Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Dynamic heuristics for the generalized job-shop scheduling problem

Fatima Ghedjati
CReSTIC-Reims (URCA)
Moulin de la Housse BP 1039
51687 Reims cedex2-France
fatima.ghedjati@univ-reims.fr

Marie-Claude Portmann
LORIA, Ecole des mines de Nancy
Parc de Saurupt, CS 14234
54042 Nancy cedex-France
portmann@loria.fr

***Abstract*—This paper proposes to solve the generalized job-shop scheduling problem by using several original static and dynamic heuristics relying on the machines' potential load. We consider a generalized job-shop problem with unrelated parallel machines which can process the operations of the different jobs and, moreover, any precedence constraints between the operations are allowed. The objective is to minimize the completion date of all the jobs (makespan). This problem is NP-hard. Experimental results using various important randomly generated benchmarks are satisfactory and promising.**

***Keywords*—scheduling, generalized job-shop, unrelated parallel machines, linear and non-linear process routing, static/dynamic heuristics.**

I. INTRODUCTION

The interest of industry needs effective and fast solving methods for the resolution of scheduling problems. Many of these latter are referred as NP-hard [12]. For NP-hard problems, exact search methods which ensure optimality have an exponential time and are not suitable for industrial-size problems [26]. Therefore, a great number of researchers adopted approximation methods, which can not generally ensure optimality, but provide good solutions in a reasonable time.

These approximate methods could be classified in several large families as:

- construction methods [2], decomposition methods [24], neighbourhood methods such as simulated annealing [1], tabu search methods [14, 29];
- methods issued from the artificial intelligence like SMA approach [9];
- methods inspired from biologic phenomena: genetic algorithms [15, 13], neuronal networks [17], ant colony systems [4, 27] and artificial immune systems [21].

In this paper, we are interested in the job-shop factory scheduling problems with several unrelated parallel machines see for example [18, 7] and precedence constraints between the operations of a job. This problem is NP-hard since a simpler problem with two identical machines and with a job restricted to one operation has been shown NP-hard [28]. We propose different new heuristics. Our paper is divided into four sections, the second section describes the problem, the third one presents the heuristics used in order to solve the considered problem, and finally, the fourth section reports the experimental results.

II. PROBLEM DESCRIPTION

We consider a very general and realistic job-shop problem, since it is the one commonly found in the factories of small series. We are therefore interested in a factory which is composed by m machines. These machines have to manufacture n parts. Each part j consists of a sequence of K operations $O_{1,j}$, ..., $O_{k,j}$. One or several machines can process an operation $O_{i,j}$ with different processing times; there exist a set of $\mathcal{M}$ machines (not necessarily identical) associated to each operation $O_{i,j}$. The considered operation has to be processed on only one machine r in $\mathcal{M}$ during $p_{i,j,r}$ time units without preemption. We consider on the one hand, the simple case of a linear process routing (case of the classical job-shop, each job j has to be processed in order of increasing indices, i.e. $O_{i+1,j}$ can start only if $O_{i,j}$ has already been completed, see for example [6]), and, on the other hand, the more generalized cases which deal with whatever precedence graph, without circuit and not necessarily related between the job operations. However, as it is about the same physical part, we don't accept an overlap between the executions of two operations of the same part. The problem is static: it means that all jobs to treat are known and can start at the date zero. The goal of scheduling is to minimize the total elapsed time between the beginning of the first operation and the completion of the last operation (the makespan denoted by C_{max}).

Using the well-known $\alpha/\beta/\gamma$ notation of [16] and which was revised by [20] (see also [3]), where the parameter α describes the machines' environment, the second field β represents the jobs' features, and the third field γ specifies the optimisation criteria. So, we formulate the considered problem as $J(R)/prec/C_{max}$.

III. THE PROPOSED APPROACH

Heuristics have taken an important position in the research of solutions in the combinatory problems [5, 10, 19, 25], and the most spread software programs are based on heuristics. A good heuristic must be, in a general manner, simple, easy to implement and must propose a good result, so that we could know if we are near the optimum and finally, produce an

acceptable solution in a reasonable time. Alternative description of a good heuristic can be found in the literature [8, 30]. For the job-shop problems, we find a great number of heuristics in for instance [2, 11, 22].

To solve the considered problem, we constructed a family of heuristics based on the progressive construction approach. Construction methods are iterative methods, which complete step by step a partial solution. Our different heuristics consist in taking a decision at each algorithm's iteration (schedule generator) regarding on one hand, the machines' affectation to the operations and on the other hand, the operations' scheduling on the machines.

A. Description of the schedule generator

Our approach is based on the construction methods. We use the strategy operation by operation for the construction of a non delay schedule generator, i.e. we don't let a vacant machine if an operation is waiting to be executed by this machine and has not been definitely assigned to another machine. Our basic algorithm or schedule generator consists in filling the intervals of time (dynamically created) in parallel on all the machines. It seeks, at each instant *t*, the first available machine having a non-empty list of candidate operations (called candidate list). The candidate operations are issued from a selection among the operations of the jobs to be processed by the current machine, and are added to the candidate list. This latter is sorted according to defined priority rules. We kept essentially the rules: SPT (Shortest Processing Time) and SPT/RANDOM (we apply SPT but in conflict case, we make a random selection). Each machine is successively examined (that we call the current machine). When there are no more available machines, the time counter progresses by one unit. Every selected operation to a placement should be assigned to a machine according to a heuristic and a specific technique of a scheduler generator. If there are no multiple choices for this operation, we place it in the planning's current time interval of the current machine. We put the state of the machine to "occupied", the state of the operation to "placed" and we remove this operation from the candidate list of the considered machine (we have therefore in this case, an automatic placement). In the case of multiple choice machines, we denote several types of heuristically defined priority rules such as: static, dynamic or hybrid. Static rules assign to each machine a priority which could be determined before the process of scheduling and which remain unchangeable during this process. The dynamic rules determine the priorities of the machines during the process of scheduling, i.e. that they use an instantaneous knowledge of a system. In the case where several machines have the same priority, we could apply another rule (arbitrary choice, priority to the first available machine, etc), or apply what we call hybrid priority rules, for example a combination of static and dynamic ones. Several choice of priority rules were discussed in literature, we notice that the numerous simulations didn't allow to associate in an efficient manner "good rules" to given problems [23]. In our case, we use static heuristics (H1, H2) and dynamic ones (H3, H4, H5) that we have designed and developed. These heuristics are detailed in what follows.

Example 1: To illustrate the heuristics algorithm, we consider the (3x8) following instance, which is a simplification for our general problem (not a precedence constraints and where a job is restricted to one operation). 8 jobs are to be scheduled on 3 non identical parallel machines with the processing times given in Table I.

TABLE I. PROCESSING TIME OF THE CONSIDERED EXAMPLE

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	6	3	10	12	11	14	8	6
M_2	10		15	6	6	11	14	7
M_3	11	9	14	14		10	10	9

B. Static Heuristics

These heuristics are called static because they don't take into account the machines' loads during the construction of the scheduling.

- **The heuristic H1** affects the highest priority operation (the first operation from the candidate list) to the first available machine.
- **The heuristic H2** affects the highest priority operation to the fastest machine.

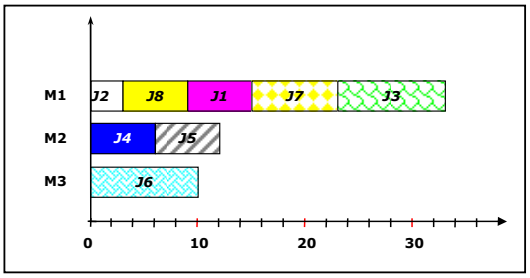

Figure 1. Gantt chart for the instance of Example1 (application of H2)

The heuristics H1 and H2 don't take into account the machines' load. We kept them in order to serve as a comparison with the dynamic heuristics.

C. Dynamic heuristics

In this paper, we are particularly interested in the dynamic introduction and the dynamic modification of heuristics. This approach is opportunistic because it controls the evolution of the machines' loads during the construction of the scheduling, and assigns in general the highest priority operation to the least loaded machine in the current state of the system. We notice that we don't make "backtracking", i.e. once we decide to assign an operation to a machine, this decision will be definitive even if we perceive later that the supposed least loaded machine is in fact the most loaded. The heuristics that we conceived are new and particular. They are either based on the maximal potential machines' load, or using "probabilities" of the machines use. The main idea of the family of heuristics using "probabilities" is to assign operations to machines with a dynamically computed compromise between choosing the fastest machine and the least loaded one. As the machines' load depends on the assignment, we use an iterative method. Considering that the load is known, we can compute "probabilities" for machine's choice and afterwards we can

modify consequently, on the one hand, the machine's loads in according to the "probabilities", and on the other hand, "probabilities" for machine's choice in according to the machine's loads, until stability. These heuristics are detailed here after.

We first reported the following notations:

***Ld*:** machine's load;

***L_{Mi}*:** list of the operations which can be processed by the machine *Mi*;

***o'*:** operations not yet placed and not affected to another machine;

***Mi'*:** machines which can process the operation o;

***rdop*:** remaining duration of the current operation;

inda: attraction index;

***nbmceo*:** number of machines which can execute operation *o;*

El: expected load;

1) Heuristics based on the maximal potential load of the machines

- **The heuristic H3** assigns the highest priority operation to the least loaded machine. The machines' load is dynamically computed during the solution's construction. At the instant *t(*where we have to place an operation with multiple choices of machines), the potential load of each machine which can execute this operation, consists of the remaining duration of the operation in progress added to the sum of the duration of all operations candidate list not yet processed and not yet affected to another machine. In the case where the current machine is not the least loaded, the operation is definitely affected to the least loaded one.

Computing the machines load depending on time is made as following:

t = 0: Computation of the initial load for any machine *Mi* according to the duration *p*(o, *Mi*) of any operation o (not yet scheduled) that it could accomplish:

$$ld(Mi)(0) = \sum_{o \in L_{Mi}} p(o, Mi)$$

Any t: Computation (dynamically) of the load for any machine *Mi* starting from the instant *t*

$$ld(Mi)(t) = rdop + \sum_{o' \in L_{Mi}} p(o', Mi)$$

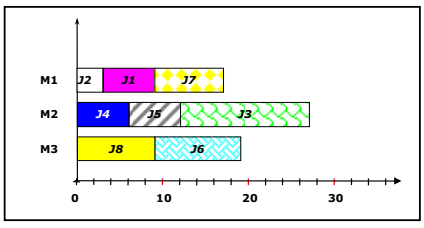

Figure 2. Gantt chart for the instance of Example1 (application of H3)

2) Heuristics using "probabilities"

These heuristics calculate dynamically (for the selection of a machine) the machines' potential load during the construction of the scheduling. The potential load is used to select the machine to be assigned to the highest priority operation, by doing what we hope to be a "good" compromise between assigning the operation to the potentially least loaded machine and assigning the operation to one of the machines which execute it as quickly as possible. Otherwise, an operation *o* which could be executed by several machines, $M_{o,1}$, $M_{o,2}$, …, $M_{o,r}$ will have more chance of being placed on the machine *Mi*, when the duration *p*(*o,Mi*) is small and when the machine *Mi* is less loaded. The problem is that this reasoning is recursive, because the machines' potential load depends on the manner in which we assign the operations to the machines. While an operation is not permanently affected to a machine, we will assume in the computation of the potential load that the operation's duration is distributed between the machines which can process it. The proportion of the duration of each machine is computed so that it is accordingly bigger than the machine is potentially less loaded and that it executes more quickly the operation. This relationship can be interpreted like a "probability" because the machines' potential load could be interpreted as an expectation value of load and takes some real values included between 0 and 1 of which the sum is equal to 1. We can also think this relationship in terms of (what we call) "attraction index" because this value will also be used as a priority indicator in the heuristics.

As the attraction index depends on the potential load of the machines and that this latter depends on the attraction index, the method used to calculate the attraction index could be seen like a method of stationary point using an iterative process of computation. The iterative algorithm's idea that we have constructed at the basis of this method is the following:

1. Initial attraction index
2. Initial machines' potential load (or expected load)
3. Compute the attraction index at the instant *t* in according to the machines' loads
4. Compute the machines' load at the instant *t* in according to the attraction index
5. Repeat steps 3 and 4 until stability.

The general scheme algorithm of the heuristics using "probabilities"

a) Details of the algorithm

The hereunder detailed algorithm starts with initials attraction index, which are identical for all associated multiple machines to an operation (equal probability to select any machine in order to process the considered operation) (step 1 of the algorithm). In function of these attraction indexes, we calculate the initial potential load (or exactly the expected load) of machines (step 2). Then, we calculate again on one hand, the new value of the attraction indexes, in according to the load (step 3); and on the other hand, the new load of machines in according to the attraction indexes (step 4). The algorithm stops when, from iteration to the following iteration, the variation of the attraction indexes becomes inferior to a given threshold.

Experiments we conducted show that in general, this algorithm converges very quickly (in less than 5 iterations). The computation of the attraction indexes is dynamically repeated as the construction of a scheduling each time that we want to place an operation with multiple choices of machines.

1. Initial "attraction index" for any operation *o* not yet placed and for any machine *Mi*:

$$inda(o, Mi)(0) = \frac{1}{nbmceo}$$

2. Initial "expected load" for any machine *Mi* in according to the attraction indexes:

$$El(Mi)(0) = \sum_{o' \in L_{Mi}} inda(o', Mi).p(o', Mi)$$

3. Compute at the instant *t* "the attraction index" for any operation *o* (with multiple choices of machines) not yet placed and for any machine *Mi*:

$$inda(o, Mi)(t) = \frac{\frac{1}{p(o, Mi).El(Mi)(t)}}{\sum_{Mi'} \frac{1}{p(o, Mi').El(Mi')(t)}}$$

4. Compute the expected load starting from the instant *t* for any machine *Mi* according to the attraction indexes:

$$El(Mi)(t) = rdop + \sum_{o' \in L_{Mi}} inda(o', Mi)(t).p(o' Mi)$$

5. Repeat steps 3 and 4 until stability.

Algorithm details of the heuristics using "probabilities"

We have integrated these attraction indexes with two different manners in the construction of our heuristics. In a first family of heuristics, we use the attraction indexes as "probabilities" of assigning an operation to a machine ("*heuristics choice of machines by drawing lots according to the attraction indexes"*). In a second family of heuristics we use the attraction indexes as indicators of the machines priority in order to process the operations ("*heuristics choice of machines on priority"*).

b) Choice of machines by drawing lots according to the attraction indexes

- **The heuristic H4 uses the attraction indexes as "probabilities**". The highest priority operation is attracted toward the most desirable machine which is obtained using a lottery technique on the attraction indexes: the probability for a machine to be drawing lot is proportional to its attraction index. The machines' load and the attraction indexes are dynamically computed again at each time we want to place an operation with multiple choices of machines. As in this heuristic, we ask only at an instant *t*, if yes or no we affect the highest priority operation to the current machine, the used technique is simple. We generate a pseudo-random number included between 0 and 1. If it is smaller than the attraction index of the current machine, then we affect the operation to this machine. When the choice of the current machine is rejected, no definitive affectation of the highest priority operation is foreseen, we pass to the following operation in the candidate list. H4 permits to get random choices, it may be executed many times and so it provides different results. H4(*nb*) consists in using *nb* times the heuristic H4 and keeping the best result. So that its processing time may be compared to those other heuristics, it is necessary to use H4(1).

c) Choice of machines on priority (priority to the biggest value of the attraction indexes)

- **The heuristic H5** is very close to the heuristic H4. However, instead of drawing lots, we use the attraction indexes as indicators of priority. The highest priority operation is affected to the machine having the biggest value of attraction index. In the case where the current machine is not the highest priority machine (therefore not selected), the operation is definitely affected to the selected machine, but will be placed only when it will be the highest priority operation on this machine, and when this latter will be the considered current machine.

All the considered heuristics results of the example 1 are reported in Table II.

TABLE II. HEURISTICS RESULTS OF THE EXAMPLE 1

	H1	H2	H3	H4(1)	H4(5)	H4(10)	H5
Cmax	27	33	27	24	22	22	25

IV. EXPERIMENTAL RESULTS

We have implemented our algorithms in a C/Unix/Sparc10 environment. In order to compare our different heuristics, we have tested them on large samples randomly generated benchmarks. We have randomly generated 360 examples of the considered problem and which have totally different process routings. The precedence graph of each job is chosen according to different linear and non-linear process routing. We have then generated 10 examples for each value of the parameters with:

- 3 values for the number of machines *m* (5, 10 and 15);
- 3 values for the number of jobs *n* (*n* = *m*, 2*m* and 3*m*), with the number of operations *NOP* = *n* if *n* = *m*; else *NOP* = random: [3,9] for the models 1, 2 and 3 (see Figure 1), [4,9] for the model 5 and [5,9] for both of the models 4 and 6;
- 2 values for the multiplicity of the machines (number of machines which can process an operation with multiple choices of machines) "weak: weak probability of having a big choice of multiple machines" and "strong: opposite case of the previous one". we consider 2, 3 or 4 machines with respective probability 1/3, 1/6, or 1/18 for the "weak" case; and probability 1/2 for the "strong" one;

- 2 values for the diversity of the durations: small processing time [1, 5] and large one [1, 50].

We reported the following characteristics of experiments:

m: machines' number;
n: jobs' number;
m x n: problem of m machines and n jobs;
H4(*nb*): (*nb*) number of executions of the stochastic heuristic H4;
AC: average of the completion time (Cmax);
% G: percentage of times where an heuristic (for the subset of retained heuristics), provides the best solution for the Cmax;
ARD: the average relative deviation made if we only use the considered heuristic instead of using the best one.

1) Comparison of the heuristics (H1, …, H5)

The heuristic H4 has the particularity of having a stochastic behaviour and therefore not giving the same result if we execute it several times. Thus, we have done the comparisons with only one execution of H4, 5 executions of H4 (keeping the best found solution) and 10 executions of H4, the other heuristics being executed only once.

The processing time is reasonable (of the order of few seconds).

The obtained results are summarized in Table III, Table IV, and Table V.

TABLE III. COMPARISON OF HEURISTICS (H1, H5) WITH 1 EXECUTION OF H4

	H1	H2	H3	H4(1)	H5
AC	262.36	244.69	284.64	243.49	**226.36**
%G	6	16	5	13	**65**
ARD	0.199	0.125	0.292	0.125	**0.029**

TABLE IV. COMPARISON OF HEURISTICS (H1, H5) WITH 5 EXECUTIONS OF H4

	H1	H2	H3	H4(5)	H5
AC	262.36	244.69	284.64	**222.94**	226.36
%G	3	11	2	44	**47**
ARD	0.217	0.143	0.312	**0.043**	0.046

TABLE V. COMPARISON OF HEURISTICS (H1, H5) WITH 10 EXECUTIONS OF H4

	H1	H2	H3	H4(10)	H5
AC	262.36	244.69	284.64	**217.85**	226.36
%G	3	10	1	**56**	40
ARD	0.227	0.153	0.323	**0.028**	0.056

We notice that with one and five execution(s) of H4, the heuristic H5 is the most efficient in percent of times where it is the best (65% and 47%). H5 is outperformed by H4 in average, only when we execute this latter five or ten times. It is necessary also to note that the heuristics H4(5) or H4(10) and H5 are much better than the static heuristics and are never very far from the best obtained results.

2) Comparison of the heuristics (H4, H5) with the heuristics (H1, H2, H3)

We compare the heuristics using the "probabilities" or the priorities (H4, H5) with the more simple heuristics taking into account or not the loads of machines (H1, H2, H3), in order to synthesize otherwise the previous results. We want in fact to know the contribution of $H4 \cup H5$ compared to $H1 \cup H2 \cup H3$. In a general manner, the principle consists, on the one hand, to select the best solution of the heuristics (H1, H2 and H3), and on the other hand, the best one of the heuristics (H4 and H5). We call **MH1** the heuristic which consists to apply the heuristics (H1, H2, H3) then to keep the best obtained result, and **MH2(*nb*)** the heuristic which consists to apply the heuristics H4(*nb*) and H5 then to keep the best obtained result. This is formulated as follows.

For each individual (i):

$$MH1_i = \min_{H1,H2,H3} C\max_i$$

$$MH2(nb)_i = \min_{H4(nb),H5} C\max_i$$

The results are carried in Table VI, Table VII and Table VIII.

TABLE VI. COMPARISON OF (MH1, MH2) WITH 1 EXECUTION OF H4

	MH1	MH2 H4(1)
AC	235.54	**223.5**
%G	26	**78**
ARD	0.080	**0.016**

TABLE VII. COMPARISON OF (MH1, MH2) WITH 5 EXECUTION OF H4

	MH1	MH2 H4(5)
AC	235.54	**217.29**
%G	17	**87**
ARD	0.097	**0.007**

TABLE VIII. COMPARISON OF (MH1, MH2) WITH 10 EXECUTION OF H4

	MH1	MH2 H4(10)
AC	235.54	**214.44**
%G	14	**91**
ARD	0.106	**0.004**

We notice that using successively 10 times H4 and once H5 rather than the simpler heuristics once each, allows to obtain the best solution in 91% of cases (87% and 78% in performing successively H4 5 times and once) and we make a relative deviation of 10.6% in using the simple methods (H1, H2 and H3) rather than the methods H4 and H5.

V. CONCLUSION

The objective of this paper is the approximate resolution of the generalized job shop scheduling problem with unrelated parallel machines and with precedence constraints (where the

process routings of the jobs are any). We have developed new heuristics: static simple, and dynamic more complexes related to the machines' loads. We assume on one hand that any operation not yet placed participate completely to the load of the capable machines. On the other hand, we try at best to take into account the potential load of the machines by using "probabilities" to affect an operation with multiple choices on the machine which is the fastest and the less loaded one. Our system allows to quickly test these different heuristics and to easily and dynamically switch (at the same time as the construction of the solution) from one heuristic to another without changing the basic schedule generator. The considered criterion is the minimization of the total duration of the scheduling.

Experiments are conducted on a sample of 360 typical randomly generated examples of our considered problem. They show, on the one hand, that the dynamic heuristics are in average better than the static ones. On the other hand, the heuristic H5 using attraction indexes as priority indicators of the machines in order to process the operations (priority to the machine which has the biggest value of attraction index) is in average more efficient than the heuristic H4 (which have stochastic character) if this later is executed less than 5 times. The simultaneous utilization of H4(5) or H4(10) with H5 dominates nearly always the other heuristics. The processing time of our heuristics (all executed only once) is reasonable (about a few seconds).

REFERENCES

[1] E.H.L. Aarts, P.J.M. Van Laarhoven, "Simulated annealing: theory and applications". D. Reidel Publishing Company, Dordrecht, Holland., 1987.

[2] K.R. Baker, "Introduction to sequencing and scheduling". John Wiley and Sons, New-York, 1974.

[3] J. Blazewicz, K.H. Ecker, G. Schmidt, J. Weglarz, "Scheduling in computer and manufacturing systems". Second, Revised Edition, Springer-Verlag, 1994.

[4] C. Blum, M. Samples, "An ant colony optimization algorithm for shop scheduling problems", Journal of Mathematical Modelling and Algorithms, vol. 3, pp. 285-308, 2004.

[5] H.G. Campbell., R.A. Dudek, M.L. Smith, "A heuristic algorithm for the n-job, m-machine sequencing problem". Mgmt. Sci., vol. 16(10), pp. B630-B637, 1970.

[6] J. Carlier, E. Pinson, "An algorithm for solving the job-shop problem". Man. Sci., vol. 35, pp. 164-176, 1989.

[7] E. Davis, J.M. Jaffe, "Algorithms for scheduling task on unrelated processors". Journal of the Association for Computing Machinery, vol. 28(4), pp. 721-736, 1981.

[8] D. De Werra, "Design and operation of flexible manufacturing systems: The kingdom of heuristic methods". R.A.I.R.O Operational Research, vol. 21, 1987.

[9] M. Ennigrou, K. Ghedira, "New local diversification techniques for flexible job shop scheduling problem with a multi-agent approach". Autonomous Agents and Multi-Agent Systems, Springer Netherlands, vol. 17(2), pp. 270-287, 2008.

[10] P.M. França., M. Gendreau, G. Laporte, F.M. Müller, "A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective". Computers and operations research, pp. 205-211, 1994.

[11] S. French., "Sequencing and scheduling: an introduction to the mathematics of the job-shop". Wiley, 1982.

[12] M.R. Garey, D.S. Johnson., "Computers and interactability: a guide to the theory of NP-completeness", Freeman and Co., 1979.

[13] F. Ghedjati, "Genetic algorithm for the factory scheduling problems with parallel machines". The 37th International Conference on Computers and Industrial Engineering, Alexandria (Egypt). CD Proceedings, pp 2116-2127, 2007.

[14] F. Glover, "Tabu search methods in artificial intelligence and operations research". ORSA, Artificial Intelligence Newletter, vol. 1(2), 1987.

[15] D. E. Goldberg, "Genetic algorithms in search, optimisation, and machine learning". Addison-Wesley Publishing Company, Inc., 1989.

[16] R.L. Graham., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, "Optimisation and approximation in deterministic sequencing and scheduling: a survey". Annals of Discrete Mathematics, vol. 5, pp. 287-326, 1979.

[17] J. Hopfield, J. Tank, "Neural computation of decisions in optimisation problems". Biological Cybernetics, vol. 52, pp. 141-152, 1985.

[18] E. Horowitz, S. Sahni, "Exact and approximate algorithms for scheduling non identical processors". Journal of the Association for Computing Machinery, vol. 23(2), pp. 317-327, 1976.

[19] O.H. Ibarra, C.E. Kim., "Heuristic algorithms for scheduling independent tasks on non identical processors". Journal of the Association for computing Machinery, vol. 24(2), pp. 280-289, 1977.

[20] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, "Recent developments in deterministic sequencing and scheduling: a survey". In M.A.H. Dempster, J.K. Lenstra, and A.H.G. Rinnooy Kan, editors, Deterministic and stochastic scheduling, pp. 35-73, Dordrecht, The Netherlands, 1982. NATO Advanced Study and Research Institute, D. Reidel Publishing Company, 1982.

[21] B. Naderi, M. Khalili, R. Tavakkoli-Moghaddam, "A hybrid artificial immune algorithm for a realistic variant of job shops to minimize the total completion time", Elsevier, Computers & Industrial Engineering, in press.

[22] R.T. Nelson, C.A. Holloway, R.M.-L. Wong, "Centralized scheduling and priority implementation heuristics for a dynamic job-shop model". A.I.I.E. Trans., vol. 9, pp. 95-102, 1977.

[23] S.S. Panwalkar, W. Iskandar, "A survey of scheduling rules". Operations Research, vol. 25(1), pp. 45-61, 1977.

[24] M.C. Portmann, Z. Mouloua, "A window time negotiation approach at the scheduling level inside supply chains", MISTA, Paris 28-31 August, pp. 410-417, 2007.

[25] C.N. Potts, "Analysis of a linear programming heuristic for scheduling unrelated parallel machines", Discrete Applied Mathmatics, vol. 10, pp. 155-164, 1985.

[26] A.H.G. Rinnooy Kan, "Machine scheduling problems: classification, complexity and computation", Nijhoff, The Hague, 1976.

[27] P.V. Senthiil., V. Selladurai, R. Rajesh, "Parallel machine scheduling (PMS) in manufacturing systems using the ant colonies optimization algorithmic rule", Journal of Applied Sciences, vol. 7(2), pp. 208-213, 2007.

[28] S.L. Van De Velde, "Duality-based algorithms for scheduling unrelated parallel machines". ORSA Journal on Computing, vol. 5(2), pp. 192-205, 1993.

[29] M. Widmer, "Job-shop scheduling with tooling constraints: a tabu search approach", J. Opl Res. Soc., vol. 42(1), pp. 75-82, 1991.

[30] S.H. Zanakis, J.R. Evans., "Heuristics optimisation: why, when, and how to use it". Interfaces, vol. 11(5), pp. 84-91, 1981.