Computers & Industrial Engineering 56 (2009) 1502–1509

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A fast hybrid tabu search algorithm for the no-wait job shop problem ☆

Wojciech Bożejko *, Mariusz Makuchowski

Wrocław University of Technology, Institute of Computer Engineering, Control and Robotics, Janiszewskiego 11-17, 50-372 Wrocław, Poland

ARTICLE INFO

Article history:
Received 19 April 2008
Received in revised form 7 September 2008
Accepted 12 September 2008
Available online 20 September 2008

Keywords:
Job shop problem
No-wait constraint
Tabu search
Hybrid algorithm

ABSTRACT

This paper describes a hybrid tabu search algorithm dedicated to a job shop problem with a no-wait constraint with a makespan criterion. The proposed here algorithm complexity is that the superior algorithm based on the tabu search technique selects parameters controlling the work of a certain constructional algorithm. This approach limits the checked solutions only to a group of solutions being able to be generated by the structural algorithm in question. It bears serious consequences both positive, for example it limits the research scope for a small fraction of relatively extremely well quality of acceptable solutions, and negative that is the lack of possibility of finding the optimal solution. In this paper numerical researches of the proposed algorithm are conducted as well as a comparative analysis with reference to the literature algorithms of the algorithm in question is made.

1. Introduction

In this work there was analyzed a strongly NP-hard job shop problem with an additional no-wait constraint with an optimalization criteria constituting a finishing moment of all tasks performance (makespan). The presented here problem comes from a classical job shop problem by setting additional requirements concerning an acceptable solution. The difference between the analyzed problem and its classical job shop problem equivalent consists in the necessity to conduct the subsequent operations of a given task exactly in the moment of finishing their technological predecessors. This requirement is often encountered in real production processes where machined components over the course of time change their physical–chemical parameters. For example, during the steel elements' production materials should be heated to a proper temperature in order to come next to form ready components. The formation process has to start exactly in the moment of obtaining a hot half-finished product (Wismer, 1972). Similarly, during some foodstuff' production because of both sanitary conditions and undergoing chemical–biological processes it is essential to carry out one activity directly after another (Hall & Sriskandarajah, 1996).

With reference to the complexity computational theory the classical job shop problem is classified to a group of strongly NP-hard problems, which is proved in a paper (Lenstra & Rinnooy Kan, 1979). In the same paper it is presented that in a case of adding a no-wait constraint the problem belongs to the same class too, i.e. strongly NP-hard problems. Although the theoretical complexity of both variants of the analyzed problem is the same, according to many researchers, with the practical point of view, a problem with a no-wait constraint is much more difficult. It results from the fact that the instance size, which can be solved exactly for example by the branch and bound (B&B) method, is much smaller for a variant with no-wait constraints. For example in a paper (Mascis & Pacciarelli, 2002) there was presented an algorithm finding an optimal solution in a reasonable time for an instance of a no-wait problem, in which the size did not exceed 15 tasks executed on 5 machines or 10 tasks executed on 10 machines. Besides, different kinds of approximate algorithms, an algorithm based on a tabu search technique (Macchiaroli, Mole, & Riemma, 1999), algorithm based on a technique of simulated annealing (Raaymakers & Hoogeveen, 2000) and a genetic algorithm with elements of simulated annealing (Schuster & Framinan, 2003), provide with solutions of a rather big deviation calculated relative to the reference solutions.

In this work there is proposed a hybrid algorithm based on a tabu search technique dedicated to a job shop problem with no-wait constraints. The presented algorithm is regarded as a hybrid since the superior component – a tabu algorithm controls an entry parameter of a constructional algorithm generating an acceptable solution of a job shop problem with no-wait constraints. A proper algorithm's construction generating a solution assures that a free placement of controlling parameters obtained by a superior algorithm always generates an acceptable solution. Besides, the number of different possible placements of controlling parameters is much more smaller than the number of all possible active solutions of a main problem, which limits the research scope. Generated solutions are of a relatively high quality taking into consideration the goal function's value which results from a very calculated constructional procedure described

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

* Corresponding author. Tel.: +48 603672142.
E-mail addresses: wojciech.bozejko@pwr.wroc.pl (W. Bożejko), mariusz.makuchowski@pwr.wroc.pl (M. Makuchowski).

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2008.09.023

in a further part of this work. Unfortunately, generally this approach makes finding optimal solutions, even in a case of the best selected controlling parameters, impossible.

The presented here hybrid tabu search algorithm was examined numerically on a literature group of test examples. These examples are commonly used to examine new algorithms for the discussed here subject. The obtained results are compared to the best known results and confronted with *GASA* algorithm (Schuster & Framinan, 2003) and the latest works of Framinan and Schuster (Framinan & Schuster, 2006; Schuster, 2006) taking into consideration both the algorithms' work time and the quality of obtained solutions. The work is summarized by author's conclusions about the proposed here algorithm.

2. Problem definition

The job shop problem with a no-wait constraint can be modelled as a classical job shop model introducing an additional condition concerning the solution acceptability. The problem can be defined as follows: there is a set of tasks: $J = \{1, 2, \ldots, r\}$, a set of operations: $O = \{1, 2, \ldots, n\}$ and a set of machines: $M = \{1, 2, \ldots, m\}$. The set of operations is divided into a r disjunctive subsets referring to individual tasks. A task $k \in J$ equates with a given sequence o_k of operations; $\sum_{k \in J} o_k = n$. This sequence establishes the required order of task's operation execution with a number k and for a notation simplification it is assumed that it looks like in the following way: $j_k + 1, j_k + 2, \ldots, j_k + o_k$, where $j_k = \sum_{i=1}^{k-1} o_i$ is a number of operations in tasks of $\{1, 2, \ldots, k-1\}$ and $j_1 = 0$. Additionally, for a further notation's simplification the sequence of task's operation ki.e. H we mark as O_K and assume that a notation $h \in O_k$ means that h is a certain element occurring in a sequence O_k (classically the $\in$ operator is defined for a set of elements, not for a sequence).

Furthermore, for an each operation $h \in O$ there is determined one machine $\mu(h) \in M$, on which it is to be conducted within a period $p_h > 0$. Additionally, in this model there exist three, typical for this kind of problem, constraints:

- each machine can execute at that particular moment not more than one operation;
- simultaneously more than one operation of a given task cannot be carried out at that particular moment;
- executing an operation on a machine cannot be broken.

The accepted schedule is defined by moments of starting an execution $S(h) \geqslant 0$ of an operation $h \in O$, N such that all above constraints are satisfied. The problem consists in finding an acceptable schedule minimizing the moment of executing all operations $\max_{h \in O} C(h)$ where $C(h) = S(h) + p_h$.

Setting an extra requirement:

- no waiting – each operation (apart from the first one) of a particular task has to begin exactly at the moment of finishing the previous operation's conduction of the same task,

we obtain a model of a job shop problem with a no-wait constraint marked in the Graham's notation (Graham, Lawler, Lenstra, & Rinnooy Kan, 1979) by $J|no - wait|C_{\max}$.

Let t_k denotes the starting time of the first operation of the kth task ($t_k = S(j_k)$). Due to the no-wait constraint this also determines the starting and finishing times of all operations $h \in O_k$ of this task. For a case with no waiting the problem solution can be represented by a vector $T = (t_1, t_2, \ldots, t_r)$ of terms of all tasks' execution beginning.

3. Algorithm generating solutions

The general idea of a proposed here algorithm consists in finding a solution by a systematic tasks adding to partly made in earlier iterations schedules. The controlling parameter, which is at the same time a governing variable for a superior algorithm, is an order of tasks schedule. This schedule is further called a loading permutation and is marked by $\pi = (\pi(1), \pi(2), \ldots, \pi(r))$, $\pi(i) \in J$, $1 \leqslant i \leqslant r$. A set of all possible loading permutations is marked as Π. In ith iteration a task with a number $\pi(i)$ is submitted to a schedule. The process of $\pi(i)$ task adding to a partial schedule consists in determining all moments of $S(h)$ beginning and moments of $C(h)$ finishing a conduction of each operation of this task $h \in O_{\pi(i)}$. After an arrangement a task does not change its place in a schedule i.e. established in ith iteration moments $S(h)$ and $C(h)$, $h \in O_{\pi(i)}$ will be the moments of beginning and finishing execution of these operations in the final arrangement. In ith iteration the way of establishing the moments of beginning and finishing the operation of a task $\pi(i)$ is determined relative to the possibly smallest moment (satisfying all required constraints) $t_{\pi(i)} \geqslant 0$ of the $\pi(i)$ task execution. Notice that each π loading permutation (a permutation of the J set) defines exactly one schedule, what's more this schedule is always acceptable, which results directly from its way of construction.

Conception of an algorithm generating solution from the sequence of jobs was proposed by Macchiaroli et al. (1999), Schuster (2006) and Schuster and Framinan (2003). In the work of Schuster (2006) the best timetabling for the given sequence is looked for, where the sequence is determined by the earliest moment of beginning the job's operation. This sequence cannot be changed by the timetabling algorithm, and that is why it is regarded as NP-hard. We proposed a different approach here: to use *loading permutation* as a base for a constructive algorithm which inserts tasks one after another. An order of jobs – in understanding of Framinan and Schuster – can be changed inside a timetable.

Fig. 1 presents a skeleton of the algorithm generating solutions. A variable *taskStart* is a time of beginning of the first operation of the task. The implemented procedure possesses a computational complexity $O(r^2N^2)$ for a case of at most one operation of job is executed on each machine (as in benchmark instances), where $N = \max_{k \in J}\{o_k\}$ means the largest number of operations in arranged tasks. In this case we can check possible collisions of operations in time $O(N)$ during inserting of each task. For a general case (no limit of the number of operations of job on each machine), an upper bound of a computational complexity of the algorithm is $O(r^3N^3)$.

As it has been mentioned before even examining all possible controlling parameter's combinations cannot guarantee finding an optimal solution. Fig. 2 shows an example of the optimal solution which cannot be obtained by the proposed constructive algorithm. However, a numerical examination was made, which gives an idea of this problem's scope. It is interesting both how often it occurs in order that the best possibly solution which can be obtained, marked as $\hat{\alpha}$, it would not be the optimal solution, marked as α^*, and what is the average deviation of the goal function's value of solution $\hat{\alpha}$ relative to the goal function's value of the optimal solution α^*. To answer the question one should have a possibly wide range of instances with known goal function's values of optimal solutions α^* and know goal function's values of the best possible ones to generate a solution $\hat{\alpha}$. For this purpose one has used 27 literature examples known as La01-La05, La06-La10, La 16-La20, Orb01-Orb10, Ft06 and Ft10. For these examples there are known optimal solutions obtained by an algorithm based on branch and bound technique