International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
http://ijcjournal.org/

Integrated Flow Shop and Vehicle Routing Problem Based on Tabu Search Algorithm

Quang Chieu Ta[a*], Phuong Mai Pham[b]

[a,b]*Department of Artificial Intelligence, Thuyloi University, 175 Tay Son, Dong Da, Hanoi. Vietnam*
[a]*Email: quangchieu.ta@tlu.edu.vn,* [b]*Email*: *maihoamieu@gmail.com*

Abstract

We consider in this paper a m-machine permutation flow shop scheduling problem and vehicle routing problem (VRP) integrated [1]. The manufacturing workshop is a flow shop and there is only one vehicle available. We are interested in the minimization of the total tardiness. related to the delivery completion times. Therefore, we are faced to three interdependent problems: scheduling the jobs in the flow shop environment, batching the completed jobs, and routing the batches. Then, we present the resolution method based on Tabu search and the results that have been obtained. Computational experiments are performed on random data sets and show the efficiency of the methods. Finally, a conclusion and some future research directions are proposed.

Keywords: flow shop problem; vehicle routing problem; metaheuristic algorithm; Tabu search.

1. Introduction and notations

We consider that the jobs have to be delivered to the customers after their production by using a single vehicle. The processing time of each job on each machine and the due date of delivery for each job are known, and a matrix of travel times is given. The jobs have to be scheduled in a m-machine flow shop environment, then batches have to be defined (one batch corresponds to one trip of the vehicle) and a route has to be determined for each batch, so that the total tardiness of delivery is minimized. The vehicle routing problem consists in defining a route starting from the production site, visiting the customers associated to the jobs in the batch, and finishing at the production site. Each customer requiring goods is visited by the vehicle (see Figure 1). The aim of this paper is to propose an algorithm for scheduling the jobs on the m-machines flow shop, for constituting batches of jobs and for determining the vehicle routing for each batch, so that the total tardiness of delivery is minimized. This problem is clearly an NP-hard problem [2].

---

* Corresponding author.

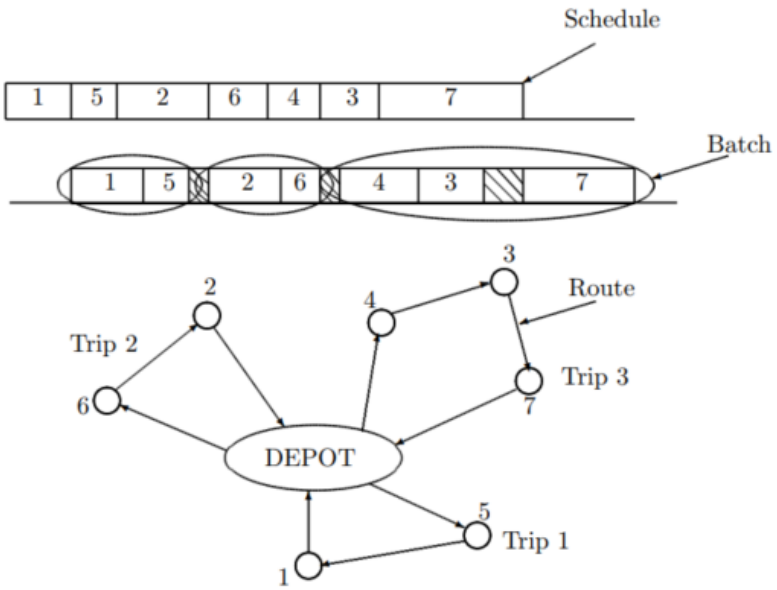

Figure 1: Illustration of the integrated production scheduling and vehicle routing problems.

The specific notations that are used in this chapter are the following:

- To each job J_j is associated a customer location number *j*, the location of the production facility has an index 0.
- The delivery time between each pair of locations *i* and *j* is denoted by $l_{i.j}$.
- The number of vehicles is equal to 1.

2. Tabu search algorithm

2.1. Coding of a solution

We use an array of *3n* elements to represent a complete solution. The first *n* elements represent the sequence of jobs (i.e. the schedule), the next $2n$ elements give for each trip the number of jobs in the trip and the list of jobs (the routing of these jobs is implicitely the order of the jobs in this list). In the following, the first part of the coding of a solution S (first n elements) is denoted by S^0 and the second part ($2n$ elements) is denoted by S^T. The second part is decomposed into several trips. Each trip λ of S^T is denoted by S^T_λ and is composed by the number of visits k_λ and the list of visits ($S^T_{\lambda[1]},\ldots, S^T_{\lambda[k_\lambda]}$). The coding is illustrated in Figure 2.

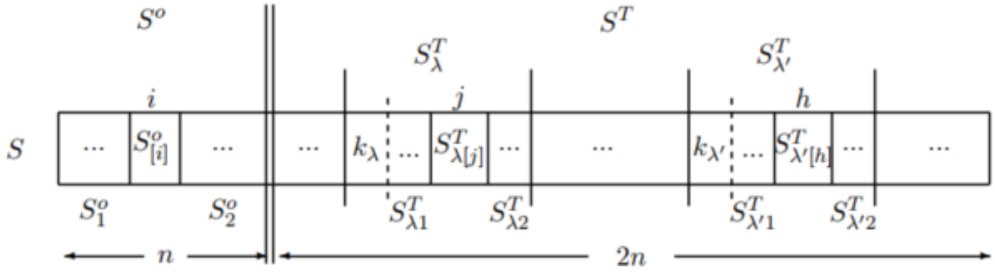

Figure 2: Illustration of the coding of a solution and notations.

Example: In the example presented in Figure 3, the schedule is {*J1, J5, J2, J6, J4, J3, J7*}, the number jobs in each trip is (2, 2, 3, 0, 0, 0, 0), i.e. two jobs in the first two trips and three jobs in the third trip, the remaining trips are empty.

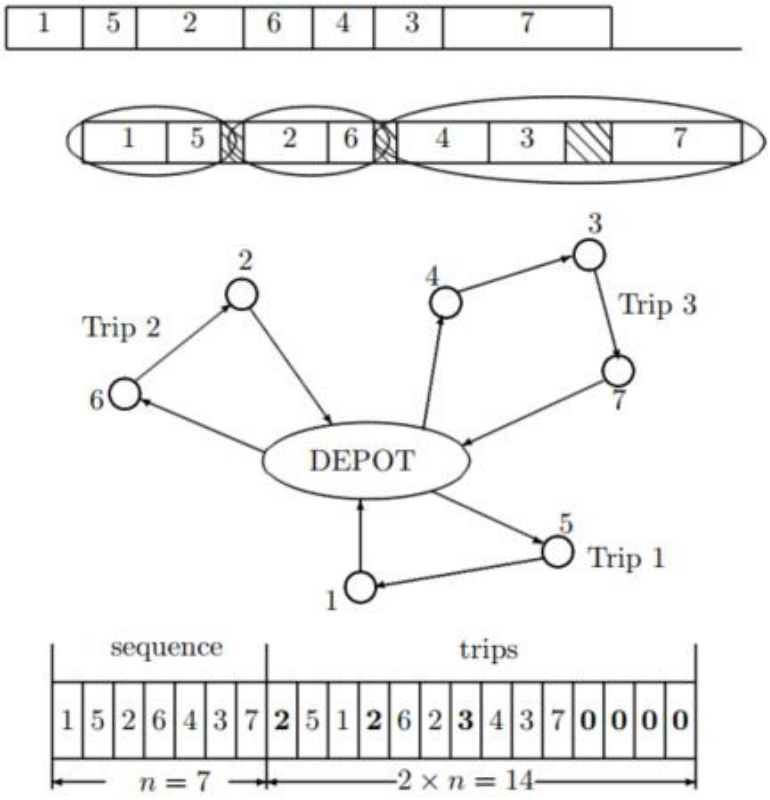

Figure 3: Example of the coding of a solution.

2.2. Initial solution

EDD algorithm is used for giving an initial solution for the scheduling problem. For the trips, each trip contains only one job. The customers are visited in the same order as in the sequence.

Example: An example of an initial solution is given in Figure 4.

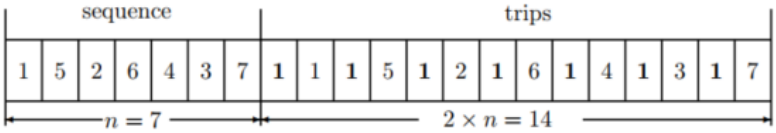

Figure 4: Example of initial solution.

2.3. Neighborhood definitions

We assume that $S = S^0 // S^T$ is the current complete solution with sequence S^0 and trips S^T, where:

- $S^0 = S_1^0/S_{[i]}^0/S_2^0/S_{[j]}^0/S_3^0$ with $S_{[i]}^0$ and $S_{[j]}^0$ the jobs in positions i and j ($i < j$) in S^0 and S_1^0, S_2^0 and S_3^0 three partial sequences.
- $S^T = k_1, S_1^T/\ldots/k_{\lambda-1}, S_{\lambda-1}^T/k_\lambda, S_\lambda^T/k_\lambda + 1, S_{\lambda+1}^T/k_n$, S_n^T with k_λ the number of jobs in trip λ. S_λ^T is the sequence of jobs in trip λ, ∀ λ ∈ {1, 2, …, n}.

If k_λ = 0 then S_λ^T is empty. Notice also that $\sum_{\lambda=1}^{n} k_\lambda = n$.

Neighborhood based on the sequence of jobs

We use *SWAP*^{0} operator for creating the neighbors of sequence S^0. This operator is the same as the one described [3].

- *SWAP*^{0}: A neighbor of S is created by interchanging the jobs in position i and j in sequence S^0. The corresponding jobs are also swapped in S^T. See Figure 5 for an illustration of this operator.

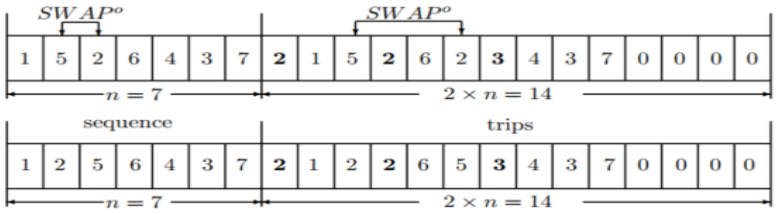

Figure 5: Illustration of SWAP^0 operator.

Neighborhood based on a single trip

We use *SWAP*^{T}, *EBSR*^{T}, *ESFR*^{T} and *Inversion*^{T} operators for creating the neighbors of a trip S_λ^T. These operators are the same as those described [3,4].

Given a sequence of visits S_λ^T of trip λ, two random positions i and j in S_λ^T ($i < j$ and $j \leq k_\lambda$): $S_\lambda^T = S_{\lambda 1}^T, S_{\lambda[i]}^T, S_{\lambda 2}^T, S_{\lambda[j]}^T, S_{\lambda 3}^T$.

- *SWAP*^{T}: A neighbor of S_λ^T is created by interchanging the jobs in position i and j, leading to sequence $S_\lambda^{T\prime} = S_{\lambda 1}^T, S_{\lambda[j]}^T, S_{\lambda 2}^T, S_{\lambda[i]}^T, S_{\lambda 3}^T$, See Figure 6 for example.

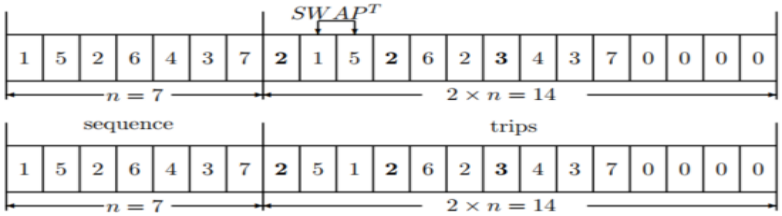

Figure 6: Illustration of SWAP^T operator for a trip.

- *EBSR*^{T}: A neighbor of S_λ^T is created by extracting the visit in position j and reinserting this visit backward just before the visit in position i, leading to sequence $S_\lambda^{T\prime} = S_{\lambda 1}^T, S_{\lambda[j]}^T, S_{\lambda[i]}^T, S_{\lambda 2}^T, S_{\lambda 3}^T$, See Figure 7 for example.

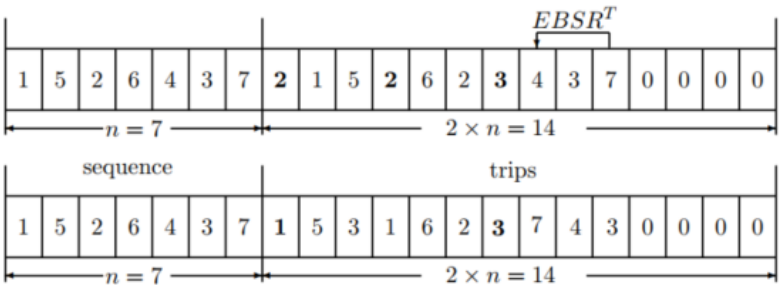

Figure 7: Illustration of EBSR^T operator for a trip.

- *ESFR*^{T}: A neighbor of S_λ^T is created by extracting the visit in position i and reinserting it forward immediately after the visit in position j, leading to a sequence $S_\lambda^{T\prime} = S_{\lambda 1}^T, S_{\lambda 2}^T, S_{\lambda[j]}^T, S_{\lambda[i]}^T, S_{\lambda 3}^T$. See Figure 8 for example.

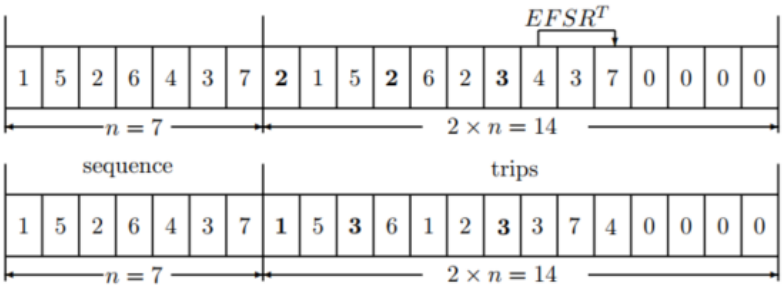

Figure 8: Illustration of ESFR^T operator for a trip.

- *Inversion*^{T}: A neighbor of S_λ^T is created by inserting $S_{\lambda[i]}^T, S_{\lambda 2}^T, S_{\lambda[j]}^T$ in the inverse order between $S_{\lambda 1}^T$ and $S_{\lambda 3}^T$. See Figure 9 for example.

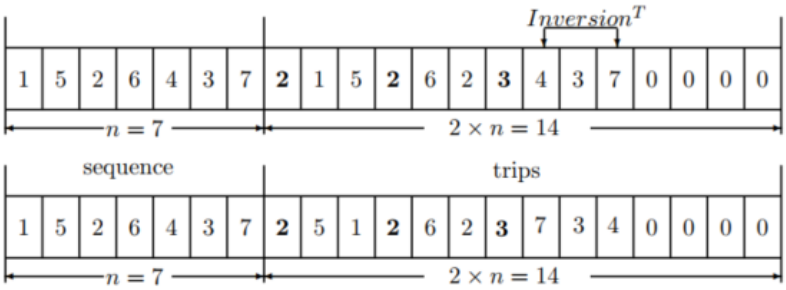

Figure 9: Illustration of Inversion^T operator for a trip.

Moves and selection of the best neighbor

The aim of the problem is to minimize the total tardiness. The best neighbor in the candidate item that is non-tabu and with the smallest total tardiness. For managing the tabu list, we use the first-in-first-out (FIFO)

strategy. Old attributes are deleted as new attributes are inserted.

Tabu list

We define three tabu lists and the size of the tabu lists is fixed. An element of the tabu list is defined by (λ, μ, J_i, J_j), λ, μ are the indexes of two trips, J_i and J_j are two jobs:

- For a swap of two jobs J_i and J_j in the sequence, we insert in the Tabu list the element (0, 0, J_i, J_j). For example, see **Error! Reference source not found.**, the element inserted in the Tabu list is (0, 0, J_5, J_2). In this case $\lambda = \mu = 0$.
- For a move in two trips λ and μ, we insert in the Tabu list the element (λ, μ, J_i, J_j) where i and j are the positions that are concerned in trips λ and μ respectively. For a move in a single trip, we have $\lambda = \mu$. For example, see **Error! Reference source not found.**, where the tabu list is (1, 1, J_1, J_5).

Stopping condition

The algorithm is stopped when the time limit has been reached. This time limit is denoted by *TimeLimitTS* = *n(m/2)t ms* [5], where $t = 90$.

Detailed algorithm

The detailed TS algorithm is given in Algorithm 1

- FlagSwap^o allow to make a selection of the neighbor operator of sequence.
- LimitSwap^o allow to limit the size of the neighborhood in sequence.
- FlagOpera^T allow to make a selection of the neighbor operators in sequence of a trip (FlagOperaT $\in$ {FlagSwapT, FlagEBSRT, FlagEF SRT, FlagInversionT}).
- LimitOpera^T allow to limit the size of the neighborhood in a trip (LimitOpera^T $\in$ {LimitSwap^T, LimitEBSR^T, LimitEFSR^T, LimitInversion^T}).
- FlagOpera$^T{}_2$ allow to make a selection of the neighbor operators in sequence of two different trips (FlagOpera^T 2 $\in$ {FlagSwap$^T{}_2$, FlagEBSR$^T{}_2$, FlagEFSR$^T{}_2$}).
- Del(T) deletes the upper element of TS
- Add(T,(λ, μ, J_i, J_j)) adds element (T,(λ, μ, J_i, J_j)) to T (tabu list), $\forall\lambda, \mu \in \{0, 1, 2, ...\}$. In Algo. 15, the Test(SWAP^o), Test(Opera^T), Test(Opera$^T{}_2$) are described in Algorithm 2, 3, 4:

1: **Initialization**

2: S_0 = initial solution, S = current solution

3: S' = S_0 // best solution of N(S)

4: S^* = S_0 // best solution of N(S) and non-tabu

5: $f^* = f(S_0)$ // f∗ value of S^* and $f(S_0)$ value of S_0

6: T = ∅ // T is the tabu list

7: while (CPU ≤ TimeLimitTS) do

8: $f(S_0) = \infty$,

9: //Selecting neighbor in sequence

10: for i = 0 to n − 1 do

11: for j = i + 1 to n do

12: Test($SWAP^o$)

13: end for

14: end for

15: //Selecting neighbor of a trip

16: for λ = 0 to n − 1 do

17: for i = 0 to nbjoboftrip[λ] − 2 do

18: //nbjobof trip[λ] is the number jobs of trip λ

19: for j = i + 1 to nbjoboftrip[λ] − 1 do

20: Test($Opera^T$)

21: end for

22: end for

23: end for

24: //Selecting neighbor in two trips

25: for λ = 0 to n − 2 do

26: for μ = 0 to n − 1 do

```
27:                     for i = 0 to nbjoboftrip[λ] − 1 do
28:                     //nbjobof trip[λ], nbjoboftrip[μ] are the number jobs of trip λ, μ
29:                         for j = 0 to nbjoboftrip[μ] − 1 do
30:                             Test(Opera^T_2)
31:                         end for
32:                     end for
33:             end for
34:     end for
35:     if (f(S_0) < f*) then S* = S_0, f* = f(S), end if
36:     if (SizeTabu ≥ TabuMax) then Del(T) end if
37:     Add(T,(λ, μ, i, j))
38: end while
```

Algorithm 1. Tabu search algorithm for scheduling and vehicle routing

```
1: if (FlagSwap^o = 1) and (j − i ≤ LimitSwap^o) then
2:     S = S_0, f(S) = f(S_0), SWAP^o (S,(0, 0, i, j)),
3:     if ((0, 0, i, j) ∉ T) then
4:         Calculate(f(S)),
5:         if (f(S) < f(S_0)) then S_0 = S, f(S_0) = f(S), move = (0, 0, i, j), end if
6:     end if
7: end if
```

Algorithm 2. Test($SWAP^o$)

```
1: if (FlagOpera^T = 1) and (j − i ≤ LimitOpera^T) then
```

2: S = S_0, f(S) = f(S_0), Opera^T(S,(λ, λ, i, j)),

3: if ((λ, λ, i, j) ∉ T) then

4: Calculate(f(S)),

5: if (f(S) < f(S_0)) then S_0 = S, f(S_0) = f(S), move = (λ, λ, i, j), end if

6: end if

7: end if

Algorithm 3. Test(Opera^T)

1: if (FlagOperaT2 = 1) then

2: S = S0, f(S) = f(S0), OperaT2(S,(λ, μ, i, j)),

3: if ((λ, μ, i, j) ∉ T) then

4: Calculate(f(S)),

5: if (f(S) < f(S0)) then S0 = S, f(S0) = f(S), move = (λ, μ, i, j)), end if

6: end if

7: end if

Algorithm 4. Test(Opera$^T_2$)

3. Computational experiments

We present in this section the generation of data and we discuss the results.

Generation data

We have tested the algorithms on a PC Intel core^{TM}i5 CPU 2.4GHz. Data sets have been randomly generated (notice that there is no benchmark instance for the m-machine flow shop and vehicle routing problem integrated). The processing times $p_{i,j}$ have been generated in [1,100], the due dates d_j have been generated in [50, 50n], the position of "custom" j is given by its coordinates (x_j, y_j) generated in [1, 70] (see Figure 10). The delivery time $l_{i,j}$ is the classical Euclidian distance $l_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$. Ten instances are used for each combination of n and m, with n ∈ {20, 30, 50, 70, 100, 150, 200} and m ∈ {2, 4}. For the TS algorithm, some preliminary experiments have conducted to the following parameters settings: *TimeLimitTS* = 10 seconds.

Tabu list = {10, 40, 60,120} elements.

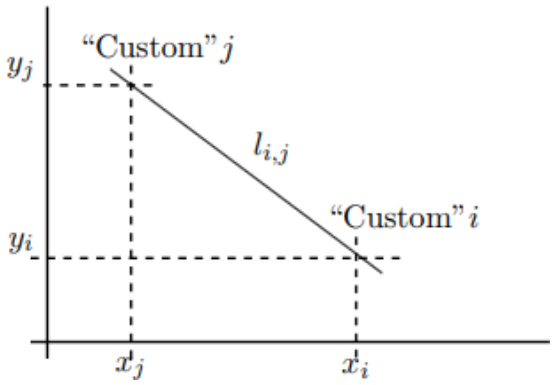

Figure 10: Illustration of calculation of $l_{i,j}$.

4. Results

In **Error! Reference source not found.**, column 'Best' for TS_X ($X \in \{10, 40, 60, 120\}$) indicates the number of times this method strictly outperforms other method. Column 'ΔTS_X' indicates the average deviation between TS_X and the best method between TS_{10}, TS_{40}, TS_{60} and TS_{120}.

$$\Delta TS_X = \frac{TS_X - \min(TS_{10}, TS_{40}, TS_{60}, TS_{120})}{TS_X}$$

Table 1: Tabu search results.

n x m	TS_{10}		TS_{40}		TS_{60}		TS_{120}	
	Best	ΔTS_{10}	**Best**	ΔTS_{40}	**Best**	ΔTS_{60}	**Best**	ΔTS_{120}
20 × 2	3	17.58%	7	4.16%	3	17.58%	0	24.81%
30 × 2	1	11.62%	7	13.34%	3	10.97%	0	30.19%
50 × 2	0	17.40%	1	21.81%	2	12.49%	7	3.23%
70 × 2	0	16.12%	0	21.88%	2	10.99%	8	2.50%
100 × 2	1	19.24%	1	25.61%	0	14.93%	8	1.65%
150 × 2	0	23.42%	0	15.27%	4	5.29%	6	5.95%
200 × 2	0	26.35%	4	5.60%	3	5.67%	3	8.19%
20 × 4	0	9.51%	9	0.00%	1	9.51%	0	12.04%
30 × 4	0	9.28%	10	0.00%	0	9.28%	0	17.63%
50 × 4	0	8.67%	5	3.19%	2	3.27%	1	4.60%
70 × 4	1	7.34%	2	9.54%	3	5.38%	5	4.47%
100 × 4	0	17.60%	3	9.84%	2	9.23%	5	4.75%
150 × 4	0	19.08%	2	2.84%	4	3.08%	4	4.10%
200 × 4	0	18.52%	3	2.60%	5	4.16%	2	10.73%
	6	14.98%	54	10.46%	34	8.49%	49	9.94%

We can see that the TS_{40} (with tabu list is 40) leads to the best results with a number of best solutions equal to 54. On average, the deviation between the solutions returned by this method and the best solutions is 10.46%. This value is around and 14.98% for TS10, 10.46% for TS_{40}, 8.49% for TS_{60} and 9.94% for TS_{120}.

In Table 2, column 'Best TS < EDD' indicates the number of times the method TS outperforms method EDD,

column Cpu(s) indicates the average computation time of TS per 10 instances. Column 'Δ' indicates the average deviation between EDD and TS.

$$\Delta = \frac{EDD - TS}{EDD}$$

Table 2 : Comparison of the TS_{40} and EDD algorithm.

n × m	**Best TS<EDD**	**Cpu(s)**	**Δ**
20 × 2	10	10.00	72.0%
30 × 2	10	10.00	79.7%
50 × 2	10	10.00	88.3%
70 × 2	10	10.01	85.2%
100 × 2	10	10.01	91.4%
150 × 2	10	10.03	90.0%
200 × 2	10	10.06	81.3%
20 × 4	10	10.00	55.9%
30 × 2	10	10.00	69.2%
50 × 4	10	10.00	75.9%
70 × 4	10	10.01	71.4%
100 × 4	10	10.01	83.7%
150 × 4	10	10.03	77.7%
200 × 4	10	10.03	70.4%

As we can see in Table 2, TS improves significantly the initial solution given by EDD, with 78,0% of improvement in average.

5. Conclusions and discussions

We approach a problem where a m-machine permutation flow shop scheduling problem and a vehicle routing problem are integrated to minimize the total tardiness. To our knowledge, this is the first time that this problem is approached in the literature. We present a direct coding for a complete solution and a neighborhood method for finding a sequence and trips. We propose a tabu search algorithm for this problem, the first results show that the TS greatly improves the initial solution given by EDD and where each trip serves only one job at a time.

In the future, the first research directions are about the evaluation of the method. We will develop genetic algorithms and other methods in order to see the real quality of the tabu search. Then, we will combine mathematical programming and local search (matheuristic), in order to see if matheuristic are performing methods for this problem.

Acknowledgements

The authors would like to thank to Professor Jean-Charles Billaut of University of Tours, France who provided insight and expertise that greatly assisted the research.

References

[1] Q. C. Ta, J. C. Billaut, J. L. Bouquard and P-E. Morin. "Minimisation de la somme des retards pour un problème d'ordonnancement de type flow-shop à deux machines et un problème de livraison intégrés." *15ème congrès de la Société de la Société Française de Recherche Opérationnelle et d'Aide à la Décision Roadef",* 2014. Bordeaux, France.

[2] J. K. Lenstra and A. H. G. Rinnooy Kan. "Complexity of vehicle routing and scheduling problems". *Networks*, vol 11, pp. 221-227, 1981.

[3] Q. C. Ta, J. C. Billaut and J. L. Bouquard. "An hybrid metaheuristic. a hybrid lower bound and a tabu search for the two-machine flowshop total tardiness problem." Proceedings of *the 10th IEEE RIVF International Conference on Computing and Communication Technologies (RIVF'13)*, 2013, Hanoi, Vietnam.

[4] Q. C. Ta, J. C. Billaut and J. L. Bouquard. "Minimizing total tardiness in the m-machine flow-shop by genetic and matheuristic algorithms". *Journal of Intelligent Manufacturing*, vol 29, pp. 617- 628, 2015.

[5] E. Vallada, R. Ruiz and G. Minella. "Minimising total tardiness in the m- machine flowshop problem: A review and evaluation of heuristics and metaheuristics". *Computers & Operations Research*, vol 35, pp. 1350-1373, 2008.