Parallel Meta^{2}heuristics for the Flexible Job Shop Problem*

Wojciech Bożejko[1], Mariusz Uchroński[1], and Mieczysław Wodecki[2]

[1] Institute of Computer Engineering, Control and Robotics
Wrocław University of Technology
Janiszewskiego 11-17, 50-372 Wrocław, Poland
{wojciech.bozejko,mariusz.uchronski}@pwr.wroc.pl
[2] Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
mwd@ii.uni.wroc.pl

Abstract. In this paper we consider a double-level metaheuristic optimization algorithm. The algorithm proposed here includes two major modules: the machine selection module which is executed sequentially, and the operation scheduling module executed in parallel. On each level a metaheuristic algorithm is used, so we call this method meta^{2}heuristics. We carry out computational experiment using Graphics Processing Units (GPU). It was possible to obtain new the best known solutions for the benchmark instances from the literature.

1 Introduction

The flexible job shop problem which is considered here constitutes a generalization of the classic job shop problem where operations have to be executed on one machine from a set of dedicated machines. Then, as a job shop problem it also belongs to the strongly NP-hard class. Exact algorithms based on a disjunctive graph representation of the solution have been developed (see Pinedo [9]) but they are not effective for instances with more than 20 jobs and 10 machines. However, many approximate algorithms, mainly metaheuristic, have been proposed. Dauzère-Pérès and Pauli [3] used the tabu search approach extending the disjunctive graph representation for the classic job shop problem to take into consideration assigning operations to machines. Also Mastrolilli and Gambardella [6] proposed a tabu search procedure with effective neighborhood functions for the flexible job shop 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 Brandimarte [2]. Also genetic approaches have been adopted to solve the flexible job shop problem (Pezzella et al. [8]). Gao et al. [4] proposed hybrid genetic and variable neighborhood descent algorithm for this problem. In this paper we propose a parallel double-level tabu search metaheuristic for the flexible job shop problem. We apply INSA [7] and TSAB [7] algorithm on the second level of parallelism.

* The work was supported by MNiSW Poland, within the grant No. N N514 232237.

L. Rutkowski et al. (Eds.): ICAISC 2010, Part II, LNAI 6114, pp. 395–402, 2010.

2 Flexible Job Shop Problem

The flexible job shop problem (FJSP), also called the general job shop problem with parallel machines, can be formulated as follows. 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 in 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),
(iv) 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 sequence 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 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 $m = \sum_{j=1}^{m} m_j$.

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 $\mathcal{M}^{\mu(v)}$ 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 ($k = 1, 2, \ldots, q$) set of machines types. A sequence of operations sets $\mathcal{Q}^k = [\mathcal{Q}^k_{t_{k-1}+1}, \mathcal{Q}^k_{t_{k-1}+2}, \ldots, \mathcal{Q}^k_{t_{k-1}+m_k}]$, such that $\mathcal{O}^k = \bigcup_{i=t_{k-1}+1}^{t_{k-1}+m_k} \mathcal{Q}^k_i$ and $\mathcal{Q}^k_i \cap \mathcal{Q}^k_j = \emptyset$, $i \neq j$, $i, j = t_{k-1} + 1, t_{k-1} + 2, \ldots, t_{k-1} + m_k$ we call an *assignment* of operations to machines in the i-th set of machines types. In a special case a machine can execute no operations and then a set of operations assigned to execute by this machine is an empty set.

A sequence $\mathcal{Q} = [\mathcal{Q}^1, \mathcal{Q}^2, \ldots, \mathcal{Q}^q]$, where $\mathcal{Q}^i$ ($i = 1, 2, \ldots, q$ is an assignment in the i-th set of machines types we call an *assignment of operations of the set* $\mathcal{O}$ *to machines from the set* $\mathcal{M}$.

If the assignment of operations to machines has been carried out, then the optimal schedule of operations execution determination (including a sequence of operations execution on machines) boils down to solving the classic scheduling problem, it means the job shop problem.

Let $K = (K_1, K_2, ..., K_m)$ be a sequence of sets where $K_i \in 2^{\mathcal{O}^i}$, $i = 1, 2, ..., m$ (in particular case elements of this sequence can constitute empty sets). By $\mathcal{K}$ we denote the set of all such sequences. The number of elements of the set $\mathcal{K}$ is $2^{|\mathcal{O}^1|} \cdot 2^{|\mathcal{O}^2|} \cdot \ldots \cdot 2^{|\mathcal{O}^q|}$.

If $\mathcal{Q}$ is an assignment of operations to machines then $\mathcal{Q} \in \mathcal{K}$ (of course, the set $\mathcal{K}$ includes also sequences which are not feasible; that is such sequences do not constitute assignments of operations to machines).

For any sequence of sets $K = (K_1, K_2, ..., K_m)$ $(K \in \mathcal{K})$ by $\Pi_i(K)$ we denote the set of all permutations of elements from K_i. Thereafter, let $\pi(K) = (\pi_1(K), \pi_2(K), ..., \pi_m(K))$ be a concatenation m sequences (permutations), where $\pi_i(K) \in \Pi_i(K)$. Therefore $\pi(K) \in \Pi(K) = \Pi_1(K) \times \Pi_2(K) \times, ..., \Pi_m(K)$. It is easy to observe that if $K = (K_1, K_2, ..., K_m)$ is an assignment of operations to machines then the set $\pi_i(K)(i = 1, 2, ..., m)$ includes all permutations (possible sequences of execution) of operations from the set K_i on the machine$i$. Further, let $\Phi = \{(K, \pi(K)) : K \in \mathcal{K} \wedge \pi(K) \in \Pi(K)\}$. Any feasible solution of the FJSP is a pair $(Q, \pi(Q)) \in \Phi$ where $\mathcal{Q}$ 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-iv*).

By Φ° we denote a set of feasible solutions for the FJSP. Of course $\Phi^\circ \subset \Phi$.

3 Graph Representation

Any feasible solution $\Theta = (\mathcal{Q}, \pi(\mathcal{Q})) \in \Phi^\circ$ (where $\mathcal{Q}$ is an assignment of operations to machines and $\pi(\mathcal{Q})$ determines the operations execution sequence on each machine) of the FJSP can be shown as a directed graph with weighted vertexes $G(\Theta) = (\mathcal{V}, \mathcal{R} \cup \mathcal{E}(\Theta))$ where $\mathcal{V}$ is a set of vertexes and a $\mathcal{R} \cup \mathcal{E}(\Theta)$ is a set of arcs, whereas:

1) $\mathcal{V} = \mathcal{O} \cup \{s, c\}$, where s and c are additional (fictitious) operations which represents 'start' and 'finish', respectively. A vertex $v \in V \setminus \{s, c\}$ has two attributes:
   - $\lambda(v)$ – a number of machine on which an operation $v \in \mathcal{O}$ has to be executed,
   - $p_{v,\lambda(v)}$ – a weight of the vertex which equals to the time of operation $v \in \mathcal{O}$ execution on the assigned machine $\lambda(v)$.

   Weights of additional vertexes $(p_s = p_c = 0)$.

2) $\mathcal{R} = \bigcup_{j=1}^{n} \left[\bigcup_{i=1}^{o_j - 1} \{(l_{j-1} + i, l_{j-1} + i + 1)\} \cup \{(s, l_{j-1} + 1)\} \cup \{(l_{j-1} + o_j, c)\} \right]$.

   A set $\mathcal{R}$ includes arcs which connect successive operations of the job, arcs from the vertex s to the first operation of each job and arcs from the last operation of each job to the vertex c.

3) $\mathcal{E}(\Theta) = \bigcup_{k=1}^{m} \bigcup_{i=1}^{|\mathcal{O}^k| - 1} \{(\pi_k(i), \pi_k(i+1))\}$.

It is easy to observe, that arcs from the set $\mathcal{E}(\Theta)$ connect operations executed on the same machine (π_k is a permutation of operations executed on the machine M_k, that is operations from the set $\mathcal{O}^k$).

Arcs from the set $\mathcal{R}$ determine the operations execution sequence inside jobs (technological order) and arcs from the set $\mathcal{E}(\pi)$ the operations execution sequence on each machine.

Remark 1. *A pair $\Theta = (\mathcal{Q}, \pi(\mathcal{Q})) \in \Phi$ is a feasible solution for the FJSP if and only if $G(\Theta)$ does not includes cycles.*

Let $\Theta = (\mathcal{Q}, \pi(\mathcal{Q})) \in \Phi^\circ$ be a feasible solution for the FJSP and let $G(\Theta)$ be a graph connected with it. A sequence of vertexes $(v_1, v_2, \ldots, v_k)$ of the graph $G(\Theta)$ such, that $(v_i, v_{i+1}) \in \mathcal{R} \cup \mathcal{E}(\Theta)$ for $i = 1, 2, \ldots, k-1$, we call a *path* from the vertex v_1 to v_k. By $C(v, u)$ we denote a longest path (called a *critical path*) in the graph $G(\Theta)$ from the vertex v to u ($v, u \in \mathcal{V}$) and by $L(v, u)$ we denote a *length* (sum of vertexes weights) of this path.

It is easy to notice that the time of all operations execution $C_{\max}(\Theta)$ related with the assignment of operations $\mathcal{Q}$ and schedule $\pi(\mathcal{Q})$ equals to the length $L(s, c)$ of the critical path $C(s, c)$ in the graph $G(\Theta)$. A solutions of the FJSP boils down to determining a feasible solution $\Theta = (\mathcal{Q}, \pi(\mathcal{Q})) \in \Phi^\circ$ for which the graph connected with this solution $G(\Theta)$ has the shortest critical path, that is it minimizes $L(s, c)$.

If $\Theta = (\mathcal{Q}, \pi(\mathcal{Q})) \in \Phi^\circ$ is a feasible solution for the FJSP then $\mathcal{Q} = [\mathcal{Q}^1, \mathcal{Q}^2, \ldots, \mathcal{Q}^q]$, is an assignment of operations to machines and $\pi(\mathcal{Q}) = (\pi_1(\mathcal{Q}), \pi_2(\mathcal{Q}), \ldots, \pi_m(\mathcal{Q}))$ is a concatenation of m permutations, where a permutation $\pi_i(\mathcal{Q})$ determines a sequence of operations from the set $\mathcal{Q}^i$ which have to be executed on the machine M_i ($i = 1, 2, ..., m$).

Let $C(s, c) = (s, v_1, v_2, \ldots, v_w, c)$, $v_i \in \mathcal{O}$ ($1 \leq i \leq w$) be a critical path in the graph $G(\Theta)$ from the starting vertex s to the final vertex c. This path can be divided into subsequences of vertexes $\mathcal{B} = [B^1, B^2, \ldots, B^r]$ called *blocks* in the permutations on the critical path $C(s, c)$ (Grabowski [5]) where

(a) a block is a subsequence of vertexes from the critical path including successive operations executed directly one after other,
(b) a block includes operations executed on the same machine,
(c) a product of any two blocks is an empty set,
(d) a block is a maximal (due to including) subset of operations from the critical path fulfilling constrains (a)-(c).

In the further part only these blocks for which $|B^k| > 1$ will be considered, i.e. non-empty blocks.

If $(k = 1, 2, ..., r)$ is a block on the machine M_i ($i = 1, 2, ..., m$) from the type of machines t ($t = 1, 2, ..., q$) then we will denote it as follows:

$$B^k = (\pi_i(a^k), \pi_i(a^{k+1}), \ldots, \pi_i(b^{k-1}), \pi_i(b^k)),$$

where $1 \leq a^k \leq b^k \leq |Q_i^t|$.

Operations $\pi(a^k)$ and $\pi(b^k)$ in the block B^k are called *the first* and *the last*, respectively. In turn a block without the first and the last operation we call an *internal block*.

The change of operations order in any block does not generate the solution with less value of the cost function (see Grabowski [5]). At least one operation from any block should be moved before the first or after the last operation of this block to generate the solution (graph) with smaller weight of the critical path. We use this property to reduce the neighborhood size, i.e. do not generate solutions with greater values (comparing the the current solution) of the cost function.

4 Proposed Algorithm

The algorithm proposed here includes two major modules: the machine selection module and the operation scheduling module.

Machine selection module. This module is based on the tabu search approach and it works sequentially. It helps an operation to select one of the parallel machine from the set of machine types to process it.

Operation scheduling module. This module is used to schedule the sequence and the timing of all operations assigned to each machine from the center. It has to solve classic job shop problems after having assigned operations to machines. Two approaches: constructive INSA [7] and TSAB [7] (tabu search) were used on this level.

On each level a metaheuristic algorithm is used, so we call this method meta^{2}heuristics (*meta-square-heuristics*).

Algorithm 1. Tabu Search Based Meta^{2}heuristics (M^{2}h)

$\mathcal{Q}^*$ – the best known assignment;
$\pi(\mathcal{Q}^*)$ – operation sequence corresponding to the best known assignment $\mathcal{Q}*$;

Step 0. Find start assignment of operations on machines $\mathcal{Q}^0$ and corresponding operation sequence $\pi(\mathcal{Q}^0)$;

Step 1. Generate the neighborhood $\mathcal{N}(\mathcal{Q})$ of the current assignment $\mathcal{Q}$. Exclude from $\mathcal{N}(\mathcal{Q})$ elements from tabu list T;

Step 2. Divide $\mathcal{N}(\mathcal{Q})$ into $k = \left\lceil \frac{|\mathcal{N}(\mathcal{Q})|}{p} \right\rceil$ groups; Each group consist of at most p elements;

Step 3. For each group k find (using p processors) operation sequence $\pi(\mathcal{Y})$ corresponding to the assignment $\mathcal{Y} \in \mathcal{N}(\mathcal{Q})$ and value of the makespan $C_{max}(\mathcal{Y}, \pi(\mathcal{Y}))$;

Step 4. Find assignment $z \in \mathcal{N}(\mathcal{Q})$ such that

$$C_{max}(\mathcal{Z}, \pi(\mathcal{Z})) = min\{C_{max}(\mathcal{Y}, \pi(\mathcal{Y})) : \mathcal{Y} \in \mathcal{N}(x)\};$$

Step 5. **if** $C_{max}(\mathcal{Z}, \pi(\mathcal{Z})) < C_{max}(\mathcal{Q}^*, \pi(\mathcal{Q}^*))$ **then** $\pi(\mathcal{Q}^*) = \pi(\mathcal{Z})$; $\mathcal{Q}^* = \mathcal{Z}$; Include z in the list T; $\mathcal{Q} = \mathcal{Z}$; $\pi(\mathcal{Q}) = \pi(\mathcal{Z})$;

Step 6. **if** (*Stop condition* is true) **then** Stop; **else go to Step 1;**

In the second step of the algorithm a neighborhood $\mathcal{N}(\mathcal{Q})$ is divided into disjoint sets $\bigcup_{i=1}^{k} \mathcal{N}_i(\mathcal{Q}) = \mathcal{N}(\mathcal{Q}), \bigcap_{i=1}^{k} \mathcal{N}_i(\mathcal{Q}) = \emptyset$. For each group k values of the makespan are calculated using p GPU processors. Number of processors used in the third step depends on the neighborhood size. In the Step 3 the value of makespan corresponding to the assignment is calculated by means of INSA or TSAB algorithms. Tabu list T stores couples (v, k) where v is the position in the assignment vector and k is the machine to which v is assigned before the move. The first assignment is generated by the search for the global minimum in the processing time table taken from [8].

5 Computational Results

The parallel meta^{2}heuristic (M^{2}h) algorithm for the flexible job shop problem was coded in C (CUDA) for GPU, ran on the Tesla C870 GPU (512 GFLOPS) with 128 streaming processors cores and tested on the benchmark problems from literature. The GPU was installed on the Hewlett-Packard server based on 2 Dual-Core AMD 1 GHz Opteron processors with 1 MB cache memory and 8 GB RAM working under 64-bit Linux Debian 5.0 operating system. We compare our results with results obtained by other authors using a set of 10 problems from Brandimarte [2] and a set of 21 problems from Barnes and Chambers [1].

Table 1. Experimental results of the M^{2}h for Brandimarte [2] instances. The INSA algorithm was used in the operation scheduling module.

problem	$n \times m$	Flex.	o	t_s [s]	t_p [s]	speedup s
Mk01	10×6	2.09	55	133.61	10.79	12.38
Mk02	10×6	4.10	58	218.02	10.55	20.67
Mk03	15×8	3.01	150	6495.35	136.19	47.69
Mk04	15×8	1.91	90	620.69	29.59	20.98
Mk05	15×4	1.71	106	1449.80	74.55	19.45
Mk06	10×15	3.27	150	8094.39	147.83	54.75
Mk07	20×5	2.83	100	1939.33	57.92	33.48
Mk08	20×10	1.43	225	8950.91	643.39	13.91
Mk09	20×10	2.53	240	24586.00	641.88	38.30
Mk10	20×15	2.98	240	31990.55	593.49	53.90

The first phase of computational experiments was devoted to parallelization efficiency determination by estimating experimental speedup values. The sequential algorithm using one GPU processor was coded with the aim of determining the speedup value of the parallel algorithm. Such an approach is called orthodox speedup and it compares times of algorithms execution on machines with the same processors (1 versus p processors). Table 1 shows computational times for the sequential and the parallel algorithm as well as speedup values. The orthodox speedup s value can by set by the following expression $s = \frac{t_s}{t_p}$, where t_s - the computational time of sequential algorithm executed on the single processor of

the GPU, t_p - the computational time of parallel algorithm executed on p processors of the GPU. Flex. denotes the average number of equivalent machines per operation. As we can notice the highest speedup values were obtained for the problem instances with a bigger number of jobs n and the number of operations o. In this phase the simple INSA algorithm was applied in the operation scheduling module of the parallel meta^{2}heuristics.

Table 2. Values of the obtaining solutions for Barnes and Chambers [1] instances. The TSAB algorithm was used in the operation scheduling module of the M^{2}h. New the best known solutions are marked out by a bold font.

problem	$n \times m$	(LB,UB)	MG [6]	hGA [4]	M^{2}h
mt10c1	10×11	(655,927)	928	927	927
mt10cc	10×12	(655,914)	910	910	**908**
mt10x	10×11	(655,929)	918	918	922
mt10xx	10×12	(655,929)	918	918	918
mt10xxx	10×13	(655,936)	918	918	918
mt10xy	10×12	(655,913)	906	905	905
mt10xyz	10×13	(655,849)	847	849	855
setb4c9	15×11	(857,924)	919	914	914
setb4cc	15×12	(857,909)	909	914	**907**
setb4x	15×11	(846,937)	925	925	925
setb4xx	15×12	(846,930)	925	925	925
setb4xxx	15×13	(846,925)	925	925	925
setb4xy	15×12	(845,924)	916	916	**910**
setb4xyz	15×13	(838,914)	905	905	905
seti5c12	15×16	(1027,1185)	1174	1175	1174
seti5cc	15×17	(955,1136)	1136	1138	1136
seti5x	15×16	(955,1218)	1201	1204	**1199**
seti5xx	15×17	(955,1204)	1199	1202	**1198**
seti5xxx	15×18	(955,1213)	1197	1204	1197
seti5xy	15×17	(955,1148)	1136	1136	1136
seti5xyz	15×18	(955,1127)	1125	1126	1128

The second phase of the tests was refer to obtaining as good results of the cost function as possible. In this phase specialized TSAB algorithm of Nowicki and Smutnicki [7] was used in the operation scheduling module of the parallel meta^{2}heuristics. Despite of being more time-consuming the quality of the obtained results is much better than in the case of using INSA. By means of this approach it was possible to obtain 5 new the best known solutions for the benchmarks of Barnes and Chambers [1], for instances `mt10cc` (the new value 908), `set64c9` (907), `set64xy` (910), `seti5x` (1199) and `seti5xx` (1198).

The obtained results were also compared to other resent approach from the literature proposed for the flexible job shop problem. The proposed parallel M^{2}h algorithm managed to obtain the average relative percentage deviation to the best known solution of the Barnes and Chambers benchmark instances on

the level of 0.014% versus 0.036% of the MG [6] algorithm of Mastrolilli and Gambardella and 0.106% of the hGA [4] algorithm of Gao et al.

6 Conclusions

We have discussed a new approach to the scheduling problems with parallel machines, where assignment of operations to machines defines a classical problem without parallel machines. We propose double-level parallel metaheuristics, where each solution of the higher level, i.e. jobs assignment to machines, defines an NP-hard job shop problem, which we are solving by the second metaheuristics (constructive INSA or tabu search based TSAB) – we call such an approach meta^{2}heuristics. Using exact algorithms on both levels (i.e. branch and bound) makes possible to obtain an optimal solution of the problem.

References

1. Barnes, J.W., Chambers, J.B.: Flexible job shop scheduling by tabu search. Graduate program in operations research snd industrial engineering, The University of Texas at Austin, Technical Report Series: ORP96-09 (1996)
2. Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 41, 157–183 (1993)
3. Dauzère-Pérès, S., Pauli, J.: An integrated approach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search. Annals of Operations Research 70(3), 281–306 (1997)
4. Gao, J., Sun, L., Gen, M.: A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research 35, 2892–2907 (2008)
5. Grabowski, J.: Generalized problems of operations sequencing in the discrete production systems. In: Monographs, vol. 9. Scientific Papers of the Institute of Technical Cybernetics of Wrocław Technical Univercity (1979) (in Polish)
6. Mastrolilli, M., Gambardella, L.M.: Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling 3(1), 3–20 (2000)
7. Nowicki, E., Smutnicki, C.: A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research 91, 160–175 (1996)
8. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the Flexible Job-schop Scheduling Problem. Computers & Operations Research 35, 3202–3212 (2008)
9. Pinedo, M.: Scheduling: theory, algorithms and systems. Prentice-Hall, Englewood Cliffs (2002)