Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 2020 – 2023

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Solving the Flexible Job Shop Problem on Multi-GPU

Wojciech Bożejko[1], Zdziław Hejducki[2], Mariusz Uchroński[3], Mieczysław Wodecki[4]

[1] *Institute of Computer Engineering, Control and Robotics Wrocław University of Technology*
Janiszewskiego 11-17, 50-372 Wrocław, Poland
wojciech.bozejko@pwr.wroc.pl
[2] *Institute of Construction Wrocław University of Technology*
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
[3] *Wrocław Centre of Networking and Supercomputing*
Wyb. Wyspańskiego 27, 50-370 Wrocław, Poland
[4] *Institute of Computer Science, University of Wrocław*
Joliot-Curie 15, 50-383 Wrocław, Poland

Abstract

We propose the new framework of the distributed tabu search metaheuristic designed to be executed using a multi-GPU cluster, i.e. cluster of nodes equipped with GPU computing units. We propose a hybrid single-walk parallelization of the tabu search, where hybridization consists in examining a number of solutions from a neighborhood concurrently by several GPUs (multi-GPU). The methodology is designed to solve the flexible job shop scheduling problem, difficult problem of discrete optimization.

Keywords: scheduling, flexible job shop, metaheuristics
PACS: 02.60.Pn, 02.70.-c
2000 MSC: 90B36, 90C35

1. Introduction

The paper deals with the flexible job shop problem, which can be briefly presented as follows. There is a set of jobs and a set of machines. Each job consists of a number of operations, which are to be processed in a given order, each on a machine from a set of dedicate machines. The processing of an operation can not be interrupted. Each machine can process at most one operation at a time. We want to find the schedule (the assignment of operations to time intervals on machines) that minimizes the makespan ($C_{\max}$). The problem, constitutes a generalization of the classic job shop problem. The job shop scheduling problem, although relatively easily stated, is strongly NP-hard, and is considered one of the hardest problems in the area of combinatorial optimization.

Many various methods have been proposed, ranging from simple and fast dispatching rules to sophisticated branch-and bound algorithms. Hurink [7] developed the tabu search method for this problem. Many authors have proposed a method of assigning operations to machines and then determining sequence of operations on each machines. Such an approach is followed by Pauli [8]. Also genetic approaches have been adopted to solve the flexible job shop problem ([9]). Gao *et al.* [6] proposed the hybrid genetic and variable neighborhood descent algorithm for this problem. We present new properties and techniques which allows us to solve the flexible job shop instances with high accuracy in a relatively short time. This paper constitutes a continuation of researches presented in the papers Bożejko [3], Bożejko *et al.* [4],[5].

doi:10.1016/j.procs.2012.04.227

2. Problem formulation and preliminaries

The flexible job shop problem can be formally defined as follows (see Bożejko *et al.* [4]). Let $\mathcal{J} = \{1, 2, \ldots, n\}$ be a set of jobs which have to be executed on machines from the set $\mathcal{M} = \{1, 2, \ldots, m\}$. There exists a partition of the set of machines into types, i.e. subsets of machines with the same functional properties. A job constitutes a sequence of some operations. Each operation has to be executed on an adequate type of machine (nest) within a fixed time. The problem consists in the jobs allocation to machines from the adequate type and the schedule of jobs execution determination on each machine to minimize the total jobs finishing time. The following constrains have to be fulfilled:

(*i*) each job has to be executed on only one machine of a determined type in each moment of time,
(*ii*) machines must not execute more than one job in each moment of time,
(*iii*) there are no idle times (i.e. the job execution must not be broken) and the technological order has to be obeyed.

Let $\mathcal{O} = \{1, 2, \ldots, o\}$ be the set of all operations. This set can be partitioned into sequences which correspond to jobs where the job $j \in \mathcal{J}$ is a sequence of o_j operations which have to be executed in an order on dedicated machines (i.e. in so-called technological order). Operations are indexed by numbers $(l_{j-1} + 1, \ldots, l_{j-1} + o_j)$ where $l_j = \sum_{i=1}^{j} o_i$ is the number of operations of the first j jobs, $j = 1, 2, \ldots, n$, where $l_0 = 0$ and $o = \sum_{i=1}^{n} o_i$.

The set of machines $\mathcal{M} = \{1, 2, \ldots, m\}$ can be partitioned into q subsets of the same type (*nests*) where i-th $(i = 1, 2, \ldots, q)$ type $\mathcal{M}^i$ includes m_i machines which are indexed by numbers $(t_{i-1}+1, \ldots, t_{i-1}+m_i)$, where $t_i = \sum_{j=1}^{i} m_j$ is the number of machines in the first i types, $i = 1, 2, \ldots, q$, where $t_0 = 0$ and $m = \sum_{j=1}^{q} m_j$.

An operation $v \in \mathcal{O}$ has to be executed on the machines type $\mu(v)$, i.e. on one of the machines from the set (nest) $\mathcal{M}^{\mu(v)}$ in the time $p_{v,j}$ where $j \in \mathcal{M}^{\mu(v)}$.

Let $\mathcal{O}^k = \{v \in \mathcal{O} : \mu(v) = k\}$ be a set of operations executed in the k-th nest. A sequence of operations sets $Q = [Q^1, Q^2, \ldots, Q^m]$, such that $\mathcal{O}^k = \bigcup_{i=t_{k-1}+1}^{t_{k-1}+m_k} Q^i$ $(k = 1, 2, \ldots, q)$ and $Q^i \cap Q^j = \emptyset$ $(i \neq j,\ i, j = 1, 2, \ldots, m,)$ we call an *assignment of operations from the set $\mathcal{O}$ to machines from the set $\mathcal{M}$*. A sequence $[Q^{t_{k-1}+1}, Q^{t_{k-1}+2}, \ldots, Q^{t_{k-1}+m_k}]$ is an *assignment* of operations to machines in the i-th nest.

Let $\pi(Q) = (\pi_1(Q), \pi_2(Q), ..., \pi_m(Q))$ be a concatenation, where $\pi_i(Q)$ is an permutation of operations executed on the machine i. Any feasible solution is a pair $(Q, \pi(Q))$ where Q is an assignment of operations to machines and $\pi(Q)$ is a permutations concatenation determining the operations execution sequence which are assigned the each machine fulfilling constrains (*i-iii*).

3. Distributed tabu search method

Currently, tabu search is one of the most effective methods using local search techniques to find near-optimal solutions to combinatorial optimization problems. The neighborhood of a basic processing order is generated by the moves. A move changes the location of some operations in the basic processing order. In order to avoid cycling, becoming trapped at a local optimum, or continuing the search in a too narrow region, the mechanisms of a tabu list and a perturbation are utilized. The tabu list records the performed moves, for a chosen span of time, treating them as forbidden for possible future moves. The search stops when a given number of iterations has been reached without improvement of the best current makespan, the algorithm has performed a given total number of iterations, time has run out, the neighborhood is empty, or a processing order with a satisfying makespan has been found, etc.

An idea of parallelization of the tabu search method was discussed in the work of Alba [1], mostly for multiple-walk, distributed parallelizations with cooperation. Here we propose to extend this application by using multiple GPUs as concurrent running threads directed by a master process. We use MPI library for the communication implementation. Using a starting solution, a number of neighbors are generated and assigned to several GPUs (see Fig.1). The method of solutions broadcasting is described in the next section. After a number of iterations, the best solutions generated by GPU threads are collected by the CPU and the best of them is chosen. This finishing the single cycle and the computations are repeated: a number of neighbors are generated and assigned to GPUs, etc.

4. Broadcasting in distributed computing environment

In this paper we propose a method of solving difficult discrete optimization problems in the distributed computing environments, such as multi-GPU clusters. Tabu search algorithm is executed in concurrent working threads, as in

multiple-walk model of parallelization (see Alba [1]). Additionally, MPI library is used to distribute calculations among GPUs (see Fig.1). Now let us consider a single cycle of the MPI data broadcasting, multi-GPU computations

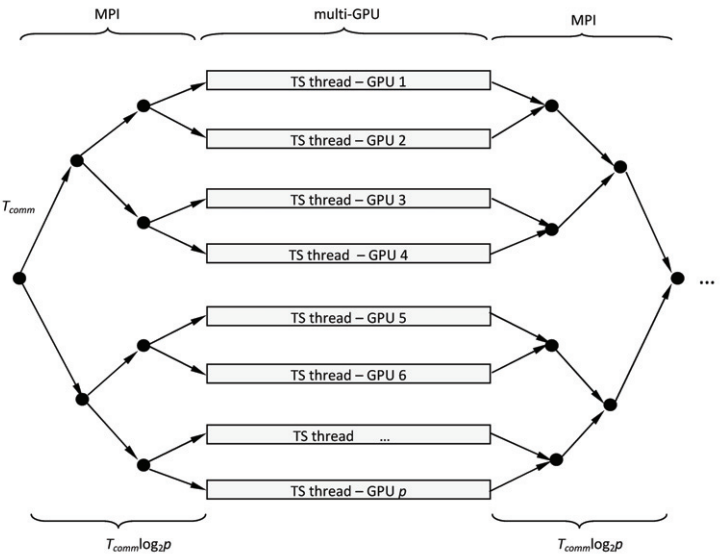

Figure 1: Skeleton of a single cycle of the Multi-Level Tabu Search metaheuristic.

and batching up of the results obtained. Let us assume, that the single communication procedure between two nodes of a cluster takes the time T_{comm}, the time of sequential tabu search computations is T_{seq} and the time of parallel is $T_{calc} = \frac{T_{seq}}{p}$ (p is the number of GPUs). Therefore, the total parallel computations time of the single cycle is

$$T_p = 2T_{comm}\log_2 p + T_{calc} = 2T_{comm}\log_2 p + \frac{T_{seq}}{p}.$$

In the case of using more processors, the parallel computing time $\left(\frac{T_{seq}}{p}\right)$ decreases, whereas the time of communication ($2T_{comm}\log p$) increases. We are looking for such a number of processors p (let us call it p^*) for which T_p is minimal. Calculating $\frac{\partial T_p}{\partial p} = 0$ we obtain

$$\frac{2T_{comm}}{p\ln 2} - \frac{T_{seq}}{p^2} = 0 \text{ and then } p = p^* = \frac{T_{seq}\ln 2}{2T_{comm}},$$

which provides us with an optimal number of processors p^* which minimizes the value of the parallel running time T_p.

5. Computational experiments

Parallel MPI+CUDA algorithm for the flexible job shop problem was coded in C (CUDA) and MPI library. Proposed algorithm was ran on the Tesla S2050 GPU and tested on the benchmark problems taken from [2]. We compare our results with results obtained of the hGA [6] algorithm of Gao *et al*. In Table 1 particular column means:

- *flex* – the average number of equivalent machines per operation,
- *PRD* – Percentage Relative Deviation to reference solution given by the formula $PRD = \frac{F_{ref}-F_{alg}}{F_{ref}} \cdot 100\%$, where F_{ref} is reference criterion function value and F_{alg} is the result obtained by the examined algorithm,
- s – speedup value given by the following formula $s = \frac{T_{seq}}{T_{par}}$, where T_{seq} is time of calculations of sequential algorithm and T_{par} is time needed by the parallel algorithm to reach cost function value equals or better than cost function value obtained by sequential algorithm.

instance			Percentage Relative Deviation *PRD*[%]			Speedup *s*	
name	$n \times m$	*flex*	1 processor	2 processors	4 processors	2 processors	4 processors
Mk01	10×6	2.09	2.50	2.50	0.00	17.51	11.21
Mk02	10×6	4.10	13.79	0.00	0.00	2.70	3.35
Mk03	15×8	3.01	0.00	0.00	0.00	3.28	2.29
Mk04	15×8	1.91	8.33	10.00	8.33	-	4.21
Mk05	15×4	1.71	0.58	0.58	0.58	0.36	0.25
Mk06	10×15	3.27	17.24	24.13	17.24	0.55	0.50
Mk07	20×5	2.83	4.31	3.59	3.59	-	2.24
Mk08	20×10	1.43	5.35	3.63	0.19	0.54	1.72
Mk09	20×10	2.53	33.55	33.55	9.77	2.27	2.27
Mk10	20×15	2.98	30.96	22.35	22.33	2.46	2.64
	average		11.66	10.03	6.37		

Table 1: Experimental results of the proposed approach.

The Table 1 presents values of average percentage relative deviation for sequential and parallel algorithm (for 2 and 4 processors). The proposed parallel tabu search algorithm with GPU acceleration managed to obtain the average relative percentage deviation from the best known solution of the Brandimarte instances on the level of 10.03% for two processors and 6.37% for four processors of the hGA [6] algorithm of Gao *et al.* Obtained results show that increasing the number of processors results in lower value of average percentage relative deviation. The Table 1 gives also values of parallel algorithm speedup. Value of the speedup was determined in the following way. Sequential version of the algorithm has been executed for the fixed number of iteration (*iter* = 10000) and its time of calculation was measured. After that parallel algorithm (for 2 and 4 processors) has been executed and works until the value of cost function found by the sequential algorithm has been reached. For most of the test instances value of speedup is greater than one.

6. Conclusions

We propose the new approach to solving flexible job shop problem by the hybrid single-walk distributed tabu search method, applied to be executed in the multi-GPUs environment. It was possible to approximate the optimal number of the environment processors, on which the algorithm has to be executed. Computational experiments shows a high efficiency of the proposed methodology.

Acknowledgement

The work was partially supported by the Polish Ministry of Science and Higher Education, within the grants no. N N514 232237 (W. Bożejko, M. Wodecki) and no. B10010 (M. Uchroński).

References

[1] Alba E., Parallel Metaheuristics. A New Class of Algorithms, Wiley & Sons Inc. (2005).
[2] Brandimarte P., Routing and scheduling in a fexible job shop by tabu search, Annals of Operations Research 41 (1993), 157-183.
[3] Bożejko W., On single-walk parallelization of the job shop problem solving algorithms, Computers & Operations Research 39 (2012), 2258-2264.
[4] Bożejko W., Uchroński M., Wodecki M., Parallel estimation of the cost function for the flexible scheduling problem, Proceedings of the ICCS 2011, Procedia Computer Science 4 (2011), Elsevier, 2236-2245.
[5] Bożejko W., Uchroński M., Wodecki M., Parallel hybrid metaheuristics for the flexible job shop problem, Computers & Industrial Engineering 59 (2010) 323-333.
[6] Gao J., Sun L., Gen M., A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, Computers & Operations Research, 35, 2008, 2892–2907.
[7] Hurink E., Jurisch B., Thole M., Tabu search for the job shop scheduling problem with multi-purpose machine, Operations Research Spektrum 15 (1994), 205–215.
[8] Pauli J., A hierarchical approach for the FMS schduling problem, European Journal of Operational Research 86(1) (1995), 32–42.
[9] Pezzella F., Morganti G., Ciaschetti G., A genetic algorithm for the Flexible Job-schop Scheduling Problem, Computers & Operations Research 35 (2008), 3202–3212.