King Saud University

College of Engineering

Department of Industrial Engineering

Manufacturing Planning and Control
IE 516
Term Project:
Production Scheduling for NCMS machines in the mechanical workshop

Submitted to:

Dr. Ibrahim Alharkan

By:

MUHAMMAD DZULQARNAIN AL FIRDAUSI	438106660
NAWAF ABDULLAH ALSUWAYYIGH	441105567

10 December 2019

Abstract

In this project, a practical mechanical workshop in NCMS Company with five main sections will be studied and analyzed in order to make the optimum operation plan, near to optimum, or rather better than existing plan. A paper founded using localization (AL) and second one is an evolutionary approach controlled by the assignment model (generated by the first approach) to solve such a similar condition with multi-machines and stages.

In this project, we will apply the same approaches used except difference in the problem size. MATLAB codes were developed for non-preempted multi-stages flexible job shop scheduling problem for the proposed algorithms. The results are analyzed in terms of minimum Makespan which will reduce the time which is has a positive financially impact.

Table of Content

List of Tables

List of Figures

1. Introduction

Traditionally, assignment and scheduling decisions are made separately at different levels of the production management framework. The combining of such decisions presents additional complexity and new problems [1].

In this term project, we present an optimization approaches to a workshop in order to achieve minimum makespan comparing to the actual existing one and determining the part start processing time and finish in order to finish 1200 systems considering the parts flow and need for other workshop to final assemble the final system. the time needed for other workshop to complete the process on the mechanical part will be determined depending on historical data. Handling time is eliminated due to the similarity for all the products.

Based on the Imed Kacem and Slim Hammadi paper job shop problem is solved using two algorithms localization (AL) and second one is an evolutionary approach controlled by the assignment model (generated by the first approach). The aim in this project is to find an effective schedule with minimum makespan. The rest of the study is formed as follows: Section 2 literature review. Section 3 briefly describes the problem addressed in this project. Section 4 presents the algorithms used. Section and include briefly discussed the structure of the code algorithms. Finally, Section 5 and 6 are existing system analysis and application of method. Section 7 is conclusion and section 8 is critique to the base paper

2. Literature Review and the Base Paper

In Imed Kacem and Slim Hammadi paper, an efficient method is proposed to solve the assignment and job-shop scheduling problem (with partial or total flexibility to minimize the overall completion time (makespan) and the total workload of the machines. This multi-objective optimization will be done in a suitable search space that will be determined by a judicious assignment algorithm. The paper was published at 2002 by Imed Kacem and Slim Hammadi member of leading professional association for the advancement of technology "IEEE"

3. Problem Introduction and Description

The workshop consisting of five sections Cutting, DMG, Lathing, Milling, and Hand Work. Cutting section is responsible for supplying the aluminum bars and cut it in the size of the final products to feed the Lathing and Milling sections. Times analysis shows that the cutting section has no delayed recorded in supplying the small raw material bars to the other sections. DMG section is an independent high mass production machines for small parts consisting of the machines (DMG50 DMG20_1 DMG20_2). Most of the big critical parts are processed in the Lathing and Milling Machines. Lathing section consist of 6 machines TC-77A, TC-77B, TC-77C, TC-77D, Feeder, and TC110. The first four machines are set of exactly the same type of machines. Each of those type of machines is assigned to some products and in this projects we are eliminating the probability of operating the products on the other set with different processing time "cause each set has different capability". Milling department has one type setoff three machines U400-A, U400-B, and U400-C. some of the products has to be processed on both milling and lathing machines in a certain sequence and could pass twice to the lathing section on the same or different type of set machines. Since all jobs are available at time zero, the makespan of optimal schedule is independent with the processing sequence of batches. Then, the makespan of the problem is equal to the total processing time of all batches.

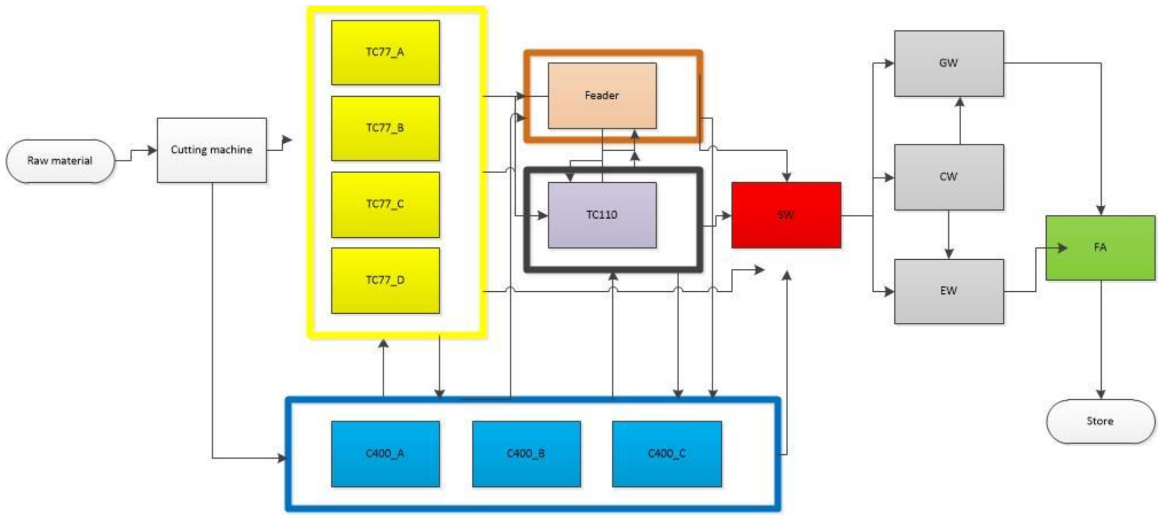


Fig. 1 Product Flow

The dispatching rule in our paper is based in the product weight. The product weight depends on the time needed to product delivery to the final assembly workshop. The problem under investigation can be described as follows:

3.1. Assumptions

Some of assumption and constraints is introduced in the projects as follows:

- All the jobs are ready at time zero
- One machine can process one product at a time
- Handling time is eliminated
- The study without the preventive maintenance scheduling
- The dispatching rule based on the product weight which is based on products flow time among other workshops reaching to the final assembly plus its processing time

- The total number of machine under study is 9 machines
- The sequence of the products: “We cannot do the stage 2 before stage 1”
- The product cannot be processed until the machine is free

3.2. Parameters

Jobshop scheduling problem (JSSP) is conducted where n-jobs

J_i= {J1, J2, J3, ………Jn}

Are processed on m-machines

M_k= {M1, M2, M3,……..Mm)

O_{ij} = operation for stage i and job j

$d_{i,j,k}$ = time for operating stage i for job j at machine k

3.3. Decision Variables

C max: The makespan, which is equal to the completion time of the last batch

4. Algorithms Description

This section as well as the next sections explains two types of algorithm used in this project, constructive type and improvement type. The constructive type starts with raw data and provides jobs assignment, sequencing, and scheduling. The improvement type is conceptually completely different from the constructive type. It starts out with a complete schedule from constructive type and then try to obtain a better schedule by manipulating the current schedule.

In this project, we refer to [1] with modifications by ourselves which proposed an efficient method to solve the assignment and job-shop scheduling problem with partial flexibility. The considered objective is to minimize the overall completion time (makespan). This objective optimization will be done in a suitable search space that will be determined by a judicious assignment algorithm. Computational experiments will be carried out to evaluate the efficiency of these methods with a large set of representative problem instances based on real industrial data. Analysis of the methods and their functioning will also be done.

4.1. Assignment Algorithm

The algorithm is based on the procedure described in Fig. 2. This procedure enables us to assign each operation to the suitable machine taking into account the processing times and workloads of machines on which we have already assigned operations. To explain this algorithm, we choose the following example (based on our data chunk).

The problem is to execute three jobs on nine machines according to the processing times possibilities $d_{i,j,k}$ $\{1 \leq j \leq N;\ 1 \leq i \leq n_j;\ 1 \leq k \leq M\}$ described in Table 1. Each Job has different number of nonpreemptable ordered operations, such as J1 has four operations. In our case, some tasks are only achievable on a part of the available machines set. In the example of Table 1, symbol "X" indicates that the assignment is impossible. According to [1], this constraint is going to make the problem more difficult, complicate the search space, and increase the computation time. However, we are going to show that such an assignment procedure is applicable in this case. In fact, the algorithm avoids to assign an operation to a machine whose processing time is long. Thus, for each forbidden assignment, we have associated an infinite fictitious processing time, which means we suppose that operation $O_{i0,j0}$ (correspondent to the forbidden assignment to machine M_{k0}) is henceforth achievable in an infinite time $d_{i0,j0,k0} = +\infty$. Thus, this assignment will be automatically rejected by the algorithm since it avoids long durations.

For this example, we only need to transform infinite processing time $d_{infinite} = 999$ for each forbidden state according to the equivalent data shown in Table 2. The application of the algorithm has confirmed this idea. In fact, no forbidden assignment has been presented by one of the solutions given by the Assignment Procedure. Such a result shows an equivalence in this type of problem. This equivalence is very interesting to implement a modular conception and construct supple solutions.

Table 1 Table D

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	5	5	6	6	6	6	X	X	X
	O2,1	85	80	100	100	100	100	X	X	X
	O3,1	X	X	X	X	X	X	60	60	60
	O4,1	20	20	25	25	25	25	X	X	X
J2	O1,2	40	45	55	55	55	55	X	X	X
	O2,2	50	55	70	70	70	70	X	X	X
	O3,2	X	X	X	X	X	X	35	35	35
	O4,2	24	20	30	30	30	30	X	X	X
J3	O1,3	17	13	20	20	20	20	X	X	X
	O2,3	50	55	70	70	70	70	X	X	X
	O3,3	X	X	X	X	X	X	35	35	35

Table 2 Table D

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	5	5	6	6	6	6	999	999	999
	O2,1	85	80	100	100	100	100	999	999	999
	O3,1	999	999	999	999	999	999	60	60	60
	O4,1	20	20	25	25	25	25	999	999	999
J2	O1,2	40	45	55	55	55	55	999	999	999
	O2,2	50	55	70	70	70	70	999	999	999
	O3,2	999	999	999	999	999	999	35	35	35
	O4,2	24	20	30	30	30	30	999	999	999
J3	O1,3	17	13	20	20	20	20	999	999	999
	O2,3	50	55	70	70	70	70	999	999	999
	O3,3	999	999	999	999	999	999	35	35	35

```
Starting from a table D presenting the processing time possibilities on the various machines, create a new table D' whose size is the same one as the table D;
Create a table S whose size is the same one as the table D (S is going to represent chosen assignments);
Initialize all elements of S to 0 ($S_{i,j,k} = 0$)
Recopy D in D';
FOR ($j=1; j \le N$)
  • FOR ($i=1; i \le n_j$)
  • Min = $+\infty$
  • r=RANDOM(M);
  • Position=r
  • FOR ($k=r, k \le M$)
      IF ($d'_{i,j,k} < Min$) Then {Min= $d'_{i,j,k}$; Position=k;}
      End IF
    End FOR
  • FOR ($k=1; k \le r-1$)
        IF ($d'_{i,j,k} < Min$) Then {Min= $d'_{i,j,k}$; Position=k;}
        End IF
    End FOR
  • $S_{i,j,Position}$ = 1: (assignment of $O_{i,j}$ to the machine $M_{Position}$)
    //Updating of D':
  • FOR ($i'=i+1; i' \le n_j$)
      $d'_{i',j,Position} = d'_{i',j,Position} + d'_{i,j,Position}$;
    End FOR
  • FOR ($j'=j+1; j' \le N$)
  •   FOR ($i'= 1; i' \le n_j$)
        $d'_{i',j',Position} = d'_{i',j',Position} + d'_{i,j,Position}$;
      End FOR
    End FOR
  End FOR
End FOR
```

Fig. 2 Assignment Algorithm

The application (step by step) of the assignment procedure yielded the following results.

- j = 1; i = 1: we assign $O_{1,1}$ to M_1 or M_2 (based on r=RANDOM(M)). Suppose the assignment is to M_1 therefore $S_{1,1,1}$ = 1 and we add $d_{1,1,1}$ = 5 to the elements of the first column of D' [the elements that follow the row (1, 1), see Table 2].
- j = 1; i = 2: we assign $O_{2,1}$ to M_2 ($S_{2,1,2}$ = 1) and we add $d_{2,1,2}$ = 80 to the elements of the second column of D' [the elements that follow the row (2, 1), see Table 3].

By following the same method, we obtain assignment S1 shown in Table 5. Notice that the result obtained depends on the jobs positions in the table of processing times (the order of jobs) and machines positions (on columns). For our case, the positions are based on the weight for each job obtained from previous calculations.

Table 3 Table D' for j = 1 and i = 1

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	5	5	6	6	6	6	999	999	999
	O2,1	90	80	100	100	100	100	999	999	999
	O3,1	1004	999	999	999	999	999	60	60	60
	O4,1	25	20	25	25	25	25	999	999	999
J2	O1,2	45	45	55	55	55	55	999	999	999
	O2,2	55	55	70	70	70	70	999	999	999
	O3,2	1004	999	999	999	999	999	35	35	35
	O4,2	29	20	30	30	30	30	999	999	999
J3	O1,3	22	13	20	20	20	20	999	999	999
	O2,3	55	55	70	70	70	70	999	999	999
	O3,3	1004	999	999	999	999	999	35	35	35

Table 4 Table D' for j = 1 and i = 2

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	5	5	6	6	6	6	999	999	999
	O2,1	90	80	100	100	100	100	999	999	999
	O3,1	1004	1079	999	999	999	999	60	60	60
	O4,1	25	100	25	25	25	25	999	999	999
J2	O1,2	45	125	55	55	55	55	999	999	999
	O2,2	55	135	70	70	70	70	999	999	999
	O3,2	1004	1079	999	999	999	999	35	35	35
	O4,2	29	100	30	30	30	30	999	999	999
J3	O1,3	22	93	20	20	20	20	999	999	999
	O2,3	55	135	70	70	70	70	999	999	999
	O3,3	1004	1079	999	999	999	999	35	35	35

Table 5 Assignment S1

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	1	0	0	0	0	0	0	0	0
	O2,1	0	1	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	1
	O4,1	0	0	0	0	0	1	0	0	0
J2	O1,2	1	0	0	0	0	0	0	0	0
	O2,2	0	0	0	0	1	0	0	0	0
	O3,2	0	0	0	0	0	0	1	0	0
	O4,2	0	0	1	0	0	0	0	0	0
J3	O1,3	0	0	0	1	0	0	0	0	0
	O2,3	0	0	0	1	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	1	0

We use Matlab to apply the Assignment Algorithm and the code is in Appendix 1.

The randomness in Assignment Algorithm provides several possibilities in job assignments to machines. Considering the distribution of our data and the similarity of processing time for some machines, we decide to generate 100 assignments and compare the makespan of each assignment with the existing makespan in the industry. The calculation is done by Scheduling Algorithm in the next section. A schedule which has smaller makespan than existing schedule will be used to construct Assignment Schemata for controlling the Genetic Algorithm in order to obtain a better makespan.

4.2. Scheduling Algorithm

The objective of the Assignment Algorithm is to reduce the search space to an area where the probabilities of the balancing of machines workloads and the minimization of the makespan are raised. To test the efficiency of this algorithm, it is therefore necessary to check the regularity of machines workloads and to see the impact of the choice of the assignments on the makespan value.

This test has been made by applying a Scheduling Algorithm after choosing the assignments. This algorithm calculates starting times $t_{i,j}$ by taking account of machines availabilities and precedence constraints. Conflicts are solved by a heuristic using Shortest Processing Time (SPT) rules see Fig. 3.

```
Beginning Scheduling Algorithm
Initialize the vector of machines availabilities Dispo_Machine[k]=0 for each machine $M_k$ ($k \leq M$);
Initialize the vector of jobs availabilities Dispo_Job[j]=0 for each job j ($j \leq N$);
FOR ($i=1$, $i \leq Max_j (n_j)$)
    • Construct the set $E_i$ of operations to schedule from S:
      $E_i = \{O_{i,j} / S_{i,j,k} = 1, 1 \leq j \leq N\}$;
    • Classify the operations of $E_i$ according to the SPT;
    • FOR ($j=1$; $1 \leq j \leq N$)
        • Calculate starting times by following the same order given by the classification of $E_i$ according to the formula:
          $t_{i,j}$ = Max(Dispo_Machine[k], Dispo_Job[j]) such that $S_{i,j,k} = 1$;
        • Updating of the vector of machine availabilities:
          Dispo_Machine[k]= $t_{i,j} + d_{i,j,k}$;
        • Updating of the vector of job availabilities:
          Dispo_Job[j]= $t_{i,j} + d_{i,j,k}$;
      End FOR
End FOR
End Scheduling Algorithm
```

Fig. 3 Scheduling Algorithm

In this following, we explain this scheduling algorithm, we then evaluate our results with an example of the literature to conclude on the assignments efficiency.

Example: We consider the example presented in Table I and we choose assignment S1 (already presented in Table 5). The application of "scheduling algorithm" using short processing time (SPT) rule yielded the following results.

- Iteration 1: $i = 1$:

  Construction of $E_1 = \{O_{1,1}; O_{1,2}; O_{1,3}\}$: the operations of E_1 are respectively achievable during $d_{1,1,1} = 5$, $d_{1,2,1} = 40$ and $d_{1,3,4} = 20$ units of time. E_1 becomes therefore: $\{O_{1,1}; O_{1,3}; O_{1,2}\}$, in fact, $d_{1,1,1} \leq d_{1,3,4} \leq d_{1,2,1}$. The starting times are computed by following the same order what gives: $t_{1,1} = t_{1,3} = t_{1,2} = 0$. The updating of machines and job availabilities gives the following vectors:

  Dispo_Machines: (45, 0, 0, 20, 0, 0, 0, 0, 0); Dispo_Jobs: (5, 45, 20).

- Iteration 2: $i = 2$:

  Construction of $E_2 = \{O_{2,1}; O_{2,2}; O_{2,3}\}$: the operations of E_2 are respectively achievable during $d_{2,1,2} = 80$, $d_{2,2,5} = 70$ and $d_{2,3,4} = 70$ units of time. E_2 becomes therefore: $\{O_{2,2}; O_{2,3}; O_{2,1}\}$, in fact, $d_{2,2,5} \leq d_{2,3,4} \leq d_{2,1,2}$. The starting times are computed by following the same order what gives: $t_{2,2} = 45$; $t_{2,3} = 20$; $t_{2,1} = 5$. The updating of machines and job availabilities gives the following vectors:

Dispo_Machines: (45, 85, 0, 90, 115, 0, 0, 0, 0); Dispo_Jobs: (85, 115, 90).

- Iteration 3: $i = 3$:

  Construction of $E_3 = \{O_{3,1}; O_{3,2}; O_{3,3}\}$: the operations of E_3 are respectively achievable during $d_{3,1,9} = 60$, $d_{3,2,7} = 35$ and $d_{3,3,8} = 35$ units of time. E_3 becomes therefore: $\{O_{3,2}; O_{3,3}; O_{3,1}\}$, in fact, $d_{3,2,7} \leq d_{3,3,8} \leq d_{3,1,9}$. The starting times are computed by following the same order what gives: $t_{3,2} = 115$; $t_{3,3} = 90$; $t_{3,1} = 85$. The updating of machines and job availabilities gives the following vectors:

  Dispo_Machines: (45, 85, 0, 90, 115, 0, 150, 125, 145); Dispo_Jobs: (145, 150, 125).

- Iteration 4: $i = 4$:

  Construction of $E_3 = \{O_{4,1}; O_{4,2}\}$: the operations of E_4 are respectively achievable during $d_{4,1,6} = 25$ and $d_{4,2,3} = 30$ units of time therefore they keep the same order ($d_{4,1,6} \leq d_{4,2,3}$). The starting times are computed by following the same order what gives: $t_{4,1} = 145$; $t_{4,2} = 150$. The updating of machines and job availabilities gives the following vectors:

  Dispo_Machines: (45, 85, 180, 90, 115, 170, 150, 125, 145); Dispo_Jobs: (170, 180, 125).

  Finally, the schedule is shown in Table 6 using the following representation:

  [Machine, Starting time, completion time]

Table 6 A schedule given by AL (approach by localization)

	Ope_1	Ope_2	Ope_3	Ope_4
J1	1, 0, 5	2, 5, 85	9, 85, 145	6, 145, 170
J2	1, 5, 45	5, 45, 115	7, 115, 150	3, 150, 180
J3	4, 0, 20	4, 20, 90	8, 90, 125	■

The makespan (C_{max}) for this schedule is 180 with J2 as the last job processed in machine 3. The large advantage of AL is a significant reduction of the computation time. Matlab code for Scheduling Algorithm is in Apendix 2. In fact, the assignment algorithm localizes most of the interesting zones of the search space. Thus, the scheduling is increasingly easy and becomes more efficient.

In general, the solutions of the previously mentioned approach are equally acceptable and satisfactory. However, the solutions of many real problems are not necessarily of the same value in the eyes of decision makers and optimum or high-quality solutions according to the desired criterion are preferred. Therefore, it is worthwhile to investigate possible gains from hybridizing the AL with the GA which have been used to produce appropriate solutions for many problems while they do not guarantee the optimality of the final solution.

4.3. The Approach by Localization (AL) and Controlled Genetic Algorithm (CGA)

In this section, we show how the AL can contribute to minimize makespan by combining it with GAs.

4.3.1. Genetic Algorithms (GAs)

GAs enable us to make an initial set of solutions evolve to a final set of solutions bringing a global improvement according to a criterion fixed at the beginning [2]. These algorithms function with the same usual genetic mechanisms (crossover, mutation, and selection). Here, we explain the important concepts of GAs. The solutions set is called population. Each population is constituted of chromosomes which each represent a particular coding of a solution. The chromosome consists of a sequence of genes that can take some values called *alleles*. These values are taken from an alphabet that has to be judiciously chosen to suit the studied problem.

The classic coding corresponds to the binary alphabet: {0, 1}. In this case, the chromosome represents simply a table of 0 and 1. The operators that intervene in the GAs are selection, crossover, and mutation. The implementation difficulty of these algorithms consists of conceiving the genes content in order to describe all data of the problem. Concerning evolutionary algorithms and flexible job-shop scheduling problems, the literature presents many interesting propositions. Some of them can be used to solve the considered optimization problem.

A GA is an algorithm that represents a special architecture. It operates on data without using preliminary knowledge on the problem processed. In fact, it consists of the following stages:

1) Genesis: This is the generation phase of the initial population.
2) Evaluation: In this stage, we compute the value of criterion for each individual of the current population.
3) Selection: After the evaluation, we choose better adapted elements for the reproduction phase.

4) Reproduction: We apply genetic operators (crossover and mutation) on the selected individuals.
5) Test: In this phase, we evaluate the improvement and decide if the solution is efficient. If the criterion reaches a satisfactory value, we take the current solution. If the result is insufficient, we return to the second phase and we repeat the same process until reaching the maximal iterations number.

4.3.2. The Schemata Theorem

This notion was introduced in GAs by Holland [1]. It consists of conceiving a model of chromosomes that suits the problem. This model will serve in the construction of new individuals in order to integrate the good properties contained in the schemata [3].

Example: In the case of a binary coding, a schemata is a chromosome model where some genes are fixed and the others are free (see the following example *S*):

$$S = 100{*}1{*}00$$

Positions 4 and 6 are occupied by the symbol: “*”. This symbol indicates that considered genes can take “0” or “1” as value. Thus, chromosome C1 and C2 respect the model imposed by the schemata *S*

C1 = 10 001 100

C2 = 10 011 100

The objective of the schemata theory is to make GAs more efficient and more rapid in constructing the solution by giving priority to the reproduction of individuals respecting model generated by the schemata and not from the whole set of chromosomes.

In the case of scheduling problems, the difficulty of the implementation of this technique is higher. In fact, it necessitates the elaboration of a well particular coding that enables us to describe the problem data and exploit the schemata theory.

Here, we show how the AL enables us to overcome this difficulty and we introduce this notion to solve flexible job-shop scheduling problems.

We are reminded that the AL enables us to construct a set *E* of assignments by minimizing the sum of machines workloads. The idea is to generate, from the set *E*, an assignment schemata that will serve us to control the GA. This schemata is therefore going to represent a constraint which newly created individuals must respect. Thus, it would enable us to optimize makespan in a search area where assignments minimize the workloads of the machines (optimization by phase).

The construction of this schemata consists of collecting the assignments $S^z(1 \leq z \leq$ cardinal$(E))$ cardinal given by the Assignment Procedure and to determine (for each operation) the set of possible machines in S^{ch} according to the algorithm shown in Fig. 4.

```
Beginning Schemata Generation
Initialize $S^{ch}$ to 0 ($S^{ch}_{i,j,k} = 0$ for all i, j, k);
Define thresholds $\alpha$ and $\beta$;
• FOR (z=1, $z \leq$ cardinal(E))
    FOR (j=1, $j \leq N$)
      FOR (i=1, $i \leq n_j$)
        FOR (k=1, $k \leq M$)
        $S^{ch}_{i,j,k} = S^{ch}_{i,j,k} + (S^z_{i,j,k}$ / cardinal(E));
        End FOR
      End FOR
    End FOR
  End FOR
• FOR (j=1, $j \leq N$)
      FOR (i=1, $i \leq n_j$)
        FOR (k=1, $k \leq M$)
          IF ($S^{ch}_{i,j,k} < \alpha$) Then $S^{ch}_{i,j,k} = 0$;
          ELSE
          IF ($\alpha < S^{ch}_{i,j,k} < \beta$) Then $S^{ch}_{i,j,k} = *$;
          ELSE $S^{ch}_{i,j,k} = 1$;
          End IF
          End IF
        End FOR
      End FOR
  End FOR
  End Schemata Generation
```

Fig. 4 Schemata Generation Algorithm

For each operation $O_{i,j}$, this algorithm associates the frequency $S^{ch}_{i,j,k}$ to be assigned to a machine M_k then, in function of chosen thresholds α and β, it reduces the set $U_{i,j}$ to a subset where the probabilities of having a good schedule are raised.

As an example, for the problem introduced in Table 1, we obtain the schemata shown in Table 7 (for $\alpha = 0.03$, $\beta = 0.95$ and cardinal(E) = 20). The value $S^{ch}_{i,j,k} = 0$ indicates that the assignment of the operation $O_{i,j}$ to the machine M_k is *forbidden*. The value $S^{ch}_{i,j,k} = 1$ indicates that the assignment of the operation $O_{i,j}$ to the machine M_k is obligatory, in this case, all values of the rest of the row (i,j) are inevitably equal to "0." The symbol "*" indicates that the assignment is possible, in this case, we cannot have the value "1" in all the rest of the row (i,j). In conclusion, this schemata covers the totality of the interesting assignment possibilities and expensive prohibitions in terms of machine workloads. The Matlab code for Schemata Generation Algorithm is in Appendix 3

Table 7 Assignment Schemata S^{ch}

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	*	*	0	0	0	0	0	0	0
	O2,1	*	*	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	*	*	*
	O4,1	*	0	*	*	*	*	0	0	0
J2	O1,2	*	*	*	0	*	*	0	0	0
	O2,2	*	0	*	*	*	*	0	0	0
	O3,2	0	0	0	0	0	0	*	*	*
	O4,2	0	0	*	*	*	*	0	0	0
J3	O1,3	0	0	*	*	*	*	0	0	0
	O2,3	*	0	*	*	*	*	0	0	0
	O3,3	0	0	0	0	0	0	*	*	*

4.3.3. New Coding: Operations Machines Coding (OMC)

4.3.3.1. Coding

It consists in representing the schedule in the same assignment table *S*. We replace each case $S_{i,j,k} = 1$ by the couple $(t_{i,j}, tf_{i,j})$ where $t_{i,j}$ is the starting time and $tf_{i,j}$ is the completion time. The cases $S_{i,j,k} = 0$ are unchanged.

This new coding presents several advantages. On the one hand, it integrates the notion of the assignment schemata that represents the "skeleton" of an optimized scheduling. On the other hand, it enables us to exchange information contained in current good solutions and make fine crossovers. Also, this coding presents an easy form to interpret. In fact, by looking at the rows, we observe the execution of the operations and by looking at the columns, we get the tasks of each machine with the starting and completion times.

4.3.3.2. Crossover

This operator is described in Fig. 5 and illustrated by the following example (see Tables 8–13):

- Construction of offsprings (copying of the assignments, see Tables 10 and 11).
- Computation of starting and completion times (See Tables 12 and 13).

The Matlab code for Crossover is in Appendix 4.

```
Select randomly 2 parents $S^1$ and $S^2$.
Select randomly 2 integers j and j' such that j ≤ j' ≤ N;
Select randomly 2 integers i and i' such that i ≤ $n_j$ and i' ≤ $n_{j'}$
(in the case where j=j', i ≤ i');
The individual $e^1$ receives the same assignments from the parent $S^1$ for all operations between the row (i,j) and the (i',j');
The rest of assignments for $e^1$ is obtained from $S^2$;
The individual $e^2$ receives the same assignments from the parent $S^2$ for all operations between the row (i,j) and the (i',j');
The rest of assignments for $e^2$ is obtained from $S^1$;
Calculate the starting and completion times according to the Scheduling Algorithm
```

Fig. 5 Crossover Algorithm

Table 8 Parent S^1

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0, 5	0	0	0	0	0	0	0	0
	O2,1	0	5, 85	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	85, 145
	O4,1	0	0	0	0	0	145, 170	0	0	0
J2	O1,2	5, 45	0	0	0	0	0	0	0	0
	O2,2	0	0	0	0	45, 115	0	0	0	0
	O3,2	0	0	0	0	0	0	115, 150	0	0
	O4,2	0	0	150, 180	0	0	0	0	0	0
J3	O1,3	0	0	0	0, 20	0	0	0	0	0
	O2,3	0	0	0	20, 90	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	90, 125	0

Table 9 Parent S^2

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0	0, 5	0	0	0	0	0	0	0
	O2,1	0	5, 85	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	85, 145	0	0
	O4,1	145, 165	0	0	0	0	0	0	0	0
J2	O1,2	0	0	0	0	0, 55	0	0	0	0
	O2,2	55, 105	0	0	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	0	105, 140	0
	O4,2	0	0	0	140, 170	0	0	0	0	0
J3	O1,3	0	0	0	0	0	0, 20	0	0	0
	O2,3	0	0	20, 90	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	90, 125

Table 10 e^1 in construction

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0	?, ?	0	0	0	0	0	0	0
	O2,1	0	?, ?	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	?, ?
	O4,1	0	0	0	0	0	?, ?	0	0	0
J2	O1,2	?, ?	0	0	0	0	0	0	0	0
	O2,2	0	0	0	0	?, ?	0	0	0	0
	O3,2	0	0	0	0	0	0	?, ?	0	0
	O4,2	0	0	0	?, ?	0	0	0	0	0
J3	O1,3	0	0	0	0	0	?, ?	0	0	0
	O2,3	0	0	?, ?	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	?, ?

Table 11 e^2 in construction

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	?, ?	0	0	0	0	0	0	0	0
	O2,1	0	?, ?	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	?, ?	0	0
	O4,1	?, ?	0	0	0	0	0	0	0	0
J2	O1,2	0	0	0	0	?, ?	0	0	0	0
	O2,2	?, ?	0	0	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	0	?, ?	0
	O4,2	0	0	?, ?	0	0	0	0	0	0
J3	O1,3	0	0	0	?, ?	0	0	0	0	0
	O2,3	0	0	0	?, ?	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	?, ?	0

Table 12 e^1 first offspring

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0	0, 5	0	0	0	0	0	0	0
	O2,1	0	5, 85	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	85, 145
	O4,1	0	0	0	0	0	145, 170	0	0	0
J2	O1,2	0, 40	0	0	0	0	0	0	0	0
	O2,2	0	0	0	0	40, 110	0	0	0	0
	O3,2	0	0	0	0	0	0	110, 145	0	0
	O4,2	0	0	0	145, 175	0	0	0	0	0
J3	O1,3	0	0	0	0	0	0, 20	0	0	0
	O2,3	0	0	20, 90	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	90, 125

Table 13 e^2 second offspring

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0, 5	0	0	0	0	0	0	0	0
	O2,1	0	5, 85	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	85, 145	0	0
	O4,1	145, 165	0	0	0	0	0	0	0	0
J2	O1,2	0	0	0	0	0, 55	0	0	0	0
	O2,2	5, 55	0	0	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	0	55, 90	0
	O4,2	0	0	90, 120	0	0	0	0	0	0
J3	O1,3	0	0	0	0, 20	0	0	0	0	0
	O2,3	0	0	0	20, 90	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	90, 125	0

4.3.3.3. Mutation

The objective of our search is to minimize the makespan. It would therefore be interesting to make genetic operators able to contribute in this optimization. The operator of artificial mutation used here is operator of mutation reducing the effective processing time (EPT_j) of a job *j* (Fig. 6)

```
Select randomly an individual S;
Choose the job j whose Effective Processing Time is the longest:
        (Max_j {EPT_j such that $EPT_j = \sum_i \sum_k S_{i,j,k} . d_{i,j,k}$});
i=1; r=0;
WHILE ($i \le n_j$ and r=0)
    • Find $K_0$ such that $S_{i,j,K0} = 1$;
    • FOR (k=1, $k \le M$)
          IF ($d_{i,j,k} < d_{i,j,k0}$) Then { $S_{i,j,K0} = 0$; $S_{i,j,K} = 1$; r=1}
          End IF
      End FOR
    • i=i+1;
End WHILE
Calculate starting and completion times according to the Scheduling Algorithm
```

Fig. 6 Mutation Algorithm

Let us consider the example S shown in Table 15. The job 2 has the most raised value of the EPT's (EPT_2 = 180 units of time). We therefore have to cover the list of its operations to reduce this duration. For operation $O_{1,2}$ the processing time $d_{1,2,1}$ corresponds to the minimum of processing times. On the other hand, operation $O_{2,2}$ can be assigned to the machine M_1 instead of the machine M_5 (because $d_{2,2,1} < d_{2,2,5}$) and thereafter we reduce the EPT_2 to 20 units of time and the makespan to 170 units instead of 180. The obtained chromosome is shown in Table 15. The Matlab code for Mutation is in Appendix 5.

Table 14 S before mutation

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0, 5	0	0	0	0	0	0	0	0
	O2,1	0	5, 85	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	85, 145
	O4,1	0	0	0	0	0	145, 170	0	0	0
J2	O1,2	5, 45	0	0	0	0	0	0	0	0
	O2,2	0	0	0	0	45, 115	0	0	0	0
	O3,2	0	0	0	0	0	0	115, 150	0	0
	O4,2	0	0	150, 180	0	0	0	0	0	0
J3	O1,3	0	0	0	0, 20	0	0	0	0	0
	O2,3	0	0	0	20, 90	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	90, 125	0

Table 15 S after mutation

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0, 5	0	0	0	0	0	0	0	0
	O2,1	0	5, 85	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	85, 145
	O4,1	0	0	0	0	0	145, 170	0	0	0
J2	O1,2	5, 45	0	0	0	0	0	0	0	0
	O2,2	45, 95	0	0	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	95, 130	0	0
	O4,2	0	0	130, 160	0	0	0	0	0	0
J3	O1,3	0	0	0	0, 20	0	0	0	0	0
	O2,3	0	0	0	20, 90	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	90, 125	0

4.3.4. Controlled Genetic Algorithm (CGA)

The objective considered is to imbricate a double control in the evolutionary approach (see the flowchart described in Fig. 7):

- A control ensured by the schemata assignment (obtained by applying the AL);
- A control ensured by the artificial mutations (genetic manipulations).

We can notice that the crossover operator preserves the membership of the new individuals to the assignment schemata. Therefore, the GA will be only controlled by testing new individuals after their mutation in order to reduce the computation time and make the search more efficient.

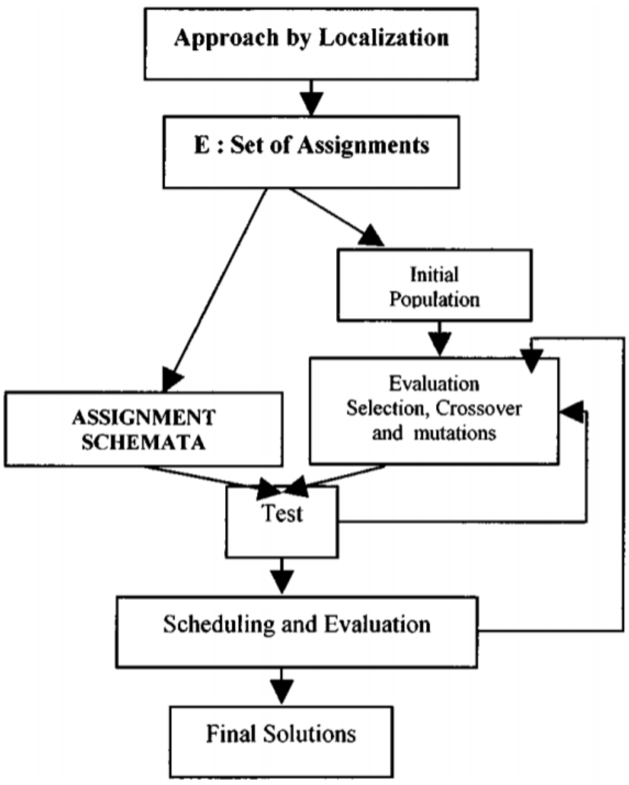


Fig. 7 Controlled genetic algorithm.

5. Existing Assignment and Schedule

We collect from the industry their existing assignment and schedule for 20 products in consideration. To model the performance of existing system, we apply Scheduling Algorithm to their existing assignment based on aforementioned assumptions we use in this study. The value of makespan which is small is more preferable and thus we use the existing makespan as upper limit for generating schedule with Assignment Algorithm and Scheduling Algorithm. Appendix 6 shows the assignments for each operations of each job to respective machines and the application of Scheduling Algorithm to the data results:

$$\boldsymbol{C_{max}} \textbf{ existing = 295}$$

The existing C_{max} is usable to eliminate schedules generated by Assignment Algorithm in advance before the schedules being used to construct Assignment Schemata (S^{ch}) hence we reduce the search space for obtaining faster a better makespan. Depiction of this elimination as follows:

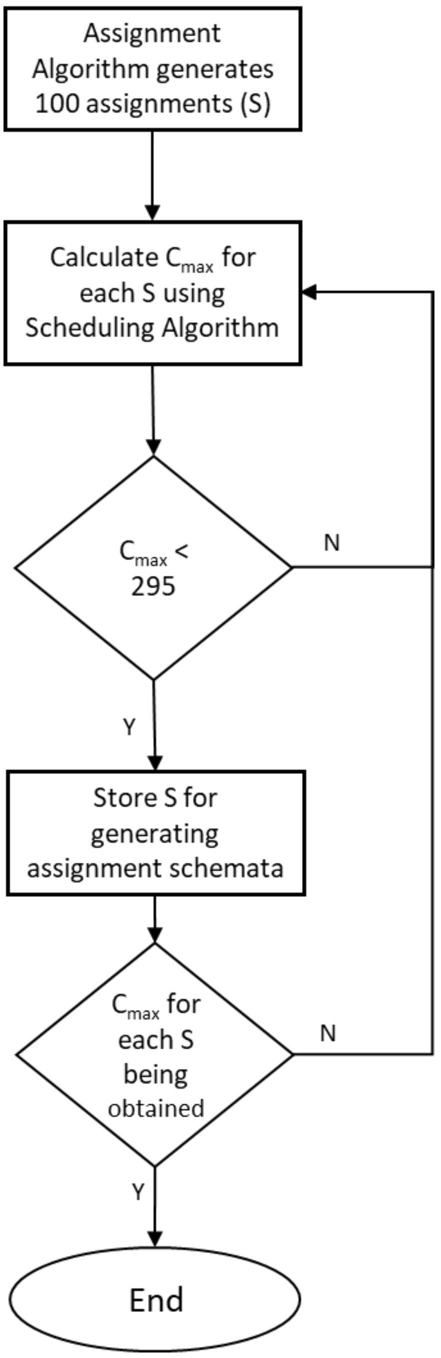

Assignment Algorithm generates 100 assignments (S)
Calculate C_{max} for each S using Scheduling Algorithm
C_{max} < 295
N
Y
Store S for generating assignment schemata
C_{max} for each S being obtained
N
Y
End

Fig. 8 Schedules Elimination

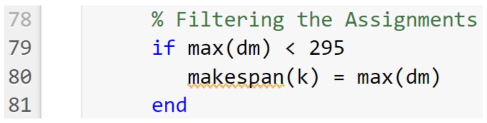
6. Application of Controlled Genetic Algorithm (CGA)

Fig. 6 provides step-by-step application of CGA and each step has been explained clearly in previous sections. The processing time data obtained from the industry is in appendix 7. Then, we have applied the controlled genetic algorithm (CGA) with the following parameters:

- Population size: N_{ind} = cardinal (E) = 100
- Crossover probability: P_c = 0.88
- Mutation probability: P_m = 0.12

6.1. Approach by Localization (AL)

AL contains of Assignment procedure and Scheduling Algorithm. Appendix 7 shows the processing time of each operation $O_{i,j}$ in machine M_k. In this study, total of 52 operations are considered to be assigned to machines. Some machines have same processing time and based on this we consider to generate 100 assignments (population size) with Assignment Procedure shown in appendix 1. We filter the population by comparing their makespan with existing makespan, the calculation of makespan is using Scheduling algorithm shown in Appendix 2. Condition for filtering the population is shown in following figure.



```
    % Filtering the Assignments
    if max(dm) < 295
        makespan(k) = max(dm)
    end
```

Fig. 9 Condition for filtering the population

Scheduling algorithm provides us with vector of machines availabilities Dispo_Machine[k] (we represent it by “dm” in Matlab) informing the completion time of each machine in each stage of Job-Shop scheduling. It is easy to find the makespan of an assignment by taking the maximum value of “dm”. Then, each individual (assignment) which has smaller makespan than existing will serve as chromosome for CGA and then will be manipulated (based on P_m and P_c) by CGA operators. An example of chromosome, S^1, is provided in Appendix 8, with the makespan of 249. After filtering, we have total 65 chromosomes with the minimum value of makespan by 235 (an instance of the assignment with this value is in Appendix 9). The result shows that there are 8 chromosomes having this minimum value with different assignments. We decide to use all the 65 chromosomes for input of CGA.

6.2. Assignment Schemata (S^{ch})

The idea of generating assignment schemata is to keep the feasibility constraint fulfilled when applying operators in CGA (no operation assigned to *forbidden* machine) and make the process of CGA faster (no need for calculation of makespan on an infeasible schedule). Fig. 3 shows the simple calculation for generating Assignment schemata and the corresponding Matlab code is in Appendix 3. Referring to our base paper, we use $\alpha = 0.03$ and $\beta = 0.95$ and get result as in Appendix 10. We recognize from our assignment schemata that there is no *obligatory* machine assignment due to large size of population. Therefore, this S^{ch} will validate a chromosome as a valid or not valid schedule.

6.3. Genetic Algorithm Operators

In brief, GA operators are consisting of Evaluation Selection, Crossover, and Mutations. In fact, we have done the Evaluation Selection by filtering the population size of 100 with the value of existing makespan. Therefore, we will show the application of Crossover and Mutations on the selected chromosome.

6.4.1. Crossover on the Chromosomes

Code in Appendix 4 is used to apply the algorithm in Fig. 4. In brief, this code allows us to construct a set of chromosomes with makespan is less than 295. The number of elements in the set is determined by P_c ("coprob" in Matlab) and all the elements will be undergoing the Crossover. Every time a child is produced, we calculate the makespan with Scheduling algorithm and hold that child if the value is smaller than 235. Selecting among 57 chromosomes for being parents and producing childs, the Crossover can **reduce the makespan becoming 204** with assignment in Appendix 11.

6.4.2. Mutation on the Chromosomes

The assignment in Appendix 11 is then applied by Mutation Algorithm to see if we can obtain a better makespan. The code in Appendix 5 is applied for mutation and then we calculate the makespan after mutation. The assignment after mutation is in Appendix 12 and **we obtain the same makespan of 204 with different assignment**. Therefore, we conclude that in this case, the mutation can give alternative assignment with same small makespan.

7. Conclusion

In this study, we have applied an efficient methodology for flexible job-shop scheduling problems. This method enables us to construct solutions with good quality in a reasonable computation limit. Our first step was to apply the AL to solve the resource allocation problem and generate the assignment schemata. Our second step was to apply a controlled evolutionary algorithm. The initial population is constructed starting from the set of assignments found in the first stage. The evolution of generations will be controlled in the two following levels.

— First level: we test if the new individual respects the model imposed by the assignment schemata.

— Second level: we apply artificial mutations (genetic manipulations) in order to reduce the blind aspect of genetic operators.

As research continues into CGA and AL, the idea of applying the schemata theory becomes more important. In fact, this idea will be effective for a wide range of problem instances for which it was designed. Also, it will amenable to the modification or extension to solve others different types of problems. The fundamental building scheme used in this paper represents the problem at the correct level of abstraction ranging from a completely specified point in the problem types or a set of aggregate qualities that describe a family of a possible good solutions.

After we apply the CGA, it is obvious that we obtain a better assignment with smaller makespan than the existing method in the industry. The makespan is reduced from 295 to 204 which the optimization in makespan is about 69%.

8. Critiques on the paper

Some critiques for the base paper we referred to:

a. The reason for selecting GA parameters and threshold values for schemata theorem are not so clear.
b. They applied the GA operators on all chromosome without considering the makespan of one assignment that has been already smaller than before. It makes the computation time a little bit longer.
c. Synthetic data used in the paper did not represent a real case problem. In order to apply the method, we need several adjustments because the nature of our data is not normally distributed.

References

[1] I. Kacem, S. Hammadi and P. Borne, "Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS,* pp. 1-13, 2002.

[2] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, New York: Springer-Verlag, 1992.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, MA: Addison-Wesley, 1989.

Appendix

Appendix 1 Matlab code for Assignment Algorithm

Assignment Procedure

```
% generate 100 assignments
for j = 1:100
    pijk_new=pijk
    minvalue = 0
% start iteration from 1 to num of machines
    for i=1:rowpijk
        % find column & row of minimum value
        imin=find(pijk_new(i,:)==min(pijk_new(i,:)))
        selectidx = imin(randi(size(imin,2), 1, 1))
        minvalue=min(pijk_new(i,:))
        % store row with minimum value
        barismin = zeros(1,colpijk)
        barismin(selectidx)=1
        % add elements in minimum column with minimum value
        pijk_new(:,selectidx)=pijk_new(:,selectidx)+minvalue
        % construct table of assignment
        S(i,:) = barismin
    end
    Sz(:,:,j) = S
end
```

Appendix 2 Matlab code for Scheduling Algorithm

Scheduling Algorithm

```
for k=1:size(Sz,3)
    X=O
    [rowX colX] = size(X);
    dm = zeros(1,colpijk)
    dj = zeros(1,colE)
    for i=1:max(O)
        d = 0
        dmk = 0
        djj = 0
        E= zeros(20,9)
        S= zeros(20,9)
        for j=1:colE
            if O(j) >= i
                row=i+sum(O(1:(j-1)))
                E(j,:)=pijk(row,:)
                S(j,:)=Sz(row,:,k);
                [maxS, idxS] = max(S(j,:))
                d(j)=E(j,idxS)
                dmk(j) = idxS
            end
        end
        % · classify the operation of E according to SPT
        djj = find(d)
        d(d==0) = []
        dmk(dmk==0) = []
        [durut, idxd]=sort(d);
        % · for j=1 ; 1<=<=N
        for j = 1:colX
            % · calculate starting time
            t(i,j) = max(dm(dmk(idxd(j))), dj(djj(idxd(j))))
            % · update dm
            dm(dmk(idxd(j))) = t(i,j) + durut(j)
            % · update dj
            dj(djj(idxd(j))) = t(i,j) + durut(j)
        end
        X=X-1;
        X(X==0)=[];
        [rowX colX] = size(X);
    end
    if max(dm) < 295
       makespan(k) = max(dm)
    end
end
```

Appendix 3 Matlab code for Schemata Generation Algorithm

Schemata Generation

```
idxmakespan = find(makespan)

Sch = zeros(size(pijk))

Szcardinal = zeros(size(pijk))



for i=1:size(idxmakespan,2)
    Szcardinal(:,:,i) = Sz(:,:,idxmakespan(i))/size(idxmakespan,2)

end



Sch = sum(Szcardinal,3)
```

Appendix 4 Matlab code for Crossover

Crossover

```
coprob=0.88
mutprob=0.12


numco = round(coprob*size(idxmakespan,2))
nummut = round(mutprob*numco)


comakespan = idxmakespan(randperm(size(idxmakespan,2),numco))
mutmakespan = comakespan(randperm(size(comakespan,2),nummut))


x=idxmakespan(randi(size(idxmakespan,2),1,2))
S1 = Sz(:,:,x(1))
S2 = Sz(:,:,x(2))
[rowS1, colS1] = size(S1)


batas=sort(randi(rowS1,1,2))


e1=zeros(size(S1))
e2=zeros(size(S1))


for i = 1:rowS1
    if i<batas(1) | i > batas(2)
        e1(i,:) = S2(i,:)
        e2(i,:) = S1(i,:)
   else
        e1(i,:) = S1(i,:)
        e2(i,:) = S2(i,:)
    end
end
```

Appendix 5 Matlab code for Mutation

Mutation

```
child = cat(3,e1,e2)
e4mut=child(:,:,randi(size(child,3)))


% calculate EPT
for j = 1:colE
    epto = 0
    for i = sum(O(1:(j-1)))+1 : sum(O(1:j))
        [~, idxe4mut] = max(e4mut(i,:))
        if j == 1
            epto(i) = pijk(i,idxe4mut)
        else
            epto(i-sum(O(1:(j-1)))) = pijk(i,idxe4mut)
        end
        epto
    end
    ept(j) = sum(epto)
end


eptmax = find(ept==max(ept))
jobeptmax = eptmax(randi(size(eptmax,2)))


for i = sum(O(1:(jobeptmax-1)))+1 : sum(O(1:jobeptmax))
    [~, idxe4mut] = max(e4mut(i,:))
    poj = pijk(i,idxe4mut)
    for k = 1:size(pijk(i,:),2)
        if pijk(i,k) < poj
            e4mut(i,idxe4mut) = 0
            e4mut(i,k) = 1
            poj=pijk(i,k)
        end
    end
end
```

Appendix 6 Existing Assignment

		M1	M2	M3	M4	M5	M6	M7	M8	M9
J1	O1,1	0	0	1	0	0	0	0	0	0
	O2,1	0	0	0	1	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	1	0
	O4,1	0	0	0	0	0	1	0	0	0
J2	O1,2	1	0	0	0	0	0	0	0	0
	O2,2	0	0	1	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	1	0	0
	O4,2	0	0	0	1	0	0	0	0	0
J3	O1,3	0	0	0	0	0	1	0	0	0
	O2,3	0	0	1	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	1
J4	O1,4	0	0	0	0	0	1	0	0	0
	O2,4	0	0	0	0	1	0	0	0	0
	O3,4	0	0	0	0	0	0	0	0	1
J5	O1,5	0	0	0	0	1	0	0	0	0
	O2,5	0	0	0	0	0	0	1	0	0
J6	O1,6	0	0	0	0	0	1	0	0	0
	O2,6	0	0	0	0	0	1	0	0	0
	O3,6	0	0	0	0	0	0	0	1	0
	O4,6	0	0	0	0	1	0	0	0	0
J7	O1,7	0	0	1	0	0	0	0	0	0
	O2,7	0	0	1	0	0	0	0	0	0
	O3,7	0	0	0	0	0	1	0	0	0
	O4,7	0	0	0	0	0	0	1	0	0
	O5,7	0	0	0	0	0	0	0	1	0
J8	O1,8	1	0	0	0	0	0	0	0	0
	O2,8	0	1	0	0	0	0	0	0	0
J9	O1,9	0	0	0	0	0	0	1	0	0
	O2,9	0	0	0	0	0	0	0	0	1
J10	O1,10	0	0	0	0	0	1	0	0	0
	O2,10	0	0	0	0	0	0	0	0	1
J11	O1,11	0	0	0	1	0	0	0	0	0
	O2,11	0	0	0	1	0	0	0	0	0
	O3,11	0	0	0	0	1	0	0	0	0
J12	O1,12	0	1	0	0	0	0	0	0	0
	O2,12	0	1	0	0	0	0	0	0	0
	O3,12	0	1	0	0	0	0	0	0	0
J13	O1,13	0	1	0	0	0	0	0	0	0
	O2,13	0	0	0	0	0	0	0	0	1
	O3,13	0	0	0	0	0	0	0	1	0
J14	O1,14	1	0	0	0	0	0	0	0	0
	O2,14	1	0	0	0	0	0	0	0	0
	O3,14	0	0	0	0	0	0	0	0	1
	O4,14	0	0	0	0	0	0	0	0	1
J15	O1,15	0	1	0	0	0	0	0	0	0
	O2,15	0	1	0	0	0	0	0	0	0
J16	O1,16	0	0	0	0	0	0	0	0	1
	O2,16	0	0	0	0	0	0	0	0	1
J17	O1,17	0	0	0	0	0	0	0	1	0
J18	O1,18	0	0	0	0	0	0	0	1	0
J19	O1,19	0	0	0	0	1	0	0	0	0
J20	O1,20	0	0	0	0	0	0	0	0	1

Appendix 7 Processing Times

		TC-110	Feeder	TC-77_A	TC-77_B	TC-77_C	TC-77_D	C400_A	C400_B	C400_C
23_031_400_01	O1,1	5	5	6	6	6	6	X	X	X
	O2,1	85	80	100	100	100	100	X	X	X
	O3,1	X	X	X	X	X	X	60	60	60
	O4,1	20	20	25	25	25	25	X	X	X
23_031_110_01	O1,2	40	45	55	55	55	55	X	X	X
	O2,2	50	55	70	70	70	70	X	X	X
	O3,2	X	X	X	X	X	X	35	35	35
	O4,2	24	20	30	30	30	30	X	X	X
23_032_200_02	O1,3	17	13	20	20	20	20	X	X	X
	O2,3	50	55	70	70	70	70	X	X	X
	O3,3	X	X	X	X	X	X	35	35	35
23_032_200_02	O1,4	5	5	8	8	8	8	X	X	X
	O2,4	50	55	70	70	70	70	X	X	X
	O3,4	X	X	X	X	X	X	35	35	35
23_022_000_01	O1,5	35	40	45	45	45	45	X	X	X
	O2,5	X	X	X	X	X	X	60	60	60
23_032_122_01	O1,6	5	5	5	5	5	5	X	X	X
	O2,6	7	7	7	7	7	7	X	X	X
	O3,6	X	X	X	X	X	X	50	50	50
	O4,6	15	15	15	15	15	15	X	X	X
23_032_110_01	O1,7	5	5	5	5	5	5	X	X	X
	O2,7	7	7	7	7	7	7	X	X	X
	O3,7	5	5	5	5	5	5	X	X	X
	O4,7	X	X	X	X	X	X	45	45	45
	O5,7	X	X	X	X	X	X	4	4	4
23_021_110_01	O1,8	25	25	35	35	35	35	X	X	X
	O2,8	35	30	50	50	50	50	X	X	X
23_032_140_01	O1,9	X	X	X	X	X	X	50	50	50
	O2,9	X	X	X	X	X	X	8	8	8
23_022_100_01	O1,10	18	20	26	26	26	26	X	X	X
	O2,10	X	X	X	X	X	X	35	35	35
23_031_500_01	O1,11	5	5	5	5	5	5	X	X	X
	O2,11	13	13	13	13	13	13	X	X	X
	O3,11	15	15	15	15	15	15	X	X	X
23_024_100_01	O1,12	10	10	12	12	12	12	X	X	X
	O2,12	10	10	12	12	12	12	X	X	X
	O3,12	10	10	12	12	12	12	X	X	X
23_032_200_03	O1,13	10	12	20	20	20	20	X	X	X
	O2,13	X	X	X	X	X	X	4	4	4
	O3,13	X	X	X	X	X	X	13	13	13
23_032_200_01	O1,14	5	5	6	6	6	6	X	X	X
	O2,14	3	3	3	3	3	3	X	X	X
	O3,14	X	X	X	X	X	X	3	3	3
	O4,14	X	X	X	X	X	X	15	15	15
23_021_400_01	O1,15	10	12	15	15	15	15	X	X	X
	O2,15	3	3	3	3	3	3	X	X	X
23_032_000_05	O1,16	X	X	X	X	X	X	10	10	10
	O2,16	X	X	X	X	X	X	4	4	4
23_032_100_02	O1,17	X	X	X	X	X	X	10	10	10
23_032_130_04	O1,18	X	X	X	X	X	X	10	10	10
23_020_000_01	O1,19	6	6	7	7	7	7	X	X	X
23_031_400_02	O1,20	X	X	X	X	X	X	5	5	5

Appendix 8 Assignment (chromosome) obtained from Assignment procedure (S^1)

		TC-110	Feeder	TC-77_A	TC-77_B	TC-77_C	TC-77_D	C400_A	C400_B	C400_C
23_031_400_01	O1,1	0	1	0	0	0	0	0	0	0
	O2,1	0	1	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	1	0	0
	O4,1	1	0	0	0	0	0	0	0	0
23_031_110_01	O1,2	0	0	0	0	1	0	0	0	0
	O2,2	1	0	0	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	0	1	0
	O4,2	0	0	0	1	0	0	0	0	0
23_032_200_02	O1,3	0	0	0	0	0	1	0	0	0
	O2,3	0	0	1	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	1
23_032_200_02	O1,4	0	0	0	0	0	1	0	0	0
	O2,4	0	0	0	1	0	0	0	0	0
	O3,4	0	0	0	0	0	0	0	1	0
23_022_000_01	O1,5	0	0	0	0	0	1	0	0	0
	O2,5	0	0	0	0	0	0	0	0	1
23_032_122_01	O1,6	0	0	0	0	1	0	0	0	0
	O2,6	0	0	1	0	0	0	0	0	0
	O3,6	0	0	0	0	0	0	1	0	0
	O4,6	0	1	0	0	0	0	0	0	0
23_032_110_01	O1,7	1	0	0	0	0	0	0	0	0
	O2,7	0	0	0	0	1	0	0	0	0
	O3,7	0	0	0	1	0	0	0	0	0
	O4,7	0	0	0	0	0	0	0	1	0
	O5,7	0	0	0	0	0	0	0	0	1
23_021_110_01	O1,8	0	0	0	0	0	1	0	0	0
	O2,8	0	0	1	0	0	0	0	0	0
23_032_140_01	O1,9	0	0	0	0	0	0	1	0	0
	O2,9	0	0	0	0	0	0	0	1	0
23_022_100_01	O1,10	1	0	0	0	0	0	0	0	0
	O2,10	0	0	0	0	0	0	0	0	1
23_031_500_01	O1,11	0	1	0	0	0	0	0	0	0
	O2,11	0	0	0	0	1	0	0	0	0
	O3,11	0	0	0	1	0	0	0	0	0
23_024_100_01	O1,12	0	0	0	0	0	1	0	0	0
	O2,12	0	0	1	0	0	0	0	0	0
	O3,12	1	0	0	0	0	0	0	0	0
23_032_200_03	O1,13	0	1	0	0	0	0	0	0	0
	O2,13	0	0	0	0	0	0	1	0	0
	O3,13	0	0	0	0	0	0	0	1	0
23_032_200_01	O1,14	0	0	0	0	1	0	0	0	0
	O2,14	0	0	0	1	0	0	0	0	0
	O3,14	0	0	0	0	0	0	0	0	1
	O4,14	0	0	0	0	0	0	1	0	0
23_021_400_01	O1,15	0	0	0	0	0	1	0	0	0
	O2,15	0	0	1	0	0	0	0	0	0
23_032_000_05	O1,16	0	0	0	0	0	0	0	1	0
	O2,16	0	0	0	0	0	0	0	0	1
23_032_100_02	O1,17	0	0	0	0	0	0	1	0	0
23_032_130_04	O1,18	0	0	0	0	0	0	0	1	0
23_020_000_01	O1,19	1	0	0	0	0	0	0	0	0
23_031_400_02	O1,20	0	0	0	0	0	0	0	0	1

Appendix 9 Assignment with makespan of 235

		TC-110	Feeder	TC-77_A	TC-77_B	TC-77_C	TC-77_D	C400_A	C400_B	C400_C
23_031_400_01	O1,1	1	0	0	0	0	0	0	0	0
	O2,1	0	1	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	0	1
	O4,1	0	0	0	0	1	0	0	0	0
23_031_110_01	O1,2	1	0	0	0	0	0	0	0	0
	O2,2	0	0	1	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	1	0	0
	O4,2	0	0	0	0	0	1	0	0	0
23_032_200_02	O1,3	0	0	0	1	0	0	0	0	0
	O2,3	0	0	0	1	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	1	0
23_032_200_02	O1,4	0	0	0	0	1	0	0	0	0
	O2,4	1	0	0	0	0	0	0	0	0
	O3,4	0	0	0	0	0	0	0	1	0
23_022_000_01	O1,5	0	0	0	0	0	1	0	0	0
	O2,5	0	0	0	0	0	0	1	0	0
23_032_122_01	O1,6	0	0	0	0	1	0	0	0	0
	O2,6	0	0	1	0	0	0	0	0	0
	O3,6	0	0	0	0	0	0	0	0	1
	O4,6	0	1	0	0	0	0	0	0	0
23_032_110_01	O1,7	0	0	0	0	0	1	0	0	0
	O2,7	0	0	0	1	0	0	0	0	0
	O3,7	0	0	0	0	1	0	0	0	0
	O4,7	0	0	0	0	0	0	0	1	0
	O5,7	0	0	0	0	0	0	1	0	0
23_021_110_01	O1,8	1	0	0	0	0	0	0	0	0
	O2,8	0	0	1	0	0	0	0	0	0
23_032_140_01	O1,9	0	0	0	0	0	0	0	0	1
	O2,9	0	0	0	0	0	0	0	1	0
23_022_100_01	O1,10	0	1	0	0	0	0	0	0	0
	O2,10	0	0	0	0	0	0	1	0	0
23_031_500_01	O1,11	0	0	0	0	0	1	0	0	0
	O2,11	0	0	0	1	0	0	0	0	0
	O3,11	0	0	0	0	1	0	0	0	0
23_024_100_01	O1,12	1	0	0	0	0	0	0	0	0
	O2,12	0	0	1	0	0	0	0	0	0
	O3,12	0	1	0	0	0	0	0	0	0
23_032_200_03	O1,13	0	0	0	0	0	1	0	0	0
	O2,13	0	0	0	0	0	0	0	0	1
	O3,13	0	0	0	0	0	0	0	1	0
23_032_200_01	O1,14	0	0	0	1	0	0	0	0	0
	O2,14	0	0	0	0	1	0	0	0	0
	O3,14	0	0	0	0	0	0	1	0	0
	O4,14	0	0	0	0	0	0	0	0	1
23_021_400_01	O1,15	1	0	0	0	0	0	0	0	0
	O2,15	0	0	1	0	0	0	0	0	0
23_032_000_05	O1,16	0	0	0	0	0	0	0	1	0
	O2,16	0	0	0	0	0	0	1	0	0
23_032_100_02	O1,17	0	0	0	0	0	0	0	0	1
23_032_130_04	O1,18	0	0	0	0	0	0	0	1	0
23_020_000_01	O1,19	0	1	0	0	0	0	0	0	0
23_031_400_02	O1,20	0	0	0	0	0	0	1	0	0

Appendix 10 Assignment Schemata (S^{ch})

		TC-110	Feeder	TC-77_A	TC-77_B	TC-77_C	TC-77_D	C400_A	C400_B	C400_C
23_031_400_01	O1,1	*	*	0	0	0	0	0	0	0
	O2,1	*	*	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	*	*	*
	O4,1	*	*	*	*	*	*	0	0	0
23_031_110_01	O1,2	*	*	*	0	*	*	0	0	0
	O2,2	*	0	*	*	*	*	0	0	0
	O3,2	0	0	0	0	0	0	*	*	*
	O4,2	0	0	*	*	*	*	0	0	0
23_032_200_02	O1,3	0	0	*	*	*	*	0	0	0
	O2,3	*	*	*	*	*	*	0	0	0
	O3,3	0	0	0	0	0	0	*	*	*
23_032_200_02	O1,4	0	0	*	*	*	*	0	0	0
	O2,4	*	0	*	*	*	*	0	0	0
	O3,4	0	0	0	0	0	0	*	*	*
23_022_000_01	O1,5	*	*	*	*	*	*	0	0	0
	O2,5	0	0	0	0	0	0	*	*	*
23_032_122_01	O1,6	0	0	*	*	*	*	0	0	0
	O2,6	0	0	*	*	*	*	0	0	0
	O3,6	0	0	0	0	0	0	*	*	*
	O4,6	*	*	0	0	0	0	0	0	0
23_032_110_01	O1,7	*	*	*	*	*	*	0	0	0
	O2,7	0	*	*	*	*	*	0	0	0
	O3,7	0	0	*	*	*	*	0	0	0
	O4,7	0	0	0	0	0	0	*	*	*
	O5,7	0	0	0	0	0	0	*	*	*
23_021_110_01	O1,8	*	*	*	*	*	*	0	0	0
	O2,8	0	0	*	*	*	*	0	0	0
23_032_140_01	O1,9	0	0	0	0	0	0	*	*	*
	O2,9	0	0	0	0	0	0	*	*	*
23_022_100_01	O1,10	*	*	0	0	0	0	0	0	0
	O2,10	0	0	0	0	0	0	*	*	*
23_031_500_01	O1,11	*	*	*	*	*	*	0	0	0
	O2,11	0	*	*	*	*	*	0	0	0
	O3,11	0	0	*	*	*	*	0	0	0
23_024_100_01	O1,12	*	*	*	*	*	*	0	0	0
	O2,12	0	0	*	*	*	*	0	0	0
	O3,12	*	*	0	0	0	0	0	0	0
23_032_200_03	O1,13	*	*	*	*	*	*	0	0	0
	O2,13	0	0	0	0	0	0	*	*	*
	O3,13	0	0	0	0	0	0	*	*	*
23_032_200_01	O1,14	0	*	*	*	*	*	0	0	0
	O2,14	0	0	*	*	*	*	0	0	0
	O3,14	0	0	0	0	0	0	*	*	*
	O4,14	0	0	0	0	0	0	*	*	*
23_021_400_01	O1,15	*	*	*	*	*	*	0	0	0
	O2,15	0	0	*	*	*	*	0	0	0
23_032_000_05	O1,16	0	0	0	0	0	0	*	*	*
	O2,16	0	0	0	0	0	0	*	*	*
23_032_100_02	O1,17	0	0	0	0	0	0	*	*	*
23_032_130_04	O1,18	0	0	0	0	0	0	*	*	*
23_020_000_01	O1,19	*	*	0	0	0	0	0	0	0
23_031_400_02	O1,20	0	0	0	0	0	0	*	*	*

Appendix 11 Assignment with makespan of 204 obtained from Crossover

		TC-110	Feeder	TC-77_A	TC-77_B	TC-77_C	TC-77_D	C400_A	C400_B	C400_C
23_031_400_01	O1,1	0	1	0	0	0	0	0	0	0
	O2,1	0	1	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	1	0
	O4,1	1	0	0	0	0	0	0	0	0
23_031_110_01	O1,2	0	0	0	0	1	0	0	0	0
	O2,2	0	0	0	1	0	0	0	0	0
	O3,2	0	0	0	0	0	0	1	0	0
	O4,2	0	0	0	0	0	1	0	0	0
23_032_200_02	O1,3	0	0	1	0	0	0	0	0	0
	O2,3	1	0	0	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	1
23_032_200_02	O1,4	0	0	1	0	0	0	0	0	0
	O2,4	0	0	0	0	0	1	0	0	0
	O3,4	0	0	0	0	0	0	0	0	1
23_022_000_01	O1,5	0	0	1	0	0	0	0	0	0
	O2,5	0	0	0	0	0	0	1	0	0
23_032_122_01	O1,6	0	0	0	0	1	0	0	0	0
	O2,6	0	0	0	1	0	0	0	0	0
	O3,6	0	0	0	0	0	0	0	1	0
	O4,6	1	0	0	0	0	0	0	0	0
23_032_110_01	O1,7	0	1	0	0	0	0	0	0	0
	O2,7	0	0	0	0	1	0	0	0	0
	O3,7	0	0	0	0	0	1	0	0	0
	O4,7	0	0	0	0	0	0	0	0	1
	O5,7	0	0	0	0	0	0	0	1	0
23_021_110_01	O1,8	0	0	0	1	0	0	0	0	0
	O2,8	0	0	0	0	1	0	0	0	0
23_032_140_01	O1,9	0	0	0	0	0	0	1	0	0
	O2,9	0	0	0	0	0	0	0	0	1
23_022_100_01	O1,10	1	0	0	0	0	0	0	0	0
	O2,10	0	0	0	0	0	0	0	1	0
23_031_500_01	O1,11	0	1	0	0	0	0	0	0	0
	O2,11	0	0	1	0	0	0	0	0	0
	O3,11	0	0	0	0	0	1	0	0	0
23_024_100_01	O1,12	0	0	0	1	0	0	0	0	0
	O2,12	0	0	0	0	1	0	0	0	0
	O3,12	1	0	0	0	0	0	0	0	0
23_032_200_03	O1,13	0	1	0	0	0	0	0	0	0
	O2,13	0	0	0	0	0	0	1	0	0
	O3,13	0	0	0	0	0	0	0	0	1
23_032_200_01	O1,14	0	0	1	0	0	0	0	0	0
	O2,14	0	0	0	0	0	1	0	0	0
	O3,14	0	0	0	0	0	0	0	1	0
	O4,14	0	0	0	0	0	0	1	0	0
23_021_400_01	O1,15	0	0	0	1	0	0	0	0	0
	O2,15	0	0	0	0	1	0	0	0	0
23_032_000_05	O1,16	0	0	0	0	0	0	0	0	1
	O2,16	0	0	0	0	0	0	0	1	0
23_032_100_02	O1,17	0	0	0	0	0	0	1	0	0
23_032_130_04	O1,18	0	0	0	0	0	0	0	0	1
23_020_000_01	O1,19	1	0	0	0	0	0	0	0	0
23_031_400_02	O1,20	0	0	0	0	0	0	0	1	0

Appendix 12 Assignment with makespan of 204 obtained from Mutation

		TC-110	Feeder	TC-77_A	TC-77_B	TC-77_C	TC-77_D	C400_A	C400_B	C400_C
23_031_400_01	O1,1	0	1	0	0	0	0	0	0	0
	O2,1	0	1	0	0	0	0	0	0	0
	O3,1	0	0	0	0	0	0	0	1	0
	O4,1	1	0	0	0	0	0	0	0	0
23_031_110_01	O1,2	1	0	0	0	0	0	0	0	0
	O2,2	1	0	0	0	0	0	0	0	0
	O3,2	0	0	0	0	0	0	1	0	0
	O4,2	1	1	0	0	0	0	0	0	0
23_032_200_02	O1,3	0	0	1	0	0	0	0	0	0
	O2,3	1	0	0	0	0	0	0	0	0
	O3,3	0	0	0	0	0	0	0	0	1
23_032_200_02	O1,4	0	0	1	0	0	0	0	0	0
	O2,4	0	0	0	0	0	1	0	0	0
	O3,4	0	0	0	0	0	0	0	0	1
23_022_000_01	O1,5	0	0	1	0	0	0	0	0	0
	O2,5	0	0	0	0	0	0	1	0	0
23_032_122_01	O1,6	0	0	0	0	1	0	0	0	0
	O2,6	0	0	0	1	0	0	0	0	0
	O3,6	0	0	0	0	0	0	0	1	0
	O4,6	1	0	0	0	0	0	0	0	0
23_032_110_01	O1,7	0	1	0	0	0	0	0	0	0
	O2,7	0	0	0	0	1	0	0	0	0
	O3,7	0	0	0	0	0	1	0	0	0
	O4,7	0	0	0	0	0	0	0	0	1
	O5,7	0	0	0	0	0	0	0	1	0
23_021_110_01	O1,8	0	0	0	1	0	0	0	0	0
	O2,8	0	0	0	0	1	0	0	0	0
23_032_140_01	O1,9	0	0	0	0	0	0	1	0	0
	O2,9	0	0	0	0	0	0	0	0	1
23_022_100_01	O1,10	1	0	0	0	0	0	0	0	0
	O2,10	0	0	0	0	0	0	0	1	0
23_031_500_01	O1,11	0	1	0	0	0	0	0	0	0
	O2,11	0	0	1	0	0	0	0	0	0
	O3,11	0	0	0	0	0	1	0	0	0
23_024_100_01	O1,12	0	0	0	1	0	0	0	0	0
	O2,12	0	0	0	0	1	0	0	0	0
	O3,12	1	0	0	0	0	0	0	0	0
23_032_200_03	O1,13	0	1	0	0	0	0	0	0	0
	O2,13	0	0	0	0	0	0	1	0	0
	O3,13	0	0	0	0	0	0	0	0	1
23_032_200_01	O1,14	0	0	1	0	0	0	0	0	0
	O2,14	0	0	0	0	0	1	0	0	0
	O3,14	0	0	0	0	0	0	0	1	0
	O4,14	0	0	0	0	0	0	1	0	0
23_021_400_01	O1,15	0	0	0	1	0	0	0	0	0
	O2,15	0	0	0	0	1	0	0	0	0
23_032_000_05	O1,16	0	0	0	0	0	0	0	0	1
	O2,16	0	0	0	0	0	0	0	1	0
23_032_100_02	O1,17	0	0	0	0	0	0	1	0	0
23_032_130_04	O1,18	0	0	0	0	0	0	0	0	1
23_020_000_01	O1,19	1	0	0	0	0	0	0	0	0
23_031_400_02	O1,20	0	0	0	0	0	0	0	1	0