Mixed Model Assembly Line Sequencing to minimize delays using meta-heuristics

Abduh Sayid Albana, Karim Aroui, Gülgün Alpan, Yannick Frein

MIXED MODEL ASSEMBLY LINE SEQUENCING TO MINIMIZE DELAYS USING META-HEURISTICS

A.S. Albana1[1*], K. Aroui[1], G. Alpan[1], and Y. Frein[1]
[1]Univ. Grenoble Alpes, G-SCOP, F-38000 Grenoble, France
CNRS, G-SCOP, F-38000 Grenoble, France
abduh.albana@gmail.com

ABSTRACT

Nowadays, the customers are increasingly demanding, pushing the companies to offer highly diversified products. This requires that different kinds of products be manufactured in intermixed product sequences on the same line. Such assembly lines are called the mixed-model assembly lines (MMAL). Workers and machinery have to be flexible to reduce the setup times and costs. A good vehicle sequence in MMAL can have many positive effects on MMAL: It can permit producing more products in a shorter time period (providing cost reductions), it can also improve the work conditions by balancing the workload of the operators. In this article, we are interested in the sequencing of a mixed model assembly line for Truck Industry. In the literature, different objectives exist to solve the MMAL sequencing problem. In this article, we present methods to minimize the total work overload. In a previous work, a linear programming (LP) approach has been proposed for this problem. The MMAL is known to be an NP-hard problem. The exact methods such as LP can only handle small problems and their applications are limited in an industrial context. Therefore, we present here solutions based meta-heuristics. Three types of meta-heuristic algorithms are used in this research: Genetic Algorithm (GA), Simulated Annealing (SA), and finally a hybrid method based on both Genetic Algorithm and Simulated Annealing (GASA). Numerical tests are carried out to compare the performance of the proposed algorithms. For small instances, a benchmark data from the literature is used to compare the performance of the meta-heuristics versus the optimal solution found by the LP approach, based on the computational time and the quality of the solutions. The comparisons are also made for larger instances, for some generated data and the data from an industrial case study.

Keywords: Mixed Model Assembly Line, Sequencing Problem, Meta-heuristics

1 INTRODUCTION

Nowadays, the customer need in specific product increases. This forces the manufacturer to produce different types of products. High product variety makes the production management more difficult. The use of flexible workers and machinery is a common solution so that different kinds of products can be manufactured in intermixed product sequences on the same line, reducing the setup times and cost. Such production lines are called the mixed-model assembly lines (MMAL) [1]. This type of assembly line is generally used in automotive industry and consumer goods industries such as electronics, white goods, furniture and clothing.

There are two main problems in MMAL: assembly line balancing and product sequencing. Assembly line balancing problem deals with the assignment of tasks to workstations. The most common objective is to minimize the number of stations needed to manufacture a product in a line given a fixed cycle time, equivalent to a fixed production rate [2]. And

* Corresponding Author

sequencing problem deals with sequencing different product models launched on an assembly line, so that work overload at the stations induced by direct succession of multiple labor-intensive models is avoided or part rate usage is minimized [3]. Assembling various options leads to variations in processing times at work stations [1]. In automotive production, for instance, the installation of an electrical sunroof requires a different amount of time than that of a manual one. The work overload appears when the operator cannot finish the required tasks on a product within the predefined time window. The work overload refers to the remaining work. If several work intensive models follow each other at the same station, work overloads might occur, which need to be compensated, e.g., by additional utility workers. Work overloads can be avoided if a sequence of models is found, to alternate the high work-intensive models with the less work-intensive ones. This Work overload is referred as delay in our research.

Numerous researchers have worked on both problems. Amen [4], Chaves et al. [5], Bautista and Pereira [2], and many others work on assembly line balancing. Hyun et al. [6], Ponnambalam et al. [7], Cano-Belmán et al. [8], Rahimi-Vahed et al. [9], Aroui et al. [10] and many others work on product sequencing. In this article we will deal with the product sequencing problem.

On product sequencing problem, some of the researchersconsider multi-objectives, such as: minimizing total utility work, minimizing total setup cost and minimize total production rate variation [6], minimize total production rate variation and minimizing total setup cost [7], and the other work in single objective minimizing, such as: total utility work (Cano-Belmán et al. [8]). Some of them solve it using exact methods, e.g. Mixed Integer Linear Programming [11], Bounded Dynamic Programming [12]. Some use meta-heuristic, e.g. Genetic Algorithm (GA) [6], [7], [13], Particle Swarm Algorithm [9], [14], Greedy Randomized Adaptive Search Procedure (GRASP) [15], and Ant Colony Optimization [16].

In 2013, Aroui et al. [11] work on the sequencing problem of MMAL. They choose to act directly on the work overload to minimize the delay. This kind of approach is poorly developed in the literature [11]. Their work is based on an industrial case of Bourg-en-Bresse plant of Renault Trucks. The truck assembly line is a typical MMAL with highly diversifying products compared to automotive industry. The number of vehicles to produce per day is much lower. They present a mixed integer linear programming (MILP). The main findings of Aroui et al. [11] is that the sequence generated at the end of 2 hours is a better solution than the actual procedure at Renault Trucks, however, no optimal solution can be found.

The method presented in Aroui et al. [11] gives good results, despite the long calculation time. In reality, managers need to make fast decisions, especially in production lines. To make a fast decision with a good solution quality, an approach less expensive in computation time is needed.

Based on problems from Aroui et al. [11], this research aims to find another approach for solving thesame problem with faster calculation times but with the same solution quality. We use three types of heuristics, Genetic Algorithm (GA), Simulated Annealing (SA) and combined Genetic Algorithm – Simulated Annealing (GASA). Then, we compare the results of the algorithms (GA, SA and GASA) among themselves and with the results of Aroui et al. [11] based on the performance and computational time.

2 DESCRIPTION OF THE PROBLEM

Mixed Model Assembly Line (MMAL) is a special assembly line where there is a single production line used for assembling multiple type of products. Normally this type of production system can be found in the automotive industry [1]. An illustration of a typical MMAL is given in Figure 1. The products in MMAL move on a continuous transportation system such as a belt conveyor. Each product has its own processing times, however, the cycle time at the workstations is constant.

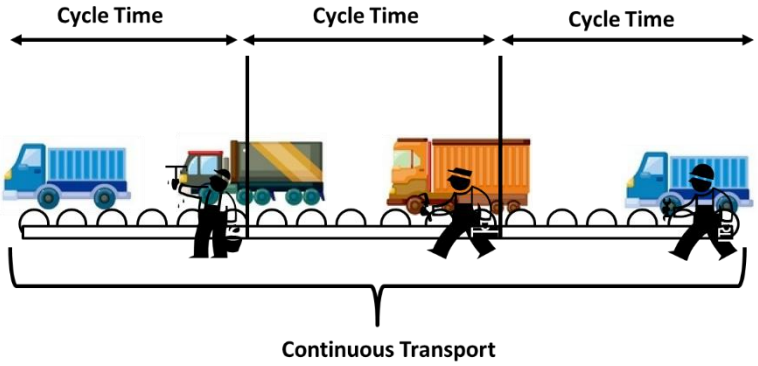

Figure 1 Illustration of MMAL

Based on Boysen et al. [1],sequencing problemscan be categorized as three types:

1. *Mixed-model sequencing*: aims at minimizing sequence-dependent work overload based on a detailed scheduling which explicitly takes operation times, worker movements, station borders and other operational characteristics of the line into account.
2. *Car sequencing*: To avoid the significant effort of data collection that accompanies mixed-model sequencing, car sequencing attempts to minimize sequence-dependent work overload in an implicit manner.
3. *Level scheduling*: While the first two approaches aim at minimizing violations of capacity constraints, level scheduling seeks to find sequences that are in line with the JIT-philosophy. For this purpose "ideal" production rates are defined and models are sequenced in such a manner that deviations between actual and ideal rates are minimized.

Each of those types has more detailed sub types; based on the type of the production line, station boundary, objective functions, etc. (for further detail see Boysen et al., 2009). In this research, wefocus on the Mixed Model Sequencing Problem where the objective function is to minimize the work overload (MMSP-W). For our type of problem, we have several hypotheses:

- Line balancing has already been done. All tasks are assigned to different workstations,
- Products move on the line at a constant speed. There are no inventory buffers between workstations,
- Products require a specific amount of work (tasks with operation time) for each position. The operation times are deterministic. Setup time and the time required by operators to return to his initial position at the end of a task are included in the operating time.

The operators here areregular operators who areassigned tasks on every vehiclem_i (see figure 2). Every vehicle requires a specific amount of work (tasks with operation time) for each position. And the cycle time is the time between two successive tasks. We denote cycle time as γ . A delay happens when the task on a vehicle cannot be completed within the cycle time (see figure 2).

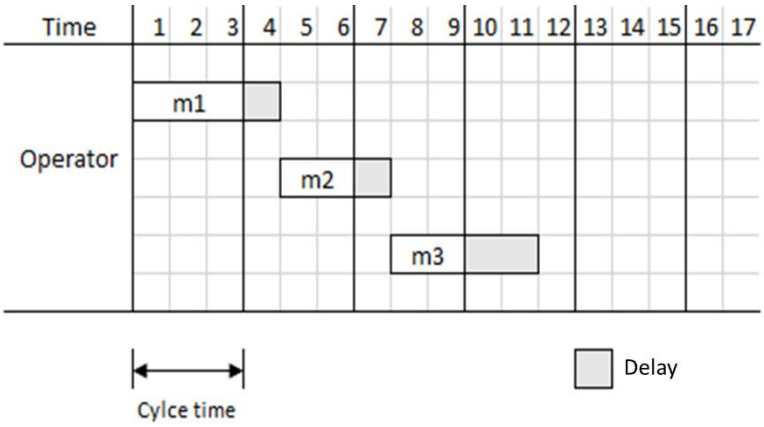

Figure 2 Illustration of delay in operator (Aroui et al. [11])

As mentioned before, Aroui et al. [11] formulate this problem by MILP. For the sake of completeness, we recall their model below:

$$Min \sum_{p=1}^{p} \sum_{j=1}^{n} r_{jp} \quad (1)$$

Subject to

$$c_{jp} = r_{j-1,p} + \sum_{i=1} d_{ip} x_{ij} \qquad \forall j \forall p \quad (2)$$

$$r_{jp} \geq 0 \qquad \forall j \forall p \quad (3)$$

$$r_{jp} \geq c_{jp} \qquad \forall j \forall p \quad (4)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \quad (5)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \quad (6)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \forall j \quad (7)$$

$$c_{jp}, r_{jp} \in R \qquad \forall j \forall p \quad (8)$$

Where,

n : Number of products

i : Product index

j : Position index

p : Operator index

γ : Cycle time

t_{ip} : Processing time required by product *i* for operator *p*

d_{ip} : Delay or tardiness for operator type 1, $d_{ip} = t_{ip} - \gamma$

x_{ij} : Equal to 1 if product *i* is assigned to position *j*, otherwise 0

c_{jp} : Delay or idle time for operator *p* for product on position *j*

r_{jp} : Total Delay operator *p*for product on position *j*

Objective function (1) minimizes the total delay. Constraint (2) establishes that the delay or idle time of the product positioned in *j* is its own delay (or idle time) in addition to overload of the position$j-1$. Constraints (3) and (4) indicate that $r_{jp} = max(0, c_{jp})$since the objective function takes into account only the delays (and not idle times). Constraint (5) guarantees that only one position can be assigned to each product; constraint (6) indicates that only one

product can be assigned in each position of the sequence; and, finally, constraint (7) requires that the assigned variables are binary.

3 META-HEURISTIC APPROACH

The objective is to find solutions having a similar quality reported by Aroui et al. [11], but faster in computation time. Meta-heuristics are chosen because they are well adapted for difficult problems in industrial context. A well-designed meta-heuristic method can usually provide a solution that is nearly optimal in short execution time.In this research, we propose three meta-heuristics: GA, SA and combination of both (GA-SA). We use two cases for numerical tests: small instances (academic instances) and big instances (Renault Trucks Case). The resultsare compared to assess the performance of each meta-heuristic. The analyses are done based on the quality of the solution and the computational time.

3.1 Genetic Algorithm

Genetic Algorithm (GA) is a meta heuristic founded by John Holland in the 1960s[17]. This algorithm is based on the Darwin's theory of natural selection. There is an initial population consisting of different chromosomes. These chromosomes represent the solution: in our case it is a vehicle sequence. This population will evolve because of genetic operations applied to improve the solution. There are three basic elements (operations) in Genetic Algorithm: reproduction, crossover, and mutation. Another element of the GA is called the fitness function. The fitness function shows the ability of individuals to survive in the population [17], [18] and is measured by the objective function to be optimized [18]. For the maximization problem, the fitness is equalor linear to the objective function (*f(x)*). But for the minimization problem, the fitness is expressed as the inverse of the objective function (*1/f(x)*) [19]. InSantosa & Willy[20]the fitness function is given as in equation (9). Constant 1 ensures that the fitness does not tend to ∞, when the *f(x) = 0*.

$$F(x) = \frac{1}{1+f(x)} \qquad \textbf{(9)}$$

Where,

$F(x)$: Fitness function,

$f(x)$: Objective function.

In GA, there are several parameters, such as: number of chromosomes in the population, number of elite chromosomes, crossover probability, and mutation probability. Crossover probability (pc) is usually very high, typically in the range of 0.7 ~1.0. The mutation probability (pm) is usually small (usually 0.001 ~ 0.05). If pc is too small, the crossover occurs sparsely, which is not efficient for evolution. If the mutation probability is too high, the solutions could still 'jump around' even if the optimal solution is approaching [17]. Elite chromosomes are the chromosomes, which will be carried over to the new generation without being modified. The purpose of the elitism is to keep a good solution in the population. The pseudo code for the proposed genetic algorithm is given in Figure 3.

We note that some improvements are brought to the classical genetic algorithm that doesn't perform well using the parameters mentioned by Yang [17]. So, we improve the native GA. In improved GA, we do several things:

- Crossover and Mutation are always done in each iteration ($p_c = p_m = 1$).
- The offspringis only accepted if it has better fitness function than their parent.
- Mutation is doneon the three best chromosomes with different types of mutation (Swap, Flip, and Slide).

```
Genetic Algorithm for MMSP-W

Generate Initial sequence
Calculating the objective
Calculating fitness
Find The best sequence

while Condition not meet
      do Elitism
      Re-Calculate Objective
      Re-Calculating fitness

      Chromosome Selection & Cross Over
     do Fortune Wheel Selection
      if random number < Crossover Probability
         do Crossover
         if The child is better
            Replace parent.
         else
            Cancel the crossover
         end
      end

      Re do the Elitism

      Mutation
      if random number < Mutation Probability
         Mutate the Best to get Three New Routes
         Mutation 1 : Flip
         Mutation 2 : Swap
         Mutation 3 : Slide
      end
end
```

Figure 3Pseudo code of Improved Genetic Algorithm for MMSP-W

The details of how to choose the parents, crossover and mutation will be described in the next section.

3.1.1 Chromosome & Genes

The essence of genetic algorithms isthe encoding of an optimization function as arrays of bits or character strings to represent the chromosomes,the manipulation operations of strings by genetic operators, and the selection according to their fitness with the aim to find a solution to the problemconcerned[17]. These arrays of bits or strings are known as genes. In our case, a chromosome is a sequence of products. And genes are product *i* in position *j*in the vehicle sequence.

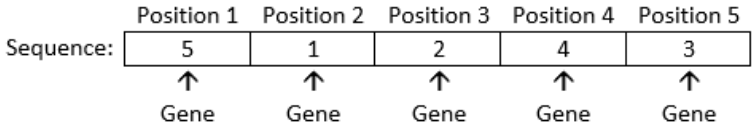

Figure 3.4 Representation of Chromosome and Genes

3.1.2 Elitism

The best individuals with higher fitness should be preserved and passed onto the next generation. Elitism is a process to select the most fit individual (in each generation) which will be carried over to the new generation without being modified by genetic operators [17]. In this research, we keep 4 individuals in each iteration as elite chromosomes.

3.1.3 Parent Selection &Crossover

Parent is the selected chromosome to do the crossover process. And parent selection is done based on fitness.Parent with high fitness ismatched with another high fitness parent to do reproduction (crossover). Then low fitness parent is matched with another low fitness one. Parent that has been chosen cannot be chosen another time.

Crossover is the exchange of genetic material between parent chromosomes that results in newchromosomes (child). Crossover is done for all chromosomes, except the elite chromosomes. Parentsare matched based on probability. More detailed explanation can be seen in Figure 5. In this example, there are 6 parents. Parent 1 is matched with parent 3 in the first draw. Then, in the second draw among the remaining parents, parent 2 is paired with parent 4 (according to probabilities). And finally, parent 5 is paired with parent 6. With this scheme, all parents have a pair and do the crossover process. No parents are matched with themselves or do the crossover twice.

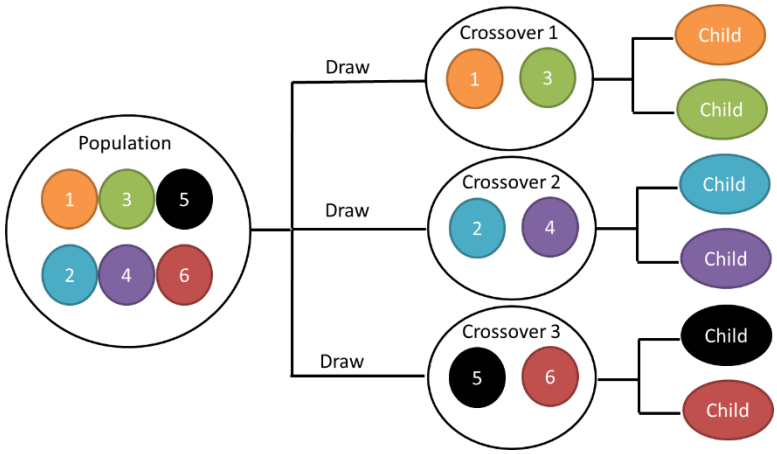

Figure 5 Parent selection and crossover process

In crossover itself, only one point crossover is used. It is when a single crossover point on both parents' chromosome is selected. All data beyond that point in either chromosomestring is swapped.The crossover point is chosen arbitrarily. In general, the crossover process is done as illustrated in Figure 6.

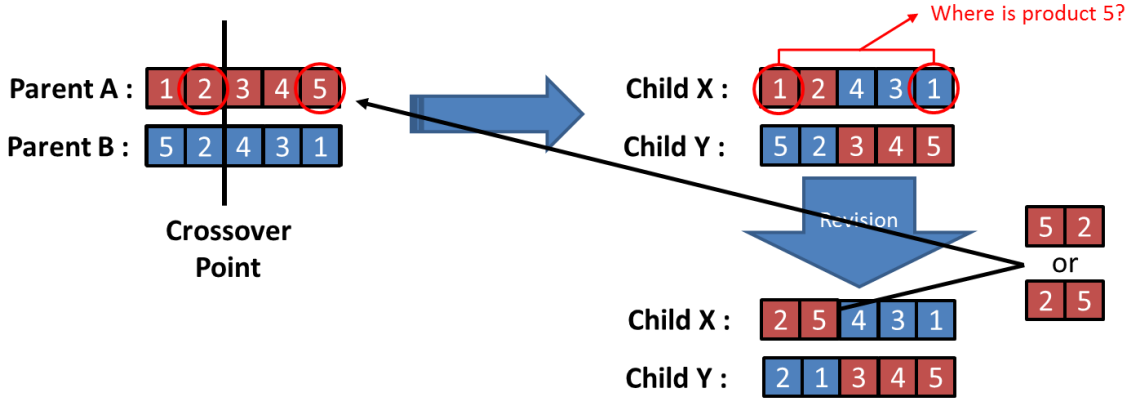

Figure 6 Crossover process

We have two parents, A and B, and one point crossover after gene number 2 (position 2 is randomly picked). After the crossover operation, we obtain 2 children, X and Y. Child X inherits his first two genes from parent A and the remaining 3 genes from parent B. (the inverse is true for child Y). None of the children is a valid offspring since a gene is repeated twice and there is one product that hasn't been listed. Here we have to revise part before the crossover point (gene no. 2). For the child X, we have to revise the first two genes. And

the product that hasn't been listed is product no. 5. Alternatively, we have two options to replace the first two genes. It can be 5-2 or 2-5. Because this part is coming from parent A and parent A has a sequence of 1-2-3-4-5 (see Figure 6), we put the sequence of 2-5 instead of 5-2 for the child X. The same process is done for the second child. When the both childrenare valid, we check their fitness. If the child's fitness is better than the parent's fitness, we replace the worst parent with the child, otherwise the parent stays in the population. This process is done to keep the better individual in the population like in the elitism procedure. So, if the fitness is not good enough, we can say that the child cannot survive in the population or die in the next iteration.

3.1.4 Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation to the next. In our case, Mutation is done to the three out of four elite chromosomes in each iteration. Mutation is done using three procedures: swap, flip, and slide.All of this process (which gene will replace which gene) is done arbitrarily. We generate two random numbers that correspond to the j^{th} position in the sequence (j^{th} gene in the chromosome).

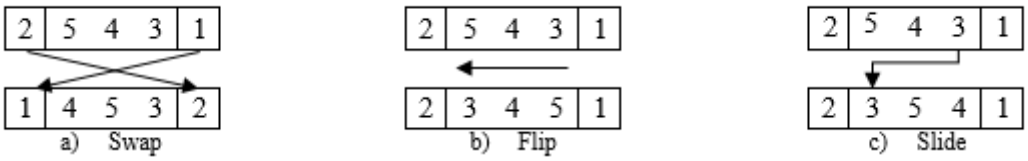

Figure 7Mutation process

Swap is done by swapping two genes. For example, we generate two random numbers corresponding to positions 1 and 5. Then we swap between product 2 and product 1 (figure 7.a). Flip or inverse is done by inverting the chosen gene sequence.In figure 7.b, we chose randomly position 4 (which is product 3 here) and position 2 (which is product 5), then we invert the sequence of 5-4-3 to 3-4-5.And slide is done by sliding the genes, for example, based on a random pick, we slide position 4 to position 2. Here we get product 3 (which is in position 4) and then we slide it to position 2 of the sequence. So, we change the sequence from 5-4-3 to become 3-5-4 (fig 7.c).

3.2 Simulated Annealing

Simulated annealing is founded by Kirkpatrick et al. [21]. This algorithm is based on the annealing process of steel. Simulated Annealing (SA) has been widely used for solving combinatorial problems such as VRP, TSP, scheduling and many more. Simulated Annealing is also known as a meta-heuristic that has the capability to get out of local optima: It has a schematic process to accept a worse solution to avoid being trapped in a local optimum.

In the literature review, there are not many researchers that have used the Simulated Annealing to solve the Mixed Model Sequencing Problem. The present work illustrates the application of SA on MMSP-W and provides some numerical tests on the performance of this type of problem. The pseudo code of simulated annealing that we use to solve the MMSP-W is shown in Figure 8.

```
Simulated Annealing for MMSP-W

Default Parameter
To  = Initial Temperature
c   = Cooling schedule (Parameter)

Generate Initial sequence
Calculating the objective
Initial Temperature

while Cycle < maximum Cycle

      while Iteration < maximum Iteration

      Generate new sequence
      Re-Calculate Objective

      if New Solution <= Old Solution;
           Solution  = New Solution
      elseif New Solution > Old Solution;
           Delta   = (New Solution - Old Solution);
           Probability = random Number;
           Acceptance = exp(-Delta/(k*T));

           if Probability <= Acceptance;
              Solution = New Solution;
           end
      end

      Iteration = Iteration + 1;
      end

      Temperature Update = T * c;
      cycle = cycle + 1;
end
```

Figure 8Pseudo code of SA

One advantage of the SA method is that it can accept a worse solution based on the acceptance probability, to avoid local optima. The acceptance probability is formulated as shown in eq. (10).

$$P(E) = e^{-E/kT} \quad \textbf{(10)}$$

Where,

$P(E)$ = Probability of acceptance

E = Difference between new solution and old solution

k = Boltzmann's Constant (k = *1*)

T = Temperature

e = Euler's number

Based on the pseudo code (Figure 8), we have two loops: inner loop (iteration) and outer loop (cycle). Outer loop (cycle) affect the probability of acceptance and inner loop (iteration) affect temperature degradation.The temperature update or temperature degradation happens after several iterations (inner loop), meaning that temperature isn't always decreasing at each iteration (inner loop). One disadvantage of the method is that the performance of the algorithm depends highly on the good choice of the parameters.

In this SA we do the neighbourhood search using flip. This procedure are done by flipping or inversing the sequence of selected position. It is the same procedure as explained in the mutation of GA (figure 7.b).

3.3 GASA

The idea is to benefit from the strengths of each meta-heuristic to improve the method. The initial population of the GA is randomly generated. If this initial population gives already good solutions, we may hope to reach a good solution faster. Using this assumption, we propose to inject the results of SA as initial population to GA. Concretely, we run the simulated annealing several times, and then we take the results of simulated annealing as the initial population for the Genetic Algorithm. We refer to it as GASA.

4 NUMERICAL TEST

The testsare done to evaluate the performance of Genetic Algorithm. They are divided into two sets of instances: Small Instances (Academic Instances) and Big Instances (Real Case from Industrial Data). Small instances are used to check whether the algorithm works correctly or not. Big instances are used to check the performance of the algorithm on real case. An instance is characterized by the number of workstations, the number of products to sequence and the process times of products on a given workstation. The big instances come from an industrial case study(Renault Truck's Bourg-en-Bress Plant). Tests for both instances are performed, using MATLAB, in Intel Core i5 – 2.4 GHz - 4 GB RAM. The results are compared to the results obtained by Mixed Integer Linear Program (MILP) developed in Aroui et al [11], based on the computational time and quality of the result (efficiency). Note that, MILP is solved using CPLEX on the same computer.

The small instances are composed of three sets of data for multi workstation as shown in Table 1 to Table 3. In these small instance sets, there are for 4 workstations and 20 products. Efficiency is calculated as in eq. (11).

$$Efficiency = \left(1 - \frac{Algorit\ hm\ Sloution\ - Optimal\ \ Solution}{Optimal\ \ Solution}\right) \times 100\% \qquad \textbf{(11)}$$

10 trials are run for each of the instance regardless the size of the instances. Tables 1 to 3 show the best run and the worst run among the 10 runs. We note that initial test runs are performed for the SA, to find the most appropriate parameter setting. Based on these tests, the c= 0.8 and Initial temperature = 10.000.

Table 1Genetic Algorithm results for 4 workstations 20 products

Problems	Linear Programming		Genetic Algorithm					
	Optimal Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
1	23.6	1373.00	23.8	23.6	20.25	6.40	99%	100%
2	16.0	136.00	17.5	16.0	23.62	5.69	91%	100%
3	5.5	311.00	5.6	5.5	39.58	5.44	98%	100%

Table 2 Simulated Annealing results for 4 workstations 20 products

Problems	Linear Programming		Simulated Annealing					
	Optimal Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
1	23.6	1373.00	23.6	23.6	2.48	2.46	100%	100%

Problems	Linear Programming		Simulated Annealing					
	Optimal Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
2	16.0	136.00	16.7	16.2	2.48	2.47	96%	99%
3	5.5	311.00	5.5	5.5	2.46	2.45	100%	100%

Table 3 GASA results for 4 workstations 20 products

Problems	Linear Programming		Genetic Algorithm with Simulated Annealing (GASA)					
	Optimal Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
1	23.6	1373.00	23.6	23.6	54.09	41.72	100%	100%
2	16.0	136.00	16.2	16.0	49.04	39.79	99%	100%
3	5.5	311.00	5.5	5.5	43.80	33.20	100%	100%

For small instances, Genetic Algorithm (GA), Simulated Annealing (SA) and combined Genetic Algorithm – Simulated Annealing (GASA) don't have much difference. In almost all cases, they can reach the optimal solution. But in terms of calculation speed, SA provides the fastest solutions (optimal solutions found in 3 seconds, with very low variability around this mean). The execution time performance of the MILP model, however, depends on the instance set.

For the big instances,we have 9 instances with 56 ~ 61 products and 77 workstations, each corresponding to a day's production.In those instances, GA, SA and GASA behave differently. Table 4 to Table 6 showresults of each algorithm.

Table 4 GAresult on industrial data

Problem Type	Linear Programming		Genetic Algorithm					
	Last Found Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
2013-07-09	673.0	3600.00	662.4	634.4	273.88	217.71	102%	106%
2013-07-10	445.1	3600.00	454.2	433.6	306.04	232.79	98%	103%
2013-07-11	597.2	3600.00	602.7	593.3	318.89	224.41	99%	101%
2013-08-13	720.0	3600.00	725.7	707.1	285.77	195.46	99%	102%
2013-08-14	1566.2	3600.00	1571.6	1451.1	329.46	249.31	100%	107%
2013-08-20	630.5	3600.00	627.7	617.7	351.68	245.90	100%	102%
2013-08-21	640.3	3600.00	661.7	623.6	336.14	249.52	97%	103%
2013-08-22	1248.2	3600.00	1268.9	1237.0	370.20	267.34	98%	101%
2013-08-23	632.4	3600.00	609.1	600.3	349.69	284.12	104%	105%

For these instances, GA gives very good results. Note that the MILP is stopped at the end of 1 hour and the best solution found is recorded on Table 4. The Genetic Algorithm can achieve solutions -3% ~ 7% better than the MILP solutions. The calculation time is also much faster than the MILP.

Table 5 SAresult big instances

Problem Type	Linear Programming		Simulated Annealing					
	Last Found Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
2013-07-09	673.0	3600.00	760.5	677.9	17.22	17.07	87%	99%
2013-07-10	445.1	3600.00	505.0	447.7	17.14	17.07	87%	99%
2013-07-11	597.2	3600.00	647.7	606.8	17.09	17.03	92%	98%
2013-08-13	720.0	3600.00	849.2	762.5	16.29	16.23	82%	94%
2013-08-14	1566.2	3600.00	1744.1	1567.5	17.35	17.26	89%	100%
2013-08-20	630.5	3600.00	704.8	634.6	17.45	17.40	88%	99%
2013-08-21	640.3	3600.00	749.7	640.9	17.48	17.38	83%	100%
2013-08-22	1248.2	3600.00	1446.4	1312.6	17.48	17.42	84%	95%
2013-08-23	632.4	3600.00	672.5	636.2	17.46	17.39	94%	99%

Here we can see that SA provides low execution times. We only need 17 seconds to execute SA. We get good results, even if this result isn't as good as MILP. On the contrary, GA needs more execution time to achieve the same quality result as MILP.

Table 6 GASA result big instances

Problem Type	Linear Programming		Genetic Algorithm with Simulated Annealing (GASA)					
	Last Found Solution	Calculation Time (Second)	Objectives Function		Calculation Time (Second)		Efficiency	
			Worst	Best	Longest	Shortest	Worst	Best
2013-07-09	673.0	3600.00	656.8	635.8	430.22	376.50	102%	106%
2013-07-10	445.1	3600.00	446.2	437.1	450.65	387.72	100%	102%
2013-07-11	597.2	3600.00	600.1	594.1	453.52	365.68	100%	101%
2013-08-13	720.0	3600.00	717.7	710.9	421.68	371.96	100%	101%
2013-08-14	1566.2	3600.00	1526.6	1460.7	471.57	381.66	103%	107%
2013-08-20	630.5	3600.00	625.6	618.7	443.82	376.97	101%	102%
2013-08-21	640.3	3600.00	629.4	623.0	486.59	401.36	102%	103%
2013-08-22	1248.2	3600.00	1263.9	1239.0	468.89	404.55	99%	101%
2013-08-23	632.4	3600.00	607.1	593.8	482.94	403.11	104%	106%

For the big instance sets,GASA's calculation time is faster (7-8 minutes) than the MILP model (1 hour of execution) and the quality of the solution issimilar to MILP. For the best runs, GASA improves solutions within the range of 1% to 7%.

We can conclude that, in terms of calculation speed, SA is the best method as can be seen in Table 4. SA only needs 17 seconds of execution.And, in terms of quality of solution, GA and GASA are the best method among the MILP and meta-heuristic methods.

5 CONCLUSION

We successfully applied several meta-heuristics for the resolution of the Mixed Model Assembly Line Sequencing Problem, in the case of delay minimization. The meta-heuristics achieved good quality solutions with faster computational time compared to the MILP model. The Genetic Algorithm gets the optimal solution in all small instances. For the big instance, GA gives very good results; in an average of 5 ~ 6 minutes it achieves solutions 0% ~ 7% better than the MILP solutions of 1 hour's calculation. The GA's calculation time is also much faster than the MILP. GA is more efficient than MILP for these instances. Despite good performance in small instances, SA is less performing for big instances than GA and MILP, in terms of quality of the obtained results but in less calculation times. GASA, on the big instances, has a calculation time between 7-8 minutes and the quality of the solution is similar to MILP. GASA even improves solutions within the range of 1% to 7% for the best run. GA or GASA are good procedures for industrial users. Both are sufficiently rapid and give better solution than manual solution

This research can still be improved on several aspects such as: type of operators, the methods, and the objectives. As reported by Aroui et al [10] in their further study, they reveal that there are other types of operators in the industrial case, for instance, operators working only on specific vehicles or operators that are working in tandem with other operators. The meta-heuristics can be extended to model such operators as well. In terms of method, SA can further be improved by modifying the neighbourhood search. Indeed, in this article, we only considered neighbours generated by a flip. We can also look at the Dynamic Programing as an alternative to meta-heuristic method. In terms of objectives, the single objective can be extended to multi objective case to take into account the industrial reality. For instance, minimizing delay and minimizing the part rate usage at the same time can be interesting to model.

6 REFERENCES

[1] **N. Boysen, M. Fliedner, and A. Scholl. 2009.**Sequencing mixed-model assembly lines: Survey, classification and model critique,*European Journalof Operation Research*, vol. 192, no. 2, pp. 349–373.

[2] **J. Bautista and J. Pereira. 2002.**Ant algorithms for assembly line balancing, *Ant Algorithms*, pp. 65–75.

[3] **N. Boysen, M. Kiel, and A. Scholl. 2011.**Sequencing mixed-model assembly lines to minimise the number of work overload situations,*Int. J. Prod. Res.*, vol. 49, no. 16, pp. 4735–4760.

[4] **M. Amen. 2000.**Heuristic methods for cost-oriented assembly line balancing: A survey,*Int. J. Prod. Econ.*, vol. 68, no. 1, pp. 1–14.

[5] **A. Chaves, L. Lorena, and C. Miralles. 2009.**Hybrid metaheuristic for the assembly line worker assignment and balancing problem, *Hybrid Metaheuristics*, pp. 1–14.

[6] **C. J. Hyun, Y. K. Kim, and Y. K. Kim. 1998.**A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines,*Comput. Oper. Res.*, vol. 25, no. 7–8, pp. 675–690.

[7] **S.G. Ponnambalam, P. Aravindan, and M. Subba Rao. Dec. 2003.**Genetic algorithms for sequencing problems in mixed model assembly lines, Comput. Ind. Eng., vol. 45, no. 4, pp. 669–690.

[8] **J. Cano-Belmán, R. Z. Ríos-Mercado, and J. Bautista. 2010.**A scatter search based hyper-heuristic for sequencing a mixed-model assembly line, *J. Heuristics*, vol. 16, no. 6, pp. 749–770.

[9] **A. R. Rahimi-Vahed, S. M. Mirghorbani, and M. Rabbani. 2007.**A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem, *Soft Comput.*, vol. 11, no. 10, pp. 997–1012.

[10] **K. Aroui, G. Alpan, and Y. Frein, 2014.**Minimizing work overload in mixed model assembly lines: A case study from truck industry, in *International Conference on Information Systems, Logistic, and Supply Chain*.

[11] **K. Aroui, G. Alpan, Y. Frein, and J. Thomazeau. 2013.**Minimisation des retards dans le séquencement des véhicules sur une ligne d'assemblage multi modèles, in *5èmes Journées Doctorales/Journées Nationales MACS*.

[12] **J. Bautista, R. Companys, and A. Corominas. 1996.**Heuristics and exact algorithms for solving the Monden problem, *Eur. J. Oper. Res.*, vol. 88, no. 1, pp. 101–113.

[13] **A. R. Rahimi-Vahed, M. Rabbani, R. Tavakkoli-Moghaddam, S. A. Torabi, and F. Jolai. 2007.**A multi-objective scatter search for a mixed-model assembly line sequencing problem,*Adv. Eng. Informatics*, vol. 21, no. 1, pp. 85–99.

[14] **S. M. Mirghorbani, M. Rabbani, R. Tavakkoli-Moghaddam, and A. R. Rahimi-Vahed.2007.**A multi-objective particle swarm for a mixed-model assembly line sequencing, in *Operations Research Proceedings 2006*, Springer, pp. 181–186.

[15] **Ş. Alpay. 2009.**GRASP with path relinking for a multiple objective sequencing problem for a mixed-model assembly line, *Int. J. Prod. Res.*, vol. 47, no. 21, pp. 6001–6017, Nov.

[16] **Q. Zhu and J. Zhang. 2011**. Ant colony optimisation with elitist ant for sequencing problem in a mixed model assembly line,*Int. J. Prod. Res.*, vol. 49, no. 15, pp. 4605–4626, Aug..

[17] **X. S. Yang**. 2010. Nature-Inspired Metaheuristic Algorithms, Second Edi. Luniver Press, p. 139.

[18] **Y. Kim, C. Hyun, and Y. Kim. 1996.**Sequencing in mixed model assembly lines: a genetic algorithm approach, *Comput. Oper. Res.*, vol. 23, no. 12, pp. 1131–1145.

[19] **Z. Ahmed. 2010.**Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, J. *Biometrics Bioinforma.*, no. 3, pp. 96–105.

[20] **B. Santosa and P. Willy**. 2011. Metoda Metaheuristik Konsep dan Implementasi, Surabaya, Guna Widya.

[21] **S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983.**Optimization by simulated annealing., *Science (80-.).*, vol. 220, no. 4598, pp. 671–80.