Computers & Industrial Engineering 59 (2010) 323–333

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Parallel hybrid metaheuristics for the flexible job shop problem ☆

Wojciech Bożejko [a,*], Mariusz Uchroński [a], Mieczysław Wodecki [b]

[a] Wrocław University of Technology, Institute of Computer Engineering, Control and Robotics, Janiszewskiego 11-17, 50-372 Wrocław, Poland
[b] University of Wrocław, Institute of Computer Science, Joliot-Curie 15, 50-383 Wrocław, Poland

ARTICLE INFO

Article history:
Received 16 October 2009
Received in revised form 22 April 2010
Accepted 6 May 2010
Available online 10 May 2010

Keywords:
Scheduling
Flexible job shop problem
Hybrid metaheuristics
Tabu search
Population-based algorithm

ABSTRACT

A parallel approach to flexible job shop scheduling problem is presented in this paper. We propose two double-level parallel metaheuristic algorithms based on the new method of the neighborhood determination. Algorithms proposed here include two major modules: the machine selection module refer to executed sequentially, and the operation scheduling module executed in parallel. On each level a metaheuristic algorithm is used, therefore we call this method meta^{2}heuristics. We carry out a computational experiment using Graphics Processing Units (GPU). It was possible to obtain the new best known solutions for the benchmark instances from the literature.

1. Introduction

The flexible job shop problem 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. Although exact algorithms based on a disjunctive graph representation of the solution have been developed (see Pinedo (2002)), they are not effective for instances with more than 20 jobs and 10 machines. Many approximate algorithms, mainly metaheuristics, have been proposed. Hurink, Jurisch, and Thole (1994) developed the tabu search method for this problem. Also Dauzère-Pérès and Pauli (1997) used the tabu search approach extending the disjunctive graph representation for the classic job shop problem taking into consideration the assignment of operations to machines. Mastrolilli and Gambardella (2000) 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 (1993) and Pauli (1995). These authors solved the assignment problem (i.e. using dispatching rules) and next used metaheuristics to solve the job shop problem. Also genetic approaches have been adopted to solve the flexible job shop problem. Recent works are these of Jia, Nee, Fuh, and Zhang (2003), Ho and Tay (2004), Kacem, Hammadi, and Borne (2002) Pezzella, Morganti, and Ciaschetti (2008). Gao, Sun, and Gen (2008) proposed the hybrid genetic and variable neighborhood descent algorithm for this problem. There are only a few papers considering parallel algorithms for the FJSP. Yazdani, Amiri, and Zandieh (2010) propose a parallel variable neighborhood search (VNS) algorithm for the FJSP based on independent VNS runs. Defersha and Chen (2009) describe a coarse-grain version of the parallel genetic algorithm for the considered FJSP basing on island-model of parallelization focusing on genetic operators used and scalability of the parallel algorithm. Both papers are focused on parallelization side of the programming methodology and they do not use any special properties of the FJSP.

In this paper we propose a parallel double-level metaheuristic approach for the flexible job shop problem based on two methods implemented on the higher level: tabu search and population-based approach. Additionally we have implemented the new method of the neighborhood determination (so-called *t-moves*) for the considered problem. We apply INSertion Algorithm INSA, (Nowicki & Smutnicki, 1996) and Tabu Search Algorithm with Backtracking TSAB, (Nowicki & Smutnicki, 1996) on the second level of parallelism. Using this method we have obtained the new best known solutions for the benchmark instances of Barnes and Chambers (1996) from the literature.

2. Hybrid metaheuristics

A hybrid approach to solving difficult optimization problems by using several metaheuristics simultaneously makes possible to

☆ This manuscript was processed by Area Editor Maged M. Dessouky.

* Corresponding author. Tel.: +48603672142.
E-mail addresses: wojciech.bozejko@pwr.wroc.pl (W. Bożejko), mariusz.uchronski@pwr.wroc.pl (M. Uchroński), mwd@ii.uni.wroc.pl (M. Wodecki).

0360-8352/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2010.05.004

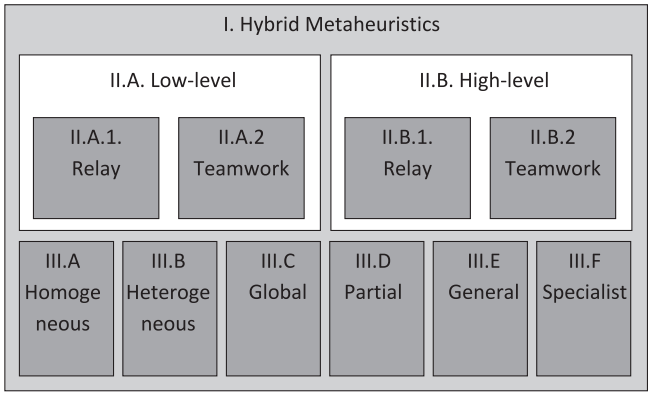

Fig. 1. Classification of hybrid metaheuristics proposed by Talbi (2002).

take advantages of all of them. Talbi (2002) provides a systematic characterization of parallel hybrid metaheuristics, which is visualized on the Fig. 1.

The upper part of the figure presents the hierarchical structure of the hybridization. In *high-level* hybrid algorithms the different metaheuristics are self-contained – there is no direct relationship to the internal workings of a metaheuristic. The *low-level* hybridization addresses the functional composition of a single optimization method – a given function of a metaheuristics is replaced by another metaheuristics. A *teamwork* hybridization represents cooperative models of optimization in which many parallel agents cooperate and each agent makes a search in its own part of the solution space. On the other hand, in *relay* hybrids, a number of metaheuristics is applied one after another; each one uses the output of the previous one as its own input, as in pipeline.

The lower part of the Fig. 1 (so-called a flat part) specifies the features of algorithms involved in the hybrid. In *homogeneous* hybrids all the combined algorithms use to same metaheuristic methods. On the contrary, in *heterogeneous* algorithms, different metaheuristics are used. In *global* hybrids, all the algorithms search in the whole solution space – the goal is to explore the space more thoroughly. All the algorithms solve the whole optimization problem. On the other hand, in *partial* hybrids, the considered problem is decomposed into subproblems; each one having its own solution space. Each algorithm is dedicated to search in one of these subspaces. Subproblems are linked with each other involving constraints between optima which are found by each algorithm. Algorithms establish communication to respect these constraints and create a solution of the main problem. In *general* hybrids, all the algorithms solve the same target optimization problem. *Specialist* hybrids combine algorithms which solve different problems, i.e. by solving another optimization problem into which the main problem is transformed.

Both algorithms proposed in this paper belong to the *high-level teamwork homogeneous general* hybrid metaheuristics. These algorithms possess two major modules: the machine selection module which is executed sequentially, and the operation scheduling module executed in parallel. Metaheuristics connected with each module are executed one after another, acting in a pipeline fashion. The proposed algorithms belong to the *partial* hybrids, because the problem to be sold is decomposed into subproblems connected with operation-machine assignments.

3. 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 $\mathscr{J} = \{1, 2, \ldots, n\}$ be a set of jobs which have to be executed on machines from the set $\mathscr{M} = \{1, 2, \ldots, m\}$. 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 allocation of jobs to machines from the adequate type and the schedule of jobs execution determination on each machine to minimize the total job 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 can 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 $\mathscr{O} = \{1, 2, \ldots, o\}$ be the set of all operations. This set can be partitioned into sequences corresponding to jobs where the job $j \in \mathscr{J}$ is a sequence of o_j operations which have to be executed in an order on dedicated machines (i.e. in the 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 $\mathscr{M} = \{1, 2, \ldots, m\}$ can be partitioned into q subsets of the same type (*nests*) where ith ($i = 1,2,\ldots,q$) type $\mathscr{M}^i$ 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 $v \in \mathscr{O}$ has to be executed on the machines of the type $\mu(v)$, i.e. on one of the machines from the set (nest) $\mathscr{M}^{\mu(v)}$ in the time p_{vj} where $j \in \mathscr{M}^{\mu(v)}$.

Let

$$\mathscr{O}^k = \{v \in \mathscr{O} : \mu(v) = k\}$$

be a set of operations executed in the kth nest ($k = 1,2,\ldots,q$). A sequence of operations sets

$$\mathscr{Q} = [\mathscr{Q}^1, \mathscr{Q}^2, \ldots, \mathscr{Q}^m],$$

such that for each $k = 1,2,\ldots,q$

$$\mathscr{O}^k = \bigcup_{i=t_{k-1}+1}^{t_{k-1}+m_k} \mathscr{Q}^i \quad \text{and} \quad \mathscr{Q}^i \cap \mathscr{Q}^j = \emptyset, \quad i \neq j,\ i,\ j = 1,2,\ldots,m,$$

we call an *assignment of operations from the set* $\mathscr{O}$ *to machines from the set* $\mathscr{M}$ (or shortly, operation-machine assignment).

A sequence $[\mathscr{Q}^{t_{k-1}+1}, \mathscr{Q}^{t_{k-1}+2}, \ldots, \mathscr{Q}^{t_{k-1}+m_k}]$ is an *assignment* of operations to machines in the ith nest (shortly, assignment in the ith nest). 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.

If the assignment of operations to machines has been completed, then the optimal schedule of operations execution determination (including a sequence of operations execution on machines) leads to the classic scheduling problem solving, i.e. the job shop problem (Grabowski & Wodecki (2005)).

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

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