PARALLEL GENETIC ALGORITHMS WITH MIGRATION FOR THE HYBRID FLOW SHOP SCHEDULING PROBLEM

K. BELKADI, M. GOURGAND, AND M. BENYETTOU

Received 17 March 2006; Revised 17 July 2006; Accepted 2 August 2006

This paper addresses scheduling problems in hybrid flow shop-like systems with a migration parallel genetic algorithm (PGA_MIG). This parallel genetic algorithm model allows genetic diversity by the application of selection and reproduction mechanisms nearer to nature. The space structure of the population is modified by dividing it into disjoined subpopulations. From time to time, individuals are exchanged between the different subpopulations (migration). Influence of parameters and dedicated strategies are studied. These parameters are the number of independent subpopulations, the interconnection topology between subpopulations, the choice/replacement strategy of the migrant individuals, and the migration frequency. A comparison between the sequential and parallel version of genetic algorithm (GA) is provided. This comparison relates to the quality of the solution and the execution time of the two versions. The efficiency of the parallel model highly depends on the parameters and especially on the migration frequency. In the same way this parallel model gives a significant improvement of computational time if it is implemented on a parallel architecture which offers an acceptable number of processors (as many processors as subpopulations).

Copyright © 2006 K. Belkadi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Sequential genetic algorithms (GAs) (see Algorithm 1.1) are stochastic research techniques first introduced by Holland in early seventies [6, 8]. They are inspired by the mechanisms of the biological evolution of the species and very much used for combinatorial optimization.

GA is an abstraction of the evolution process which is a strongly parallel process. So it is well designed to parallelization. But GA suffers from a major problem which is premature convergence. In order to prevent excessive convergence rate into suboptimal solution, several selection and replacement strategies were introduced to create a genetic diversity

Hindawi Publishing Corporation
Journal of Applied Mathematics and Decision Sciences
Volume 2006, Article ID 65746, Pages 1–17
DOI 10.1155/JAMDS/2006/65746

Begin
– Generate an initial population;
– Evaluate each individual of the population;
– Determine the best solution;
Repeat
– *Select* M individuals;
– *Crossover/Mutation*;
– Evaluate the resulting children;
– *Replacement* of the children;
– Determine the best solution
Until Stop Criterion reached
End

Algorithm 1.1. Simplified sequential genetic algorithm.

in the population. Parallel genetic algorithms appeared and in particular the parallel genetic algorithms with migration [4, 7].

In the parallel genetic algorithm model with migration (PGA_MIG), the population is divided into several subpopulations. The number of subpopulations should not be too significant to guarantee that in each subpopulation there exists sufficient number of individuals to be able to carry out suitably a local GA without risk to be promptly absorbed by a local minimum. Each processor executes, independently of the other processors, its sequential GA on its partition. From time to time exchanges of individuals are managed between the different subpopulations. This operation is denoted by *migration*. Subpopulations evolve in parallel and exchange the genes at the same time. This parallelization strategy is linked to several parameters which influence considerably on GA like the migration strategy, the migration frequency, the migration interval, and the interconnection topology between subpopulations [11].

In the following sections, we shortly describe the HFS problem. Then we apply the PGA_MIG to the HFS problem by giving the numerical experiments and the results obtained. A comparative study from the parallel GA version and sequential GA is provided. This comparison relates to the quality of the solution and the execution time of the two versions.

2. Presentation of hybrid flow shop

A hybrid flow shop (HFS) is a system composed of a set of stages, where each stage is composed of one or more parallel machines. The different jobs visit the stages in the same order. On each stage, a job is treated by one machine only. Between each stage, the jobs can wait or not in limited or unlimited buffers [3, 15].

For the experimental study, the choice was directed first on hybrid flow shop with 2 stages and 3 machines on the first stage and 2 machines on the second one (Figure 2.1). A buffer of infinite capacity is incorporated between stages of the system. Moreover all jobs are assumed to be available at the system entrance with release date with value 0 [1, 12].

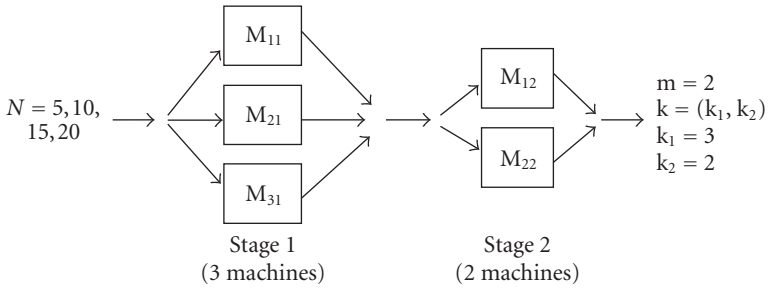


Figure 2.1. FH2 (P3, P2) || Cmax.

The problem consists in scheduling n jobs at the system entrance and the assignment of these jobs on the machines. The objective consists in scheduling (sequencing) jobs on machines and assigning each job on one machine at each stage in order to minimize a criterion of performance. The criterion to be optimized (to minimize here) is Cmax (the completion time of the last job on the last stage). By using the notation of Vignier [14], the system can be defined by FH2 (P3, P2) || Cmax.

3. Presentation of parallel genetic algorithm model with migration (PGA_MIG)

PGA_MIG consists in dividing the population into disjoined subpopulations and authorizing from time to time to exchange individuals between the different subpopulations (migration).

Algorithm PGA_MIG is implemented in master/slaves mode and on a biprocessor architecture [1].

Initially the master generates an initial population and divides it into P subpopulations (P: the number of slaves). Each slave executes sequential GA for a number N of iterations (Nbr_Iter_Mig), with N being the size of intervals which separates two migration operations (see Algorithm 3.1). At the point of migration, the master recovers the results of the research carried out by the different slaves (see Algorithm 3.2) and launches the migration operation which consists in copying genes of some individuals from one subpopulation towards one of its neighbor subpopulations [2, 11].

4. Numerical experiments

The algorithm of the PGA_MIG was implemented on a biprocessor architecture and it was tested on hybrid flow shop of the same type as that studied for sequential GA (see Section 2).

Sequential GA is based on a direct adaptation of the basic operators of GA for HFS problem [14] in particular in the process of the coding/decoding of the chromosome which is strongly linked to the problematic of HFS scheduling. The variables of decisions linked to the HFS are of two types. One is linked to the assignment problem and the other to the scheduling problem. The genome constitutes of two chromosomes, one for the assignment and the other for scheduling (sequencing). For the assignment, the

```
Begin
    – Generate the initial population: Pop_Ini;
    – Evaluate Pop_Ini;
    – Determine the best solution: Best_Global_Sol;
    – Global_Sol ← Best_Global_Sol;
    – Divide Pop_Ini into P subpopulations; (* P the number of slaves *)
    – Calculate the migration interval size: Nbr_Iter_Mig; (* iteration
      number to be made before a migration *)
    – Calculate the number of migrations to carry out Nbr_Mig;
    – Nbr ← 0; (* meter of the number of migrations, for the stop test *)
    – Non_ improv ← 0; (* meter of the number of generations without
      improvement, for the stop test *)
    – To launch the P slaves;
     Repeat
          – WaitFor (End_Of_Cycle j); (* Await the end of a treatment cycle
            before the migration for all the slaves *)
          – Suspend (P j) with j = 1,2,3,..., P; (* Suspend the execution
            of the slaves temporarily *)
          – Migration (Topology, Choice_Strat); (* Migration according to the
            topology and the choice strategy *)
          – Nbr ← Nbr +1;
           If H (Global_Sol) < H (Best_Global_Sol)
           Then No_improv ← 0;
           Else No_improv ← No_improv + 1;
           Endif
          – To begin again (P j);
    Until ((Nbr > Nbr_Mig) or (Non_improv > Max_Non_improv)) (* criterion
    of stop reached *)
End
```

Algorithm 3.1. Algorithm of the master for PGA_MIG.

chromosome is represented by a vector A (a vector for each stage) whose components materialize each work. Thus the A(i) number indicates the number of the machine (in the stage) witch performs task *i*. In the same way, for the sequencing problem, a matrix P contains the set of the precedence constraints between the operations of works at a given stage [14].

Example 4.1. We consider the hybrid flow shop (FH2, (P2, P1) || Cmax) with 5 jobs to be scheduled. The five jobs are J1, J2, J3, J4, and J5.

If jobs J1 and J3 are assigned in this order on machine 1 of the first stage and jobs J5, J2, and J4 are assigned in this order on machine 2, then the order on the first stage is J1,

```
Begin
    – Iter ← 1;
    – End_Of_Cycle j ← False;
    Repeat
        – Selection; (* the selection operates on local subpopulation P j *)
        – Crossover;
        – Mutation;
        – Replacement;
        – Iter ←Iter +1;
    Until (Iter > Nbr_Iter_Mig) (* max number of iterations to make
      a migration *)
    – Determine the best local solution: Best_Local_Sol;
    – Lock (Global_Sol);
    If  H (Global_Sol) > H (Best_Local_Sol)
    Then Global_Sol ← Best_Local_Sol
    End If
    – Unlock (Global_Sol);
    – End_Of_Cycle j ← True;
End
```

Algorithm 3.2. Algorithm of a slave P_j for PGA_MIG.

J3, J5, J2, and J4. We need 2 vectors ($A^{(1)}$ and $A^{(2)}$) for the assignment:

$$A^{(1)} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 2 \end{bmatrix} \quad A^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

We need 1 matrix P for the sequencing:

$$P1 = \begin{array}{c|ccccc} & J1 & J2 & J3 & J4 & J5 \\ \hline J1 & 0 & 2 & 1 & 2 & 2 \\ J2 & -2 & 0 & -2 & 1 & -1 \\ J3 & -1 & 2 & 0 & 2 & 2 \\ J4 & -2 & -1 & -2 & 0 & -1 \\ J5 & -2 & 1 & -2 & 1 & 0 \end{array}.$$

The value and the significances of the elements of P are indicated in Table 4.1.

The selection and the replacement are the two basic operators of GAs. The selection strategy used is the roulette wheel strategy [6] where the resulting children, and after being evaluated, replace certain individuals in the initial population and they can be selected several times according to their fitness value. The replacement strategy used is the

Table 4.1. Value and significances of the elements of P.

Value of P1(i, j)	Significances
1	If the operation l of job i precedes the operation of the job j on a machine
-1	If the operation l of job j precedes the operation of the job i on a machine
0	If i is equal to j
2	If the operation l of job i and the operation of the job i are carried out by two different machines and the operation l of job i starts to be carried out before the operation of job j
-2	If the operation l of job i and the operation of the job i are carried out by two different machines and the operation l of job i starts to be carried out after the operation of job j

Table 4.2. Average Cmax and average CPUs for sequential GA.

A number of jobs	Average Cmax	Average CPUs (s)
5	145,0	0.5
10	260.4	2
15	447.5	3
20	526.1	6

selective breading strategy [9], where a given child replaces an individual only if its fitness value is better than of the individual that to be replaced.

The crossover and mutation operators apply to the two chromosomes (assignment and scheduling) of the genome of any individual. The crossover used is a classical uniform crossover. The mutation operator is very similar to the crossover operator but it operates only on the sequencing (scheduling) chromosome by choosing a random machine and changing the order of a couple of the works assigned to this machine [14].

Concerning the parameters for sequential GAs which must be defined, their values are fixed and given as follows [12]: the population size is 200, the probability of crossover is 0.5, the probability of mutation is 0.005, and the number of iterations is 100.

Sequential GA applied to the HFS scheduling problem gave the results presented in Table 4.2 (the cost values of average Cmax and execution time). These values will be used as reference for a comparison between the sequential and parallel implementation of GA.

The PGA_MIG depends on several parameters: the number of subpopulations, the interconnection topology between subpopulations, the choice/replacement strategy of the migrant individuals, and the migration frequency.

We studied the influence of these parameters on quality of the solution obtained by the PGA_MIG.

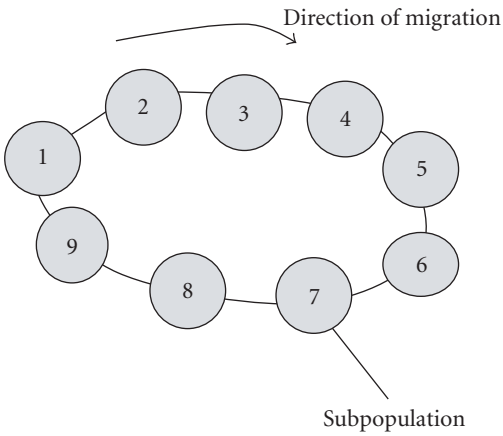


FIGURE 4.1. Ring topology.

4.1. Influence of interconnection topology and choice/replacement strategies on the results. (a) The notion of interconnection determines the neighboring of subpopulation which is the set of subpopulations which carried out the migration. In the implementation, we chose two interconnection topologies (or neighboring): ring topology and grid_2D topology [10].

– *Ring topology.* A logical interconnection or a logical neighboring is established between subpopulations. Each one of subpopulations is connected (logically) to two subpopulations, one as upstream and the other downstream. During a migration, a choice of direction of migration is established. This direction determines for each subpopulation his neighbor subpopulation. Thus it can send and receive its individuals. This direction of migration is respected by all subpopulations (Figure 4.1).

– *Grid topology with two dimensions (Grid_2D).* Each of subpopulations is connected with four other subpopulations The interconnection is a matrix with two dimensions (Figure 4.2). During a migration, each subpopulation chooses a population among its four direct neighbors. In this topology, the choice of direction is not imposed.

(b) The choice/replacement strategy of the migrant individuals determines which individuals are selected to migrate from the current subpopulation, and which individuals to replace in the next subpopulation. Two strategies have been investigated: random and best/bad [5].

– *Random.* the individuals who migrate from the current subpopulation are selected randomly and replace individuals chosen randomly in the next subpopulation.

– *Best/bad.* the individuals who migrate from the current subpopulation are selected among the best individuals, that is, the one with the greatest fitness value in the current subpopulation, and replace the worst individuals with the lowest fitness in the next subpopulation.

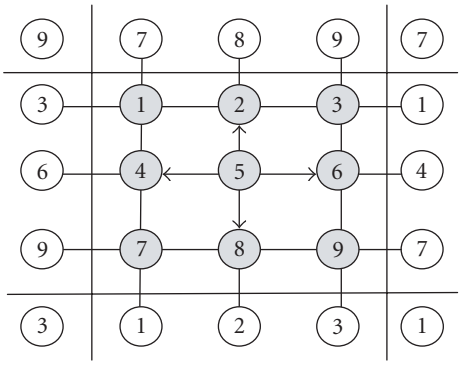


FIGURE 4.2. Grid_2D topology by taking account of the edges effect in the neighboring.

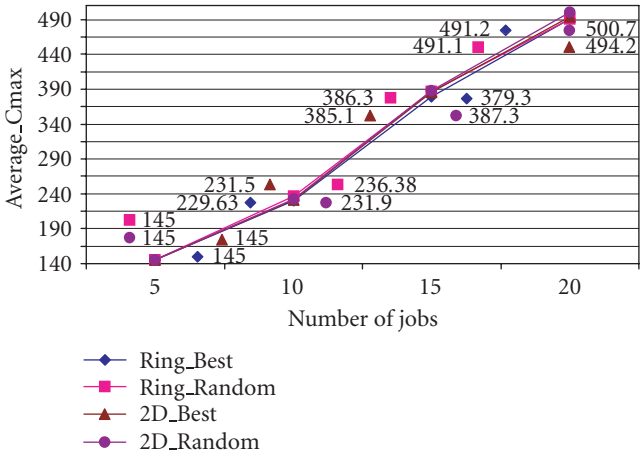


FIGURE 4.3. Variation of Cmax for different interconnection topologies.

(c) The two interconnection topologies of subpopulations (Ring, 2D) described previously were implemented for the PGA_MIG. The two choice/replacement strategies of the migrant individuals (Best/Bad, Random) were also implemented for each interconnection topology. Four mechanisms can be investigated by coupling the topology with the choice/replacement strategy: Ring _Random, Ring _Best, 2D_Random, and 2D_Best.

Figure 4.3 presents the variation of average Cmax as regards the different mechanisms implemented. This variation is for the different numbers of jobs taken in test ($N = 5, 10$, 15, and 20). The figure is composed of four series, each one is reserved for an interconnection mechanism given.

In this figure, it is possible to note that the influence of the interconnection topology is not significant in the variation of Cmax. Nevertheless ring topology is the topology which gave the greatest number of best results.

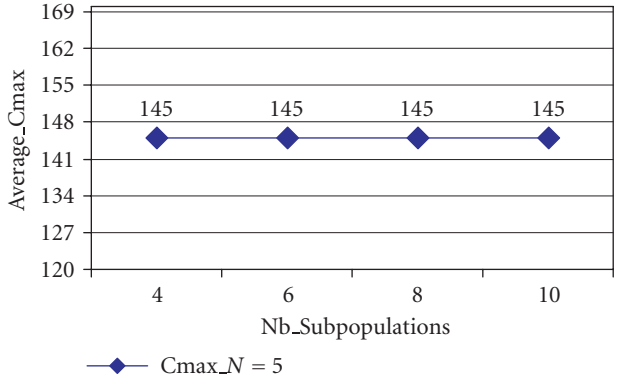


FIGURE 4.4. Cmax variation with the increase of the number of subpopulations for $N = 5$.

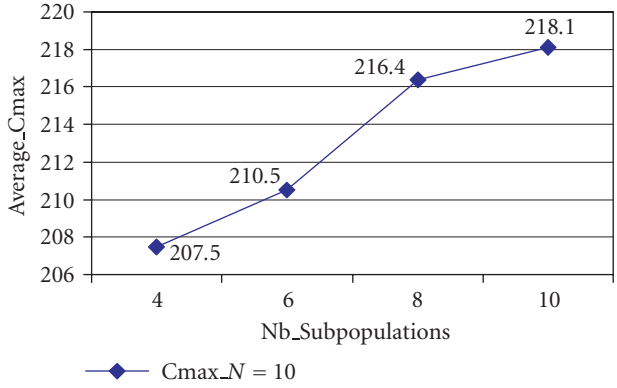


FIGURE 4.5. Cmax variation with the increase of the number of subpopulations for $N = 10$.

In what follows, we chose the ring interconnection topology, with the best choice/replacement strategy, for studying better the influence of the other different parameters.

4.2. Influence of number of subpopulations on the results. The number of subpopulations determines the size of subpopulation because the origin population is distributed between subpopulations. The more the number of subpopulations is high, the more the size of each population is reduced [4]. The influence of the number of subpopulations on the quality of solution is investigated. The number of subpopulation (slaves) varied from 4 to 10. To measure the influence of the number of subpopulations, a migration frequency equal to 90% is used.

Figures 4.4, 4.5, 4.6, and 4.7 represent the variation of average Cmax compared to the variation of the number of subpopulations. This variation is for the different number of jobs taken in test ($N = 5, 10, 15$, and 20).

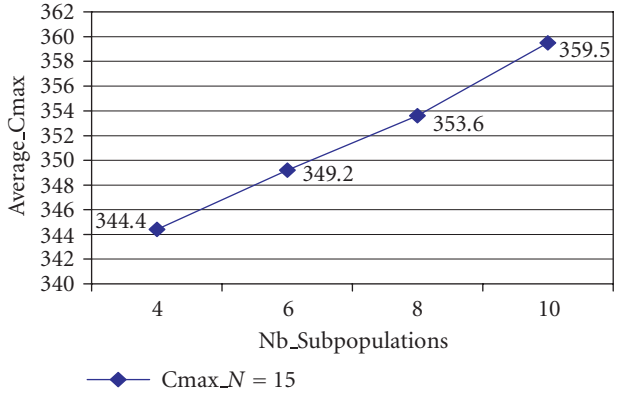


FIGURE 4.6. Cmax variation with the increase of the number of subpopulations for $N = 15$.

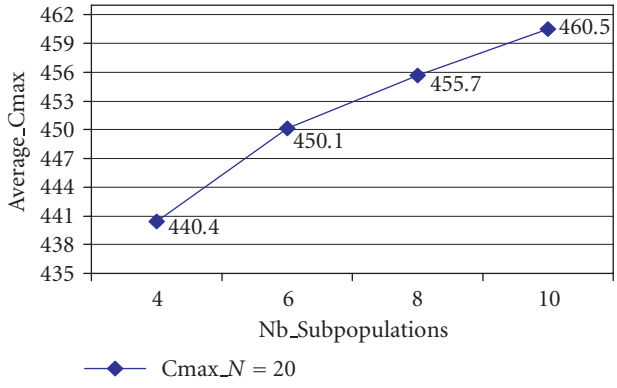


FIGURE 4.7. Cmax variation with the increase of the number of subpopulations for $N = 20$.

In these figures, one can note that the quality of the solution decreases progressively at the same time as the number of subpopulations increases. Indeed, the number of subpopulations has a direct influence on the genetic diversity of this parallel model (PGA_MIG). A great number of subpopulations implies a reduction of the number of individuals by subpopulation (reduction of the number of genes). Subpopulation becomes quickly homogeneous and is prevented from evolving to interesting subspaces of solutions.

One can also note that the gap between the upper bound and the lower bound in absolute terms is increasing but in relative terms it is stable with the number of studied jobs (increase in complexity of HFS system). Thus, the PGA_MIG with a significant number of subpopulations loses a little its genetic diversity as soon as the size of the space of solution becomes significant (see Figures 4.12(a) and 4.12(b)). We can say that the subpopulation number is not a parameter which influences truly the results.

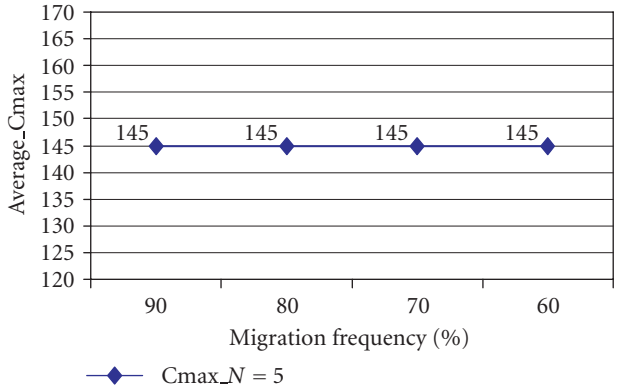


FIGURE 4.8. Cmax variation with the reduction of the migration frequency for $N = 5$.

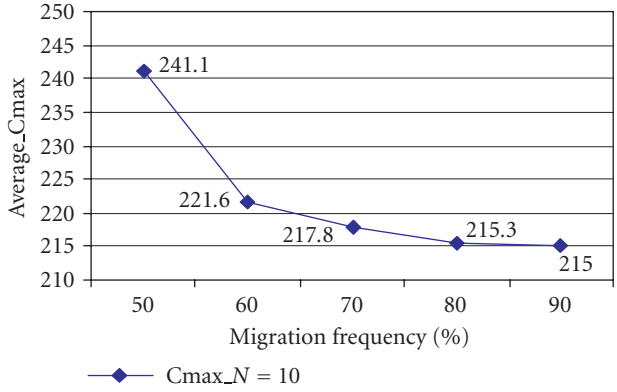


FIGURE 4.9. Cmax variation with the reduction of the migration frequency for $N = 10$.

4.3. Influence of migration frequency on results. The migration frequency gives information on the number of generations carried out in parallel by the different subpopulations between two migration operations, often it is noted by "migration interval." It represents the iteration interval which separates two migration operations [13]. To study the impact of this parameter, the migration frequency is successively assigned to 90%, 80%, 70%, 60%, and 50%. This frequency determines the number of migrations carried out during the execution of algorithm PGA_MIG. It determines also the migration interval which separates two migration operations. We studied the impact of this parameter for the HFS problem taken in example and the results are given by Figures 4.8, 4.9, 4.10, and 4.11. Each figure encompasses a series which represent the Cmax variation as regards the migration frequency variation. The number of subpopulations used is 4.

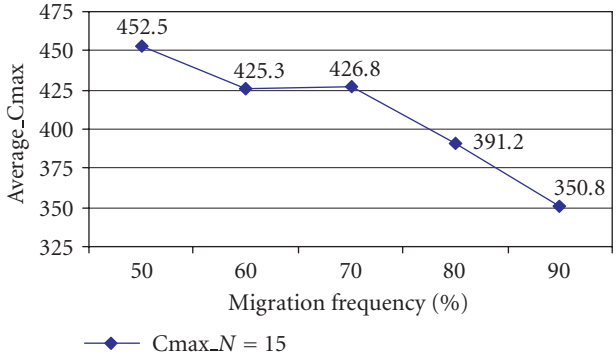


FIGURE 4.10. Cmax variation with the reduction of the migration frequency for $N = 15$.

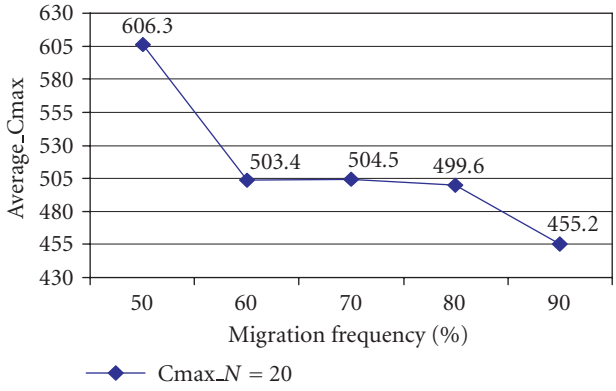
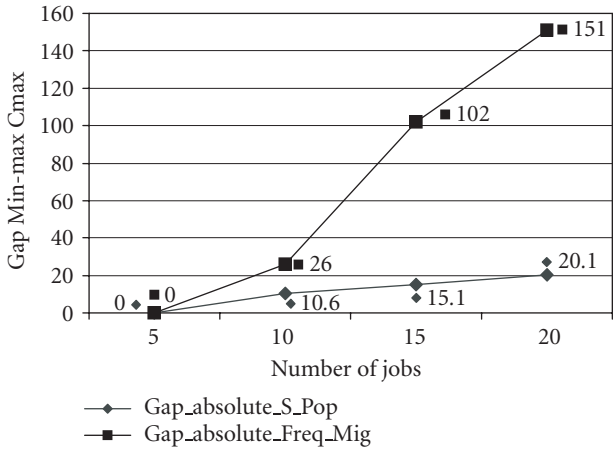


FIGURE 4.11. Cmax variation with the reduction of the migration frequency for $N = 20$.

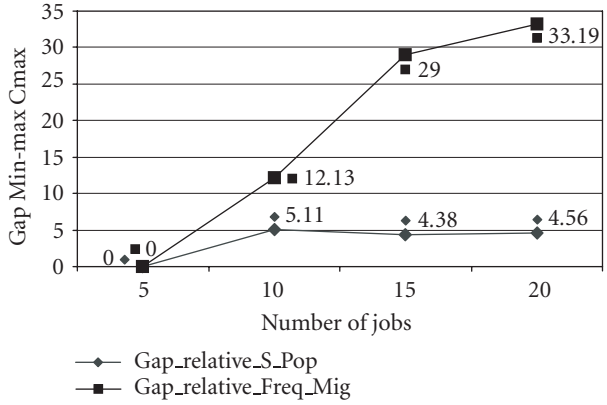
In these figures, we notice first that the quality of the solution tries to improve gradually with the migration frequency. One can accept that subpopulations strongly increase the GA performance of HFS problem. Subpopulations receive foreign individuals with the genetic history of origin subpopulations.

Let us note that the interval [60%, 80%] for the migration frequency gives a linear decrease. It is the stability interval for the algorithm. For the value 90%, a considerable improvement is noted for all the figures, but the value 50% gives the weakest quality solutions.

Concerning the influence of the migration frequency, the gap between the upper bound and the lower bound in absolute and relative terms is increasing with the increase in the complexity of the HFS. That shows the influence of this parameter on the solution quality. Also it should be noted that this gap is relatively significant compared with that obtained by varying the number of subpopulations.



(a)



(b)

FIGURE 4.12. (a) Influence of subpopulations number and the migration frequency on the absolute gap between the bounds of Min-max Cmax, (b) influence of subpopulations number and the migration frequency on the relative gap between the bounds of Min-max Cmax.

Figure 4.12(a) represents the influence of the subpopulation number (Gap_absolute_S_Pop) and the migration frequency (Gap_absolute_Freq_Mig) on the absolute gap between the bounds of Min-max Cmax. Figure 4.12(b) represents the influence of the subpopulation number (Gap_relative_S_Pop) and the migration frequency (Gap_relative_Freq_Mig) on the relative gap between the bounds of Min-max Cmax.

We can say that the migration frequency is a parameter which influences the results of the PGA_MIG but on the other hand we can say that the subpopulation number is not a parameter which influences truly the results.

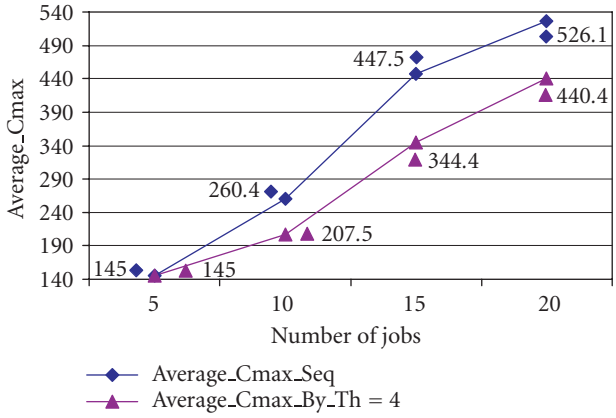


FIGURE 5.1. Comparison of the quality of solution between sequential GA and the PGA_MIG with 4 subpopulations.

5. Comparison between the PGA_MIG and sequential GA

In this section analysis, we present a comparison between the performances of the PGA_MIG and sequential GA. This comparison relates to the quality of the solution and the execution time of the two versions.

(a) Concerning the performances on the *quality of solution* of the PGA_MIG, they are represented in Figure 5.1. This figure is composed of two series of data: the series of the sequential GA results and the series of the PGA_MIG results, obtained with 4 Threads (indeed the number of four subpopulations was the number which gave good results in term of quality of solution) and the migration frequency is taken to 90%.

According to this figure, the Cmax obtained by the PGA_MIG is always better compared with sequential GA. This result remains valid while increasing the HFS instance scale (number of jobs). This phenomenon is highlighted by the progressive divergence between the two curves in the preceding figure.

(b) Concerning the performances in *execution time* of the PGA_MIG, theoretically, this model must lead to a considerable acceleration if subpopulations are distributed at a rate of subpopulation by processor. Indeed, the architecture chosen in our implementation does not permit to have such an acceleration. Nevertheless the estimation of the time taken by each Thread (each Thread is assigned to one subpopulation) gives us information on the acceleration which can lead this implementation if it is executed on an adequate multiprocessors architecture.

Table 5.1 represents CPUs time taken by sequential GA and CPUp time taken by the PGA_MIG executed with 10 subpopulations (10 Threads). Threads execute each one with the same number of generations as sequential GA; therefore, 10 times more generations but on a population size reduced by a factor of 1/10 for each Thread. CPUp time is the time which the two processors of architecture take to lead to the solution. Each processor is thus responsible to evaluate 5 Threads. By dividing CPUp time on the 5 Threads, we will have the time taken by each Thread in the execution of the algorithm on its

Table 5.1. Data for Figure 5.2.

Number of jobs	CPUs	CPUp	CPUp_1Th
5	0.5	2	0.4
10	2	4	0.8
15	3	6	1.2
20	6	11	2.2

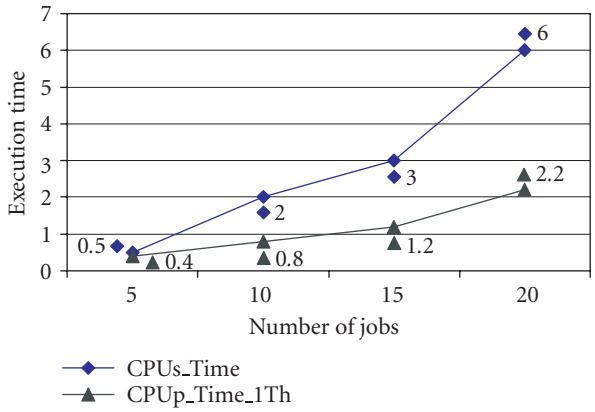


Figure 5.2. Comparison of the execution time of sequential GA and parallel GA.

subpopulation. Note that the time of migration is distributed equitably between different Threads. The time taken by each Thread is noted CPUp_Th and it is also represented in Table 5.1. The time taken by sequential GA (CPUs) is represented in Figure 5.2. And in the same way the time taken by a Thread of the PGA_MIG executed on 10 subpopulations is represented in Figure 5.2.

We clearly notice the acceleration which this parallelization model can reach. This acceleration is not due to the reduction of the number of generations compared to sequential GA, since this model executes P times the number of generations carried out by sequential GA (P: number of subpopulations), but rather to the reduction of the number of individuals treated by each Thread (size of subpopulations). This reduction results from the distribution of the origin population on different Threads.

6. Conclusion

The goal of the article is the resolution of the hybrid flow shop production systems scheduling problem by using a parallel genetic algorithm. The implementation investigated is based on the parallel genetic algorithms model with migration (PGA_MIG).

The parallelization model used for the genetic algorithms, which is migration (PGA_MIG), proved its capability and adequacy in the resolution of the HFS scheduling problem. It shows also the efficiency of the population distribution in the resolution of the premature convergence problem.

Indeed parallelization according to the migration model strongly improves quality of the solutions, compared as regards sequential GA. The efficiency of this parallel model highly depends on parameters including but not limited to the number of independent subpopulations, the interconnection topology between subpopulations, the choice/replacement strategy of the migrant individuals, and the migration frequency. The migration frequency maintains the link between the different subpopulations and guarantees the aspect of the single population which is distributed in distinct subpopulations.

This parallel model (PGA_MIG) gives an acceleration if it is implemented on a parallel architecture which offers an acceptable number of processors. Indeed, in this model an acceleration is significant if the parallel architecture contains as many processors as of subpopulations and if the communication cost is low.

This resolution of the HFS by the PGA_MIG was not treated in the literature. In the literature, the authors have used another parallel model for the resolution of this problem (HFS) or they solved another problem with this parallel model (PGA_MIG).

Our research is now directed

(i) to an implementation of our proposed model in a grid computing to obtained a fully parallel implementation;

(ii) to a full evaluation of the proposed model with previous published one including but not limited to parallel taboo search.

References

[1] A. Aribi and K. Belkadi, *Parallel computing and metaheuristics*, The 2nd International Workshop on Advanced Computation for Engineering Application (ACEA '03), Electronic Research Institute, Cairo, 2003.

[2] M. Babbar and B. S. Minsker, *A multi-scale master-slave genetic parallel algorithm with application to groundwater remediation design*, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02), Morgan Kaufmann, New York, 2002.

[3] J. C. Billaut, A. Vignier, C. Proust, and M. C. Portmann, *A survey of hybrid flowshop problems*, Ordonnancement Déterministe pour l'Informatique et la Production (ODIP '00), Aussois, 2000, Ecole d'été.

[4] E. Cantü Paz, *Markov chain models of parallel genetic algorithms*, IEEE Transactions on Evolutionary Computation **4** (2000), no. 3, 216–226.

[5] A. E. Eiben, I. G. Sprinkhvizen-Kuyper, and B. A. Thijssen, *Competing crossovers in an adaptative GA framework*, Proceedings of the 5th IEEE Conference on Evolutionary Computation, IEEE Press, New York, 1998, pp. 787–792.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Kluwer Academic, Massachusetts, 1989.

[7] M. Heijligers, *The application of genetic algorithms to high-level synthesis*, Ph.D. dissertation, Eindhoven University of Technology, Eindhoven, 1996.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Michigan, 1975.

[9] D. A. Linkens and H. O. Nyongesa, *Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications*, IEE Proceedings: Control Theory and Applications **143** (1996), no. 4, 367–386.

[10] T. Matsuzawa, *Asynchronous island parallel GA using multiform subpopulations*, Proceedings of International Symposium on Firth Generation Computer, 1996.

[11] M. Nowostawski, *Parallel genetic algorithm in geometry atomic cluster optimization algorithm and other applications*, Master's thesis, School of Computer Science, University of Birmingham, Birmingham, 1998.

[12] M. Sahraoui, *Optimization methods applied to the hybrid flow shop production systems*, Thesis of Magister, Department of Computer Science, USTO, Oran, 2002.

[13] T. Shisam, *Comparison between synchronous and asynchronous implementation of parallel genetic programming*, IEEE International Symposium on Intelligent Signal Processing and Communication Systems, Hawaii, November 2000.

[14] A. Vignier, *Contribution to the resolution of scheduling problems of the single range type, multi-machines HSF*, Ph.D. dissertation, University of Tours, Cedex, 1997.

[15] A. Vignier, J.-C. Billaut, and C. Proust, *Scheduling problems of hybrid flowshop type: state of the art*, RAIRO. Operations Research **33** (1999), no. 2, 117–183.

K. Belkadi: Department of Computer Science, University of Sciences and Technology of Oran Mohamed Boudiaf, BP 1505 Oran M'Naouer, Oran 31000, Algeria
Current address: LIMOS Laboratory, University of Blaise Pascal, Clermont Ferrand, Campus of Cézeaux, 63173 Aubière Cedex, France
E-mail address: belkadi@univ-usto.dz

M. Gourgand: LIMOS Laboratory, University of Blaise Pascal, Clermont Ferrand, Campus of Cézeaux, 63173 Aubière Cedex, France
E-mail address: gourgand@isima.fr

M. Benyettou: Department of Computer Science, University of Sciences and Technology of Oran Mohamed Boudiaf, BP 1505 Oran M'Naouer, Oran 31000, Algeria
E-mail address: mbenyettou@univ-usto.dz