Advances in Engineering Software 40 (2009) 118–127

Contents lists available at ScienceDirect

Advances in Engineering Software

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

A hybrid heuristic to solve the parallel machines job-shop scheduling problem

Andrea Rossi [a,*], Elena Boschi [b]

[a] Department of Mechanical, Nuclear and Production Engineering, Università di Pisa, Via Bonanno Pisano, 25/B, Pisa 56126, Italy
[b] Department of Oncology, Transplants and Advanced Technologies in Medicine, Università di Pisa, Via Paradisa, 2, Pisa 56124, Italy

ARTICLE INFO

Article history:
Received 1 August 2006
Received in revised form 31 August 2007
Accepted 14 March 2008
Available online 3 June 2008

Keywords:
Hybrid systems
Ant colony optimization
Genetic algorithms
Parallel machines
Statistical analysis

ABSTRACT

This paper presents an advanced software system for solving the flexible manufacturing systems (FMS) scheduling in a job-shop environment with routing flexibility, where the assignment of operations to identical parallel machines has to be managed, in addition to the traditional sequencing problem. Two of the most promising heuristics from nature for a wide class of combinatorial optimization problems, genetic algorithms (GA) and ant colony optimization (ACO), share data structures and co-evolve in parallel in order to improve the performance of the constituent algorithms. A modular approach is also adopted in order to obtain an easy scalable parallel evolutionary-ant colony framework. The performance of the proposed framework on properly designed benchmark problems is compared with effective GA and ACO approaches taken as algorithm components.

1. Introduction

The job-shop scheduling problem with parallel machines (JSP-PM) represents an important problem encountered in current practice of manufacturing scheduling systems [1,2]. It consists of assigning any operation for each job to a resource of a candidate set of identical parallel machines (*assigning subproblem*), in addition to the classic JSP where the operations must be arranged on each (assigned) resource in order to minimize the makespan (*sequencing subproblem*). A candidate set of identical parallel machines is termed a machine type [1], a workcenter [2] or also a flexible manufacturing cell [3]. In opposition to classic job-shops where there is a single resource for each machine type, in flexible manufacturing systems (FMS) a number of parallel machines are available in order to both increase the throughput rate and avoid production stop when machines fail or maintenance occurs [4].

According to the $\alpha|\beta|\gamma$ notation of Graham et al. [5], the problem under consideration can be denoted by $FJ_m|prec|C_{max}$, where the field α denotes a flexible job-shop [6], the field β indicate *linear routings*, i.e. the occurrence of simple precedence constraints in the job routing and the field γ denotes the makespan which is the adopted measure of performance.

The problem is an extension of the classical $J_m||C_{max}$ which is strongly NP-hard [7] and, therefore, unlikely to reach an optimal solution in an acceptable amount of time by a computing strategy. An extensive and rapidly growing series of approaches have been reported in literature, but only approximate or heuristic methods make a tradeoff between solution quality and effective computing times (see [8,9] for a review). Recently, genetic algorithms (GA) became the state-of-the-art algorithms when computation time it not a concern [10]. More recently, this approach used in combination with taboo search or ant colony optimization (ACO) achieved good results and drastically reduced the CPU time [11,12].

In opposition to JSP, the literature on job-shop scheduling with parallel machines is scarce. As optimization techniques result in exponential computational complexity, until last decade manufacturing systems widely implemented dispatching rules systems [13,14]. In the last few years, effective heuristics have become available and some systems are proposed for the literature subject by means of a genetic algorithm [15], a Lagrangian relaxation [16] and an extension of the shifting bottleneck heuristic [17,18]. The SB procedure decomposes the overall problem into multiple instances of the scheduling problems for single group of parallel machines. More tractable scheduling subproblems solved by dispatching rules are a result of the decomposition approach. Upasani et al. [19] approach the problem by means of a temporal decomposition method. A rolling horizon (RH) framework, which divides the scheduling problem in subproblems which cover decisions for a limited time period into the future, are more effective to implement dynamic scheduling systems [20,21]. More recently an ant colony optimization for $FJ_m|s_{jk}$, $prec|C_{max}$, is developed where s_{jk} denotes the presence of sequence-dependent setup time [22]. This ACO is quite robust, because it uses a problem representation where assigning and sequencing constraints are deeply integrated. Even though those approaches succeeded in find the best solution in a number of simulated cases, the search efficiency seemed to offer a considerable improvement when evaluated by benchmark problems.

* Corresponding author. Tel.: +39 050 9130111; fax: +39 050 913040.
E-mail address: arossi@ing.unipi.it (A. Rossi).

0965-9978/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2008.03.020

ACO and GA, known as population-based algorithms, perform a multiple directed-parallel probabilistic searching technique by maintaining a population of solutions, also referred to as *agents*, each of these provides the capability of finding near-optimal solutions in a complex multi-dimensional searching space. A GA is modelled on an evolutionary approach, where recombination and selection operators are inspired by the natural evolution process. ACO is an emerging class of research, dealing with *swarm intelligence*, a set of artificial life methods which exploits the experience of an ant colony as a model of self-organisation in co-operative food retrieval by means of a proper pheromone trail model. Agents are capable of exploring and exploiting pheromone information which has been left on the pheromone trail when they traversed it.

A GA provides the advantage of performing a global search, but may cause stagnation or degeneration of search performance [23]. In particular, if GA mimics natural evolution, it should not operate on a single population in which a given individual has the potential to mate with any other partner in the entire population (*panmixia*). Instead, species are structured and tend to reproduce within subgroups or within neighbourhoods.

Among the existing types of structured genetic algorithms, distributed GAs are especially popular [24]. Their premise lies in partitioning the population into several subpopulations, each one being processed by a GA, independently of the others. Furthermore, a sparse migration of individuals produces an exchange of genetic material between the subpopulations that enhances diversity and usually improves the accuracy and efficiency of the algorithm. By making different decisions on the sub-algorithms of a distributed GA through the application of different search strategies, the search occurs at multiple exploration and exploitation levels [10]. Moreover, in recent years it has become evident that concentration on a sole heuristic is rather restrictive. A skilled combination of concepts derived from different heuristics can provide a more efficient behaviour and a higher flexibility when dealing with large-scale and real-world problems. A truly cunning hybrid strategies, combining GA and other systems as dispatching rules, TS or Simulated Annealing (SA). Mönch et al. apply a more sophisticated subproblem solution procedure which hybridizes dispatching rules and GA [25]. A simple priority index allows to select the more profitable batch of jobs from the overall set of batch and a genetic algorithm allows to select the more profitable assignment of the batches to the parallel machines. Wang and Zheng propose a SA procedure which controls the search process by means of an adaptive probability of mutation and an efficient strategy of selection [26]. This method is able to improve the search mechanism of GA.

The most general level of distributed search arises when each subpopulation can potentially run a different algorithm; this level is termed the *algorithm level* [27]. Another orthogonal level of distributed search can be defined by the relationship maintained among the subpopulations. Basically, if the amount of individuals of each subpopulation is fixed during evolution, i.e. the size of a subpopulation is made independent of the current success of its strategy, then it can be considered that the subpopulations are *collaborating* to find the optimum [28]. Otherwise, the kind of distributed search is competing [29,30].

In this context, the paper proposes an easy scalable *distributed-collaborative search* at the algorithm level between GA and ACO for the job-shop scheduling problem with parallel machines in order to improve the performance of the individual algorithms.

2. Problem formulation

In JSP-PM, O operations of n jobs have to be scheduled on m pools of machines, P_j ($j = 1,\dots,m$), each including m_j identical parallel machines, so that the quantity $k = \min_{j=1,\dots,m} m_j$ represents the degree of *parallelization capability* of the system according to the following equations:

$$\sum_{j=1}^{m} m_j = |M|, \tag{1}$$

$$P_w \bigcap P_z = \emptyset, \quad w,z = 1,\dots,m, \ \ w \neq z, \tag{2}$$

where M is the machine set. The problem with equal-size pools, i.e. $m_j = k$ for each P_j, is also referred to as the JSP-kPM *problem*. Each job i must be processed in accordance with its *linear routing* represented by a sequence of $l_i \leqslant m$ operations, $O_{i,j,r}$, each of which has to be processed as the rth operation on a single machine among the ones belonging to the jth pool which takes the processing time t_{ijr}. No machine can process more than one operation at a time; no operation $O_{i,j,r}$ can start until $O_{i,j,r-1}$ is completed or can stop after it starts; finally, an operation must be processed by one, and only one, machine. The schedule is represented by the starting times $st(O_{i,j,r})$, $i = 1,\dots,n$, $j = 1,\dots,m$, $r = 1,\dots,l_i$. The objective of JSP-kPM is to minimize the makespan.

Job-shop scheduling is a particular case of JSP-PM where the number of machines in each pool is $m_j = 1$. The assignment of operations to a machine of a pool gives a sort of further flexibility, and hence an increase in complexity, in addition to the flexibility represented by the possibility of sequencing operations on the machines. In this paper, schedules are semi-active and represented in the form of a precedence-feasible operation list, S, where the operation start times are set to be as early as possible while satisfying the routing constraints:

$$S = \left\{ (o_h, w_h)_{h=1,\dots,O} \middle| \begin{array}{l} \bigcup_{h=1}^{|O|} o_h = O, \quad w_h \in M, \\ \qquad st(o_{h_1}) \leqslant st(o_{h_2}), \quad h_1, h_2 \in O, \\ \qquad st(o_{h_1}) + t_{ijr} \leqslant st(o_{h_2}) \\ \qquad \text{if } o_{h_1} = O_{i,j,r}, o_{h_2} = O_{i,j,(r+1)}, \\ \qquad\quad r = 1,\dots,l_i - 1 \end{array} \right\}. \tag{3}$$

The precedence-feasible operation list is represented by a sequence of pairs (operation, machine on which the operation is processed) where the starting times are not decreasing and operations of the same job satisfying the routing constraint.

3. The hybrid GA–ACO heuristic for JSP-*k*PM

This section describes the basic principles of a distributed-collaborative search algorithm between GA and ACO. The co-operation mechanism is based on the parallel activity of the individual algorithms, where each of these performs the proper stochastic process and co-operates by sharing and modifying the common environment. The main loop consists of the execution of the two heuristics which are repeatedly synchronized at each epoch in order to mutually exchange information.

Algorithm 1. The hybrid GA–ACO heuristic for JSP-*k*PM

epoch = 0;
run an ACO epoch;
While (*loop* < *loop_limit*) **do**
  If (*epoch* < *epoch_limit*)
    1. If (*epoch* = 0) generate the GA subpopulations;
    2. Mutual exchange of information;
    3. Run a GA generation;
    4. Run an ACO epoch;
    5. For each operation list apply the local search procedure;
    6. **If** (the best solution, S_b, is improved) *epoch* = 0;
       Else *epoch* = *epoch* + 1;