Computers & Industrial Engineering 95 (2016) 156–163

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Parallel metaheuristics for the cyclic flow shop scheduling problem

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

[a] Department of Control Systems and Mechatronics, Faculty of Electronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland
[b] Wrocław Centre of Networking and Supercomputing, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
[c] Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland

ARTICLE INFO

Article history:
Received 11 November 2015
Received in revised form 5 March 2016
Accepted 7 March 2016
Available online 16 March 2016

Keywords:
Scheduling
Parallel algorithm
Cyclic flow shop problem
Block properties

ABSTRACT

In the paper there was proposed a new method of detection of block properties for cyclic flow shop problem with machine setups that uses patterns designated for each machine by solving the adequate traveling salesman problem. The proposed method is intended to be run in the environment of shared memory in concurrent computations, such as coprocessors, GPU, or machines with multi-core CPUs. The proposed method of accelerating the review of the neighborhood through the use of the blocks was tested on two parallel metaheuristics: tabu search and simulated annealing.

1. Introduction

Recently, there has been observed a growing interest in cyclic problems of tasks scheduling in both the environment of theorists dealing with discrete optimization problems and in the environment of practitioners in the industry. Cyclic production is, in fact, a very effective method in modern flexible manufacturing system as it significantly simplifies control, i.e. a fixed schedule is repeated in many periods. The most important benefit of the method is its ability to produce, in predetermined intervals, multi-assortment product mix resulting from customer demand. This process provides not only systematic replenishment of usually relatively small inventory of customers but also generates a systematic demand for both semi-finished or raw materials and materials obtained from suppliers. This method simplifies the management of supply chain. Another very important advantage of cyclic production is relatively easy detection of anomalies during manufacturing process which may indicate a deterioration of either the quality parameters of the production system or manufactured products themselves.

In the world literature there are many studies concerning various aspects of cyclic control in enterprises which manufacture products on a mass scale. There are examples of application of cyclic scheduling in various spheres of industry, transport and logistics (e.g. Gertsbakh & Serafini, 1991; Kats & Levner, 2010; Mendez, Cerda, Grossmann, Harjunkoski, & Fahl, 2006; Pinedo, 2005, 2008; Pinto, Barbosa-Póvoa, & Novais, 2005). Unfortunately, the existing models and calculation tools enable determination of the optimal (minimizing cycle time) control for production systems executing only a small number of tasks. In the work, there is considered a cyclic flow shop problem with setup times. Strong NP-hardness of many simple versions of cyclic scheduling problems, in particular, of the considered problem, limits the scope of applications of exact algorithms to instances with a small number of tasks, nevertheless, in this context of minimizing the cycle time, both the design and the use of exact algorithms seems to be fully justified (Brucker, Burke, & Groenemeyer, 2012). However, due to the NP-hardness, in determination of satisfactory solutions there are commonly fast approximate algorithms used based on local search techniques, for example: simulated annealing (in parallel version – Bożejko, Pempera, & Wodecki, 2015) or tabu search (Bożejko, Uchroński, & Wodecki, 2015, 2014). Methods of this type are usually based on a two-level decomposition of the problem: the first – determining the optimal sequence of tasks (upper level) and the second – multiple determining the minimum value of criteria for a given sequence of tasks (bottom level). In case of the conventional, non-cyclic scheduling problem, solution to the lower level can be obtained in a time-efficient manner by analyzing a specific graph. However, in case of the defined, in this paper, problem obtaining the solution to the lower level is a relatively time-consuming process since, in general, it requires the solution of a certain linear programming problem. Therefore, any special properties, including those that enable obtaining more efficient calculation of the cycle time, the search schedule and reduction of the

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

http://dx.doi.org/10.1016/j.cie.2016.03.008
0360-8352/

multiplicity of the locally browsed neighborhood or acceleration of its browse, are very desirable.

In this paper, we propose the use of new properties, the so-called blocks which reduce the number of solutions viewed while generating the neighborhood executed by local search algorithms, such as tabu search or simulated annealing. Determination of the blocks can be performed both sequentially and simultaneously, using multiprocessor calculations environment. Appropriate methods of construction are shown in this work with the use of model PRAM machine equipment, which for many years, has not only been the standard for the theoretical verification of the computational complexity of the parallel algorithms but also close to practice approximation of contemporary parallel architectures.

2. Problem description

The cyclic manufacturing process, considered in the work, can be formulated as follows: there is a set of n tasks given $\mathcal{J} = \{1, 2, \ldots, n\}$, which are to be performed in cycles (repeatedly) on machines from the set $\mathcal{M} = \{1, 2, \ldots, m\}$. A given task should be executed in a sequence, on each of the m machines $1, 2, \ldots, m$, in a technological order. The task $j \in \mathcal{J}$ is a sequence of m operations $O_{1,j}, O_{2,j}, \ldots, O_{m,j}$. The operation $O_{k,j}$ corresponds to execution of task j on machine k, in time $p_{k,j}$ $(k = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$. After the completion of a certain operation and before the start of the next operation there should be setups of a machine performed. Let $s^k_{i,j}$ $(k \in \mathcal{M},\ i \neq j\ i, j \in \mathcal{J})$ be the setup time between the operation $O_{k,i}$ and $O_{k,j}$.

A set of tasks executed in a single cycle is called PMS (*minimal part set*). MPSs are processed cyclically one after another. The order of tasks is to be determined (the same on each machine) in such a way which minimizes the cycle time, i.e. the time of commencement of the tasks from the set $\mathcal{J}$ in the next cycle. The following restrictions must be fulfilled:

(a) each operation can be executed only by one machine,
(b) none of the machines can execute more than one operation at the same time,
(c) the technological order of operations must be preserved,
(d) the execution of any operation cannot be interrupted before its completion,
(e) each machine, between successively performed operations, requires setup,
(f) each operation is executed sequentially (in successive MPSs) after the completion of cycle time.

The considered problem relies in determination of the moments of starting of the tasks' execution on machines that meet the limitations (a)–(f), so that the cycle time (the time at which the task is executed in the next MPS) was minimal. Let us assume that in each of the MPSs, on each machine, the tasks are executed in the same order. Thus, in the cyclic schedule the order of tasks' execution on the machines can be represented by a permutation of the tasks in the first MPS. In fact, on its basis, we can determine the starting moments of tasks' execution on the machines in the first MPS. Increasing them by a multiplication of the cycle time, one gets the starting moments of tasks' execution in any of the MPS (the starting moment of execution of any operations in the next MPS should be increased by cycle time). Let Φ be the set of all permutations of the elements from the set of tasks $\mathcal{J}$. Therefore, the considered in the work problem boils down to determining of permutations of tasks (elements of the set Φ) which minimizes the length of the cycle time. In short, this problem will be denoted by **CFS**.

3. Mathematical model

Let $[S^k]_{m\times n}$ be the matrix of starting moments of tasks' execution of kth MPS (for the established order $\pi \in \Phi$), where $S^k_{i,j}$ denotes the starting moment of execution of task j on the machine i. We assume that tasks in the next MPS-s are carried out cyclically. This indicates that there is a constant $T(\pi)$ (period) such that

$$S^{k+1}_{i,\pi(j)} = S^k_{i,\pi(j)} + T(\pi), \quad i = 1, \ldots, m,\ j = 1, \ldots, n,\ k = 1, 2, \ldots \tag{1}$$

Period $T(\pi)$ undoubtedly depends on permutation π and is called *time period* of the system. The minimum value $T(\pi)$, for a fixed π, will be called *minimum cycle time* and will be denoted by $T^*(\pi)$. Since the order of tasks within the given MPS is the same, it is sufficient just to designate the order of tasks π for a single (*the first*) MPS and move it by the quantity $k \cdot T(\pi),\ k = 1, 2, \ldots$ on the timeline. For the established order of execution of tasks $\pi \in \Phi$, optimum value of cycle time $T^*(\pi)$ can be determined by solving the following optimization task:

$$T^*(\pi) = \min\{T : T \in \mathbb{R}\}, \tag{2}$$

$$S_{i,\pi(j)} + p_{i,\pi(j)} \leqslant S_{i+1,\pi(j)}, \quad i = 1, \ldots, m-1,\ j = 1, \ldots, n, \tag{3}$$

$$S_{i,\pi(j)} + p_{i,\pi(j)} + s_{\pi(j,\pi(j+1))} \leqslant S_{i,\pi(j+1)}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n-1, \tag{4}$$

$$S_{i,\pi(n)} + p_{i,\pi(n)} + s_{\pi(n,\pi(1))} \leqslant S_{i,\pi(1)} + T, \quad i = 1, \ldots, m, \tag{5}$$

$$S_{i+1,\pi(n)} \leqslant S_{i,\pi(1)} + T, \quad i = 1, \ldots, m-1. \tag{6}$$

Without loss of generality, it is possible to assume that the starting moment of the first task's execution on the first machine is $S_{1,\pi(1)} = 0$. For any order of tasks in the first MPS and solving the above linear programming task, it is possible to determine the minimum cycle time in polynomial time. In case of an exact algorithm (complete overview) the solution to **CFS** problem should therefore be done for each of $n!$ permutation – element of a set Φ. The next chapter includes a presentation of an approximate solutions method to the considered in this work problem.

4. Solution method

In many heuristic algorithms solutions to NP-hard problems are constituted by reviewed neighborhoods, i.e. subsets of solution space. In case where solutions to problems are permutations neighborhoods are usually generated by insert or swap type moves and their combinations (Bożejko & Wodecki, 2007). They consist in changing positions of elements in the permutation. The number of elements of such a neighborhood is at least $n(n-1)/2$, where n is the size of the data. In practical applications (with large n), viewing the neighborhood is the most time consuming element of the algorithm. The description of computational experiments presented in the literature shows that the number of iterations of the algorithm has a direct impact on the quality of designated solutions. Hence, there is observed the search for methods accelerating the work of a single iteration of the algorithm. One of them is the reduction of the number of the neighborhood elements, their parallel generation and viewing. In case of tasks scheduling problems on multiple machines with the minimization of time of tasks' execution (C_{max}) there are 'blocks eliminating properties' (Grabowski & Wodecki, 2004) successfully used. Similar properties are implemented in the algorithm solving the problem of determining minimum cycle time, more specifically – a minimum time of a single machine run. The properties enable elimination of elements from the neighborhood that do not directly provide the improvement