)$ returns first element from the list
- $remove(list, entity)$ removes entity element from the list

With the adaptive distance, the speed of the search process is improved due to increasing the local-minimum-found probability. If a total number of search rounds to locate a local minimum is M (M is a positive integer) without the adaptive distance, the time consumed is $M \times Time(x)$ where $Time(x)$ is the time consumed to visit any single solution x in the search space. Once the **TSA**'s restrictions are applied (forumulas 4 and 5), the total search round is reduced by:

$$\alpha \times M \times Time(x) < M \times Time(x) where 0 < \alpha \leq 1 \tag{7}$$

Let be X' the best solution of $NH(SL)$, $Cmax(SL) < Cmax(X')$, then

$$0 < \alpha \leq ADIST = 1, \alpha = Cmax(X') - Cmax(SL) \tag{8}$$

4. RESULTS

Taillard proposed 12 sets of the processing times, each set with 10 instances of n jobs and m machine. The first set is the small one, the number of jobs is 20 and the number of machines is 20. For the next sets Taillard increased gradually the number of jobs along with the number of the machines until 500 jobs and 200 machines. Each **TSA**'s solution represented by $Cmax$ is compared with the Upper Bound denoted by $Mean$ provided by Taillard for each set, using for the gap the following formula:

$$GAP = \frac{Cmax - UB}{UB} * 100\% \tag{9}$$

and for all 12 instances from a benchmark set it is calculated the average of the gaps. Beside the gap, for each problem are calculated: the average ($Mean$), Standard Deviation(SD), Standard Score(S) - how many Standard Deviation from the Mean of Cmax and the Confidence Interval of the Mean with a 95% percent Level of Confidence:

$$Mean = \frac{1}{IterNum} \sum_{i=1}^{IterNum} Cmax_i \tag{10}$$

where $IterNum$ is the iterations number.

$$SD = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (Cmax_i - Mean)^2} \tag{11}$$

$$S = \frac{Cmax - Mean}{SD} \tag{12}$$

TSA runs using 5000 iterations and obtains results very closely or identical with the known upper bounds for Taillard's[8] data sets. After the successive running of **TSA**, the maximum number of the iterations was limited to 5000, $IterNum = 5000$, and the size of the neighbourhood to 100 for all Taillard's[8] benchmarks set over 120 instances.

TABLE 1. Results of **TSA** running over each problem from Taillard benchmark sets

Results of **TSA** running over each problem from Taillard's **20 jobs and 5 machines** benchmark set:									
$Problem$	UB	TSA	AD	$Mean$	SD	S	$[CI$	$CI]$	CI_{width}
1	1278	1278	0.00	1,279.44	1.64	1	1,279.40	1,279.49	0.09
2	1359	1365	0.44	1,366.16	1.49	1	1,366.12	1,366.20	0.08
3	1081	1081	0.00	1,096.69	7.15	3	1,096.49	1,096.89	0.4
4	1293	1293	0.00	1,308.69	5.45	3	1,308.54	1,308.85	0.30
5	1236	1235	**-0.08**	1,249.64	4.43	4	1,249.52	1,249.77	0.25
6	1195	1210	1.26	1,211.30	1.92	1	1,211.25	1,211.36	0.11
7	1239	1251	0.97	1,251.89	1.14	1	1,251.85	1,251.92	0.06
8	1206	1206	**0.00**	1,212.64	4.64	2	1,212.51	1,212.77	0.26
9	1230	1230	**0.00**	1,235.53	8.27	1	1,235.30	1,235.75	0.46
10	1108	1108	**0.00**	1,112.84	5.07	1	1,112.70	1,112.98	0.28
$Average$			**0.26**						0.23
Results of **TSA** running over each problem from Taillard's **20 jobs and 10 machines** benchmark set:									

Continued on next page

Table 1 continued from previous page									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	1582	1583	0.06	1,611.46	11.15	3	1,611.15	1,611.77	0.62
2	1659	1664	0.30	1,698.61	9.28	4	1,698.35	1,698.86	0.51
3	1496	1500	0.27	1,527.46	8.60	4	1,527.22	1,527.70	0.48
4	1378	1377	***-0.07***	1,400.43	7.96	3	1,400.20	1,400.65	0.44
5	1419	1419	**0.00**	1,447.01	10.27	3	1,446.72	1,447.29	0.57
6	1397	1401	0.29	1,426.70	6.51	4	1,426.52	1,426.88	0.36
7	1484	1484	**0.00**	1,503.46	9.22	3	1,503.21	1,503.72	0.51
8	1538	1538	**0.00**	1,576.34	11.43	4	1,576.03	1,576.66	0.63
9	1593	1593	**0.00**	1,611.42	6.94	3	1,611.23	1,611.61	0.39
10	1591	1598	0.44	1,625.43	9.21	3	1,625.17	1,625.68	0.51
Average			**0.13**						0.50
Results of **TSA** running over each problem from Taillard's **20 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	2297	2298	0.04	2,338.04	12.11	4	2,337.70	2,338.37	0.67
2	2100	2111	0.52	2,139.21	9.01	4	2,138.96	2,139.46	0.50
3	2326	2328	0.09	2,370.37	13.07	4	2,370.01	2,370.73	0.72
4	2223	2233	0.45	2,261.82	9.92	3	2,261.55	2,262.10	0.55
5	2291	2298	0.31	2,338.72	11.84	4	2,338.39	2,339.05	0.66
6	2226	2229	0.13	2,263.12	12.86	3	2,262.76	2,263.48	0.71
7	2273	2281	0.35	2,319.05	12.69	3	2,318.70	2,319.40	0.70
8	2200	2207	0.32	2,241.08	11.07	4	2,240.78	2,241.39	0.61
9	2237	2242	0.22	2,275.21	12.58	3	2,274.86	2,275.55	0.70
10	2178	2179	0.05	2,219.79	12.85	4	2,219.44	2,220.15	0.71
Average			**0.25**						0.65
Results of **TSA** running over each problem from Taillard's **50 jobs and 5 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	2724	2724	**0.00**	2,724.60	0.69	1	2,724.58	2,724.61	0.04
2	2834	2838	0.14	2,838.67	0.78	1	2,838.65	2,838.69	0.04
3	2621	2621	**0.00**	2,621.73	0.90	1	2,621.70	2,621.75	0.05
4	2751	2762	0.40	2,766.94	6.88	1	2,766.75	2,767.13	0.38
5	2863	2864	0.03	2,864.61	0.71	1	2,864.59	2,864.63	0.04
6	2829	2835	0.21	2,835.81	0.98	1	2,835.78	2,835.83	0.05
7	2725	2725	**0.00**	2,732.18	3.21	3	2,732.09	2,732.27	0.18
8	2683	2683	**0.00**	2,684.37	2.69	1	2,684.29	2,684.44	0.15
9	2552	2561	0.35	2,561.76	0.90	1	2,561.73	2,561.78	0.05
10	2782	2782	**0.00**	2,783.83	1.51	2	2,783.78	2,783.87	0.08
Average			**0.11**						0.11
Results of **TSA** running over each problem from Taillard's **50 jobs and 10 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	3025	3075	1.65	3,099.26	13.10	2	3,098.90	3,099.62	0.73
2	2892	2911	0.66	2,947.72	15.70	3	2,947.28	2,948.15	0.87
3	2864	2905	1.43	2,928.51	11.26	3	2,928.19	2,928.82	0.62
4	3064	3071	0.23	3,081.95	10.45	2	3,081.66	3,082.24	0.58
5	2986	3024	1.27	3,040.77	17.38	1	3,040.29	3,041.26	0.96
6	3006	3026	0.67	3,055.78	24.54	2	3,055.10	3,056.46	1.36
7	3107	3165	1.87	3,171.32	10.70	1	3,171.02	3,171.62	0.59
8	3039	3060	0.69	3,068.82	5.52	2	3,068.67	3,068.98	0.31
9	2902	2932	1.03	2,940.12	9.11	1	2,939.87	2,940.38	0.51
10	3091	3120	0.94	3,147.51	15.73	2	3,147.07	3,147.94	0.87
Average			**1.04**						0.74
Results of **TSA** running over each problem from Taillard's **50 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	3875	3926	1.32	3,977.79	20.68	3	3,977.22	3,978.36	1.15
2	3715	3786	1.91	3,837.90	16.19	4	3,837.46	3,838.35	0.90
3	3668	3730	1.69	3,783.79	19.55	3	3,783.25	3,784.34	1.08
4	3752	3796	1.17	3,859.72	24.04	3	3,859.06	3,860.39	1.33
5	3635	3731	2.64	3,776.80	21.24	3	3,776.21	3,777.39	1.18
6	3698	3761	1.70	3,808.23	27.11	2	3,807.48	3,808.98	1.50
7	3716	3776	1.61	3,823.12	19.28	3	3,822.59	3,823.66	1.07
Continued on next page									

Table 1 continued from previous page									
8	3709	3794	2.29	3,850.41	22.45	3	3,849.79	3,851.04	1.24
9	3765	3815	1.33	3,873.25	18.54	4	3,872.73	3,873.76	1.03
10	3777	3827	1.32	3,875.01	22.00	3	3,874.40	3,875.62	1.22
Average			**1.70**						1.17
Results of **TSA** running over each problem from Taillard's **100 jobs and 5 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	5493	5495	0.04	5,496.31	4.10	1	5,496.19	5,496.42	0.23
2	5268	5284	0.30	5,284.76	2.27	1	5,284.70	5,284.82	0.13
3	5175	5179	0.08	5,181.07	6.05	1	5,180.90	5,181.24	0.34
4	5014	5023	0.18	5,023.49	0.55	1	5,023.48	5,023.51	0.03
5	5250	5255	0.10	5,255.90	1.54	1	5,255.86	5,255.95	0.09
6	5135	5139	0.08	5,139.47	0.56	1	5,139.46	5,139.49	0.03
7	5246	5251	0.10	5,252.73	1.03	2	5,252.70	5,252.76	0.06
8	5106	5114	0.16	5,114.47	0.62	1	5,114.45	5,114.48	0.03
9	5454	5454	0.00	5,474.94	11.42	2	5,474.62	5,475.26	0.63
10	5328	5339	0.21	5,342.30	0.89	4	5,342.28	5,342.33	0.05
Average			**0.12**						0.16
Results of **TSA** running over each problem from Taillard's **100 jobs and 10 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	5770	5790	0.35	5,797.52	9.75	1	5,797.25	5,797.79	0.54
2	5349	5365	0.30	5,384.17	16.15	2	5,383.72	5,384.61	0.90
3	5677	5719	0.74	5,730.49	17.14	1	5,730.01	5,730.96	0.95
4	5791	5812	0.36	5,836.99	17.94	2	5,836.49	5,837.48	0.99
5	5468	5510	0.77	5,525.32	15.86	1	5,524.88	5,525.76	0.88
6	5303	5312	0.17	5,322.65	10.41	2	5,322.36	5,322.94	0.58
7	5599	5675	1.36	5,679.87	6.87	1	5,679.68	5,680.06	0.38
8	5623	5695	1.28	5,697.68	5.15	1	5,697.54	5,697.82	0.29
9	5875	5940	1.11	5,950.02	11.10	1	5,949.71	5,950.33	0.62
10	5845	5903	0.99	5,903.61	0.82	1	5,903.59	5,903.63	0.05
Average			**0.74**						0.62
Results of **TSA** running over each problem from Taillard's **100 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	6286	6367	1.29	6,434.09	24.63	3	6,433.41	6,434.78	1.37
2	6241	6351	1.76	6,396.25	29.75	2	6,395.43	6,397.08	1.65
3	6329	6461	2.09	6,500.94	23.87	2	6,500.28	6,501.60	1.32
4	6306	6408	1.62	6,431.68	22.09	2	6,431.07	6,432.30	1.22
5	6377	6475	1.54	6,532.58	31.03	2	6,531.72	6,533.44	1.72
6	6437	6512	1.17	6,563.04	29.20	2	6,562.23	6,563.85	1.62
7	6346	6422	1.20	6,490.71	32.86	3	6,489.80	6,491.62	1.82
8	6481	6552	1.10	6,620.51	35.37	2	6,619.53	6,621.49	1.96
9	6358	6440	1.29	6,496.68	40.01	2	6,495.57	6,497.79	2.22
10	6465	6599	2.07	6,609.70	11.09	1	6,609.39	6,610.01	0.61
Average			**1.51**						1.55
Results of **TSA** running over each problem from Taillard's **200 jobs and 10 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	10868	10892	0.22	10,930.37	19.98	2	10,929.82	10,930.93	1.11
2	10494	10555	0.58	10,577.32	20.13	2	10,576.76	10,577.88	1.12
3	10922	11017	0.87	11,019.35	3.35	1	11,019.26	11,019.45	0.19
4	10889	11010	1.11	11,013.59	11.71	1	11,013.26	11,013.91	0.65
5	10524	10575	0.48	10,579.12	9.48	1	10,578.85	10,579.38	0.53
6	10331	10378	0.45	10,384.23	12.98	1	10,383.87	10,384.59	0.72
7	10857	10936	0.73	10,941.75	15.44	1	10,941.32	10,942.18	0.86
8	10731	10828	0.90	10,828.57	0.77	1	10,828.55	10,828.59	0.04
9	10438	10478	0.38	10,485.67	12.07	1	10,485.33	10,486.00	0.67
10	10676	10728	0.49	10,746.67	16.70	2	10,746.21	10,747.14	0.93
Average			**0.62**						0.68
Results of **TSA** running over each problem from Taillard's **200 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	11294	11406	0.99	11,457.89	54.00	1	11,456.39	11,459.39	2.99
2	11420	11472	0.46	11,530.50	32.46	2	11,529.60	11,531.40	1.80
Continued on next page									

Table 1 continued from previous page									
3	11446	11555	0.95	11,635.79	63.65	2	11,634.03	11,637.56	3.53
4	11347	11566	1.93	11,632.65	35.95	2	11,631.65	11,633.64	1.99
5	11311	11455	1.27	11,497.29	43.05	1	11,496.10	11,498.48	2.39
6	11282	11488	1.83	11,496.73	17.52	1	11,496.24	11,497.21	0.97
7	11456	11591	1.18	11,650.84	39.73	2	11,649.74	11,651.94	2.20
8	11415	11599	1.61	11,636.32	46.84	1	11,635.02	11,637.62	2.60
9	11343	11457	1.01	11,518.17	49.46	2	11,516.80	11,519.54	2.74
10	11422	11590	1.47	11,676.66	61.09	2	11,674.97	11,678.35	3.39
Average			**1.27**						2.46
Results of **TSA** running over each problem from Taillard's **500 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	*[CI*	*CI]*	CI_{width}
1	26189	26429	0.92	26,480.42	35.83	2	26,479.42	26,481.41	1.99
2	26629	26907	1.04	27,023.21	68.29	2	27,021.31	27,025.10	3.79
3	26458	26721	0.99	26,745.47	32.12	1	26,744.58	26,746.36	1.78
4	26549	26799	0.94	26,843.07	63.34	1	26,841.31	26,844.82	3.51
5	26404	26588	0.70	26,619.27	31.30	1	26,618.41	26,620.14	1.74
6	26581	26784	0.76	26,833.32	51.03	1	26,831.91	26,834.74	2.83
7	26461	26600	0.53	26,649.42	53.46	1	26,647.94	26,650.91	2.96
8	26615	26926	1.17	26,961.00	42.15	1	26,959.83	26,962.17	2.34
9	26083	26430	1.33	26,459.02	24.88	2	26,458.33	26,459.71	1.38
10	26527	26741	0.81	26,802.75	42.96	2	26,801.56	26,803.95	2.38
Average			**0.92**						2.47

5. DISCUSSION

TSA, coded in Java, ran on a PC INTEL™Core-i5 CPU @ 2.30 GHz processor 16 GB. **TSA**'s CPU times vary from 5 seconds (on the 20 benchmark set) to 20 minutes (on the 500 benchmark set).

Standard Deviation is a measure of central tendency. As Standard Deviation is the measure of the central tendency of Cmax set, a small value means that the Cmax-s are in the vicinity of the Mean. A large standard deviation value means that the Cmax values are farther away from the Mean. On the 200 and 500 benchmark sets, high standard deviations indicates that **TSA**'s exploration has conducted far distant from NEH[14] solution. For 20 jobs and 10 machines, 20 jobs and 20 machines and 50 jobs and 20 machines benchmark sets the Score is 3 or 4 and for many other sets Score is 2 (95% of all Cmax are within two standard distributions).

When Score=1, the global's makespan (Cmax) is located at a short distance to Mean. In order to see if these results are statistically significant it is provided the 95% confidence interval. The width of the confidence interval depends on the large sample size of the Cmax set. In some situations, the large sample size of Cmax and small standard deviation have combined to give very small intervals (TABLE 1).

TSA performs well with different heuristics, especially for the larger benchmark set. In TABLE 2 the proposed algorithm, **TSA** is compared with TS – a simplistic approach of the tabu search and with the best results of ALA algorithm extracted from the paper of Agarwal[1]. ALA is an improvement heuristic based on adaptive learning approach using a constructive heuristic and improving the solution by perturbing the data based on a weight factor and allowing a non-deterministic local neighbourhood search. TS is a version of **TSA** starting with NEH[14] where random interchange of two jobs indexes alters the current solution and without the formulas 4 and 5 and the buffer zone. **TSA** is compared with the other heuristics proposed by Gupta[16] – called here H and by Eskenasi[13] marked here as HGA. HGA is a hybridization of the genetic algorithm with the iterated greedy search algorithm. H is similar to CDS heuristic[11] and converts the original m-machines problem into m-1 artificial 2-machines problems. Johnson's rule[17] is then applied to first artificial 2-machine problem to determine the sequence of jobs and the process is repeated by reducing the weight parameter until m-1 sequences are found.

TABLE 2. The average of the GAPS for Taillard's set for HGA,TSA,H

Taillard sets	**TSA**	**TS**	**ALA**[1]	**Taillard's set**	**HGA**[13]	**TSA**	**H**[16]
20 jobs sets	0.21	1.07	0.26	20 jobs and 5 machines	0.04	0.26	7.7
				20 jobs and 10 machines	0.03	0.13	10.61
				20 jobs and 20 machines	0.03	0.25	8.76
50 jobs sets	0.95	1.15	2.62	50 jobs and 5 machines	0.01	0.11	4.09
				50 jobs and 10 machines	0.73	1.04	10.96
				50 jobs and 20 machines	1.18	1.70	12
100 jobs sets	0.79	2.84	2.04	100 jobs and 5 machines	0.01	0.12	2.88
				100 jobs and 10 machines	0.26	0.74	7.64
				100 jobs and 20 machines	1.63	**1.51**	10.53
200 / 500 jobs sets	0.93	1.13	2.07	200 jobs and 10 machines	0.23	0.62	5.32
				200 jobs and 20 machines	1.54	1.27	9.4
				500 jobs and 20 machines	-	**0.92**	6.29

6. CONCLUSION

Since the PFSP is NP-hard for more than two machines, the advantage of this approach compared to other heuristics and meta-heuristics is that the medium and large problems can be solved optimally in this way. In general, since the number of jobs and the number of machines can be high, it is difficult to find the right solution with an exact method. In the proposed approach, tabu search starts with NEH[14] and marches through the non-explored areas. The adaptive distance that is used on filtering the candidate's makespans is combined with the buffer zone that collects all the "invisible" neighbors. With the adaptive distance, the speed of the search process is improved. This strategy can be extended on future researches on PFSP with a set of jobs constituting a production's batch.

References

[1] A.Agarwal, S.Colak, and E.Eryarsoy. Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach. *European Journal of Operational Research 169* (2006), 801–815.

[2] Baker, K. R. Introduction to sequencing and scheduling.

[3] C.R.Reeves, and T.Yamada. Genetic algorithms, path relinking and the flowshop sequencing problem. *Evol Comput 16* (1998), 230–234.

[4] D.G.Dannenbring. An evaluation of flow-shop sequence heuristics. *Management Science 23* (1977), 1174–1182.

[5] D.S.Palmer. Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near-optimum. *Operational Research Quarterly 16* (1965), 101–107.

[6] E.Nowicki, and C.Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research 91* (1996), 160–175.

[7] E.Taillard. Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of Operational Research 6* (1990), 65–74.

[8] E.Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research 64* (1993), 278–285.

[9] F.Glover. Tabu search-part i. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[10] G.Ochoa, and N.Veerapen. Deconstructing the big valley search space hypothesis. In *Evolutionary Computation in Combinatorial Optimization* (2016), F. Chicano, B. Hu, and P. García-Sánchez, Eds., Springer International Publishing, pp. 58–73.

[11] H.G.Campbell, R.A.Dudek, and M.L.Smith. A heuristic algorithm for the n-job, m-machine, sequencing problem. *Management Science 6* (1970), 630–637.

[12] J.N.D.Gupta. A functional heuristic for the flow-shop scheduling problem. *Operational Research Quarterly 22* (1971), 39–47.

[13] M.Eskenasi, and M.Mehrandezh. The permutation flow-shop scheduling using a genetic algorithm-based iterative method. *Industrial Engineering and Management* (2016).

[14] M.Nawaz, E.Enscore, and I.Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA 11* (1983), 91–95.

[15] M.R.Garey, D.S.Johnson, and R.Sethi. The complexity of flowshop and job shop scheduling. *Mathematics of Operations Research 1* (2016), 117–129.

[16] R.A.Gupta, and S.Chauhan. A heuristic algorithm for scheduling in a flow shop environment to minimize makespan. *International Journal of Industrial Engineering Computations 6* (2015), 173–184.

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

[18] Z.A.Drezner. New heuristic for the quadratic assignment problem. *Journal of Applied Mathematics and Decision Sciences 6* (2002), 163–173.

Technical University of Cluj-Napoca, Department of Manufacturing Engineering, B-dul Muncii 103–105, 400641 Cluj-Napoca, Romania

Email address: cristina.dodu@tcm.utcluj.ro, mircea.ancau@tcm.utcluj.ro