New load balancing strategy for solving Permutation Flow Shop Problem Using Grid Computing

Samia Kouki[1], Talel Ladhari [2,3] and, Mohamed Jemni[1]

[1] High School of Sciences and Technics of Tunis, Research Laboratory LATIC.
Tunis, Tunisia.
Samia.kouki@esstt.rnu.tn mohamed.jemni@fst.rnu.tn
[2] ROI – Combinatorial Optimization Research Group, Polytechnic School of Tunisia, B.P. 743, 2078, La Marsa, Tunisia
[3] Princess Fatimah Al-Nijriss Research Chair for Advanced Manufacturing Technologies, Department of Industrial Engineering, College of Engineering, King Saud University, PO Box 800 Riyadh 11421, Saudi Arabia.
talel_ladhari2004@yahoo.fr

Abstract. This paper describes two parallel algorithms solving the m-machines, n-jobs permutation flow shop scheduling problem (PFSP) deployed in the French national grid Grid'5000. The first algorithm called *GAUUB* uses the Branch and Bound method to find optimal solutions of the problem and distributes the tasks among all processors. The second algorithm called *GALB* is an improvement of the *GAUUB* algorithm based on a parallelization strategy ensuring better load balancing between the processors and therefore, better efficiency of the algorithm. Computational results of the *GAUUB* algorithm performed on the grid showed good results and significant improvements of our initial algorithm thanks to our load balancing technique.

1 Introduction

The optimization of scheduling problems can be based on different criteria to optimize, but the mainly target used in the literature is the minimization of completion time of the last task on the last machine called makespan. In this paper, we deal with the permutation flow shop scheduling problem (PFSP) which is a combinatorial optimization problem. In this problem we have to determine the best solution for scheduling ***n*** jobs in ***m*** machines, which minimized the total completion time (the Makespan) of the schedule (C_{max}) this problem is denoted ($F||C_{max}$) and is known to be strongly NP-hard. The most known exact method for solving such a problem is the Branch and Bound. Nevertheless, this method is limited, as it can solve only small size instances. Because of the large size of handled problems, finding an optimal

solution using a single machine can be impossible for some instances of data. Thus, there exist two approaches which can help us to solve large size instances in an acceptable time: approximate methods and parallel methods. Many previous work deals with the parallel Branch and Bound method as reported in [1] by Crainic and all. Thanks to the huge number of resources provided by the grid of computers, it seems to be a suitable architecture to solve such kind of problems (strongly NP-hard problems) and especially the permutation flow shop problem, as it still exists many instances of Taillard which are not yet solved [2].

The high computing time of the PFSP solved by B&B algorithm is due to the computation of bounds which accomplished at each node of the search tree. The size of this tree is huge and can attempt several billions of nodes. Furthermore, not all combinatorial optimization problems can be parallelized, since this depends strictly on the nature and the characteristics of the problem to treat, and in particular on the quality of the upper and lower bounds.

2 Our first algorithm : Grid Algorithm Updating the Upper Bound (GAUUB)

Our implemented algorithm is based on updating the upper bound value in order to make the algorithm converging rapidly to the optimal solution. In this paper we use the depth first strategy. Our parallelization strategy is based on dividing the PFSP to many sub-problems to be performed independently and simultaneously by different processors [3]. However, the parallel execution of the B&B solving the PFSP, can lead to the treatment of some needless sub problems. Especially, when sub problems are treated in parallel, although some of these problems may be pruned in sequential execution. For this reason, in our algorithm we used an intelligent technique to distribute and update the upper bound values in order to reduce the size of the search tree and therefore to obtain as quickly as possible the optimal solution.

We implemented our algorithm with C language; we used also the MPI library (Message Passing Interface) in order to ensure communication between processors. All the computational results were carried on the French national Grid (Grid'5000) [4]. In order to test and improve the scalability of our algorithms, we used the well known Taillard's benchmarks. These instances are known to be very hard and about a decade after their publication, many of them are still open (instances with 20

machines and some others with 10 machines). We used also these instances in our previous works [5,6].

The experimentation study of our algorithm *GAUUB*, shows clearly the contribution of parallelism to improve execution times of many hard instances which have not been solved in single processor. But by analyzing many results obtained in our experimental study we noticed that when several nodes are pruned from the first level of the search tree, the use of parallelism is not very significant. However, when the processors share the workload that is normally assigned to a single processor, we can achieve reasonable running times for some instances. Based on such observations, we propose in the next section our new algorithm called Grid Algorithm with Load Balancing (*GALB*), in order to remedy the unbalancing load of processors in the *GAUUB* algorithm.

3 Our new algorithm : Grid Algorithm with Load Balancing (GALB)

This algorithm called Grid Algorithm with Load Balancing (GALB), is an improvement of our previous algorithm (GAUUB) by establishing an efficient load balancing technique between all the available processors. This algorithm is based on the master/slave paradigm and is composed of two main steps. The first step is done by the master processor and is performed in serial until a level L of the search tree. The goal of this step is to generate a large amount of work (many nodes to explore) to distribute among the slaves processors and then to warranty a load balancing between all processors. The master assigns the unexplored nodes to the processors, ordered by the ranks of processors. Then all processors execute a local B&B.

Furthermore, the second step is accomplished in parallel, by all the available processors (slaves). Each processor will give the new solution to the master in order to update the global Upper bound.

Once a processor completes the task assigned to it, it requests data from the master and so on. Indeed, in the algorithm GALB, we note that the processors finish at almost the same time and therefore, that means that all processors participate in solving the problem during almost the same duration. The use of the GALB algorithm gives two advantages, first distributing the load across the majority of processors and

second reduce the execution time, which may allow us to solve more complicated and hard instances of data.

Experimental study of our new algorithm deployed on the National French Grid: Grid 5000 confirmed the improvement of load balance of our new algorithm and therefore the improvement of its performance. For instance, the execution time of the the instance Tail0106 on 50 processors using the *GAUUB* algorithm is about 105 seconds whereas the resolution of the same instance by the *GALB* algorithm using 50 processors requires only 14 seconds as showed in **Fig 1**.

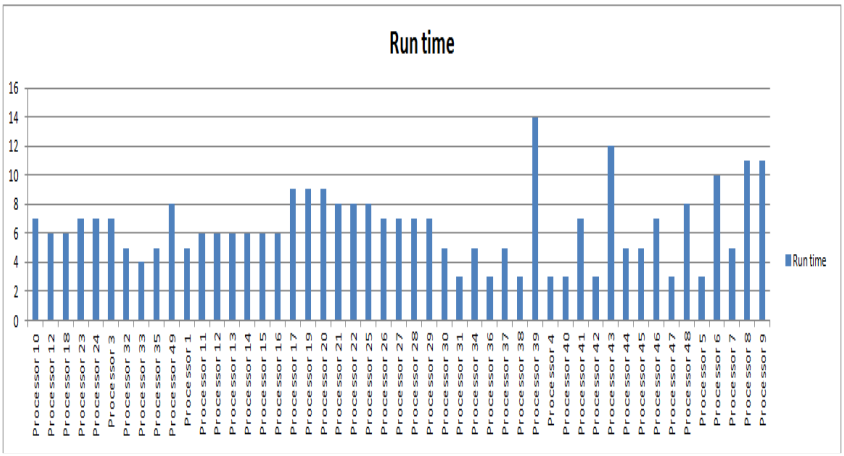

Fig. 1. Charge of the processors in the GALB algorithm (Tail0106)

References

1. Gendron, B., Crainic, T. G. : Parallel B&B Algorithms: Survey and synthesis. Operation Research, Vol. 42, No. 6, pp. 1042--1066, November-December (1994).
2. Taillard, E. :Benchmarks for basic scheduling problems. European Journal of Operational Research, vol. 64, pp. 278--285, (1993).
3. Kouki, S., Jemni, M., Ladhari, T. l: Deployment of Solving Permutation Flow Shop Scheduling Problem on the Grid. Conference on Grid and Distributed Computing (GDC) 2010, LNCS, Springer, December 13-15, 2010, Jeju Island, Korea.
4. www.grid5000.fr/
5. Kouki, S., Jemni, M., Ladhari, T.: Design of parallel distributed algorithm for the Permutation Flow Shop Problem. Conférence internationale sur les NOuvelles TEchnologies de la REpartition (NOTERE) IEEE, pp : 65-72, 31 May-2 June, 2010, Tozeur, Tunisia.
6. Kouki, S., Jemni, M., Ladhari, T.: A Parallel Distributed Algorithm for the Permutation Flow Shop Scheduling Problem. International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), LNCS, Springer, pp 328-337, May 21-23, 2010, Busan, Korea.
7. Kouki, S., Ladhari, T., Jemni, M., :Solving the Permutation Flow Shop Problem with Makespan Criterion using Grids. International Journal of Grid and Distributed Computing Vol. 4, No. 2, June, 2011.