Expert Systems with Applications 36 (2009) 5800–5806

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A hybrid genetic algorithm for no-wait job shop scheduling problems

Jason Chao-Hsien Pan [a,*], Han-Chiang Huang [b]

[a] Department of Business Administration, Takming University of Science and Technology, Taiwan
[b] Department of Industrial Management, National Taiwan University of Science and Technology, Taiwan

ARTICLE INFO

Keywords:
Scheduling
No-wait job shop
Hybrid genetic algorithm

ABSTRACT

A no-wait job shop (NWJS) describes a situation where every job has its own processing sequence with the constraint that no waiting time is allowed between operations within any job. A NWJS problem with the objective of minimizing total completion time is a NP-hard problem and this paper proposes a hybrid genetic algorithm (HGA) to solve this complex problem. A genetic operation is defined by cutting out a section of genes from a chromosome and treated as a subproblem. This subproblem is then transformed into an asymmetric traveling salesman problem (ATSP) and solved with a heuristic algorithm. Subsequently, this section with new sequence is put back to replace the original section of chromosome. The incorporation of this problem-specific genetic operator is responsible for the hybrid adjective. By doing so, the course of the search of the proposed genetic algorithm is set to more profitable regions in the solution space. The experimental results show that this hybrid genetic algorithm can accelerate the convergence and improve solution quality as well.

1. Introduction

A no-wait job shop (NWJS) is a job shop with the constraint that no waiting time is allowed between operations within any job. This phenomena often occurs in steel productions, plastic modeling and chemical industries, where it is physically prohibited to wait between processing. Another application can be found in modern manufacturing environment. Traditional scheduling models take little considerations about minimizing inventory while meeting required performance; no-wait scheduling problem can model this environment and the methodologies of it becomes a recent interest.

An NWJS problem with makespan (C_{max}), i.e., the maximum completion times of all jobs, as objective function is known to be NP-hard in the strong sense (Lenstra, Rinnooy Kan, & Brucker, 1977). Sahni and Cho (1979) showed that even when restricted to two machines, the problem is still NP-hard. Hall and Sriskandarajah (1996) reviewed relevant literatures on no-wait and blocking shop problems and found that NWJS problems have received surprisingly little attention from either a theoretical or computational perspective. Most researches on NWJS problems tackled the mathematical complexity issues (Goyal & Sriskandarajah, 1988; Hall & Sriskandarajah, 1996; Kamoun & Sriskandarajah, 1993; Sahni & Cho, 1979; Sriskandarajah & Ladet, 1986). As to the solution methodologies, Sriskandarajah and Ladet (1986), as well as Kubiak (1989) proposed pseudo-polynomial time algorithms for two machines no-wait job shop problems with unit processing time. Reddi and Ramamoorthy (1973) derived a lower bound for a special case of NWJS by requiring no subsuming jobs in the job set. Goyal (1975) found that solving NWJS problem is the same as solving a set of asymmetrical traveling salesman problems (ATSP), and the one with the smallest tour length is thus the optimal solution for the problem.

As for meta-heuristics, Raaymakers and Hoogeven (2000) presented a simulated annealing (SA) approach to NWJS problems with parallel machines and their results showed that SA improved on the best of the initial solutions by 10% on the average. Macchiaroli, Molè, Riemma, and Trifiletti (1996) presented a two-phase tabu search (TS) approach and found that it generated better solutions than those of the dispatching rules by an average of 9%. Brizuela, Zhao, and Sannomiya (2001) proposed a genetic algorithm (GA) using simple encoding and adequate decoding techniques to solve classical job shop benchmark problems with no-wait constraints.

Recently, Mascis and Pacciarelli (2002) extended the idea of disjunctive graph formulation to the so-called alternative graph formulation. By this formulation, solving a NWJS problem with makespan as objective is the same as finding a selection of the alternative arc set that minimizes the longest path of the graph with no cycles occurred. Computational results for the proposed greedy heuristics were conducted on classical job shop benchmark problems by imposing no-wait constraint.

This study proposes a hybrid genetic algorithm (HGA), which encapsulates a problem-specific genetic operator into the GA

* Corresponding author. Address: Department of Business Administration, Takming University of Science and Technology, 43 Huanshan Road, Section 1, Taipei 11451, Taiwan. Tel.: +886 22658 5801x2730; fax: +886 22658 5801x2731.
E-mail address: jpanbox@gmail.com (J.C.-H. Pan).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.07.005

procedures, to solve the NWJS problems. This problem-specific genetic operator randomly selects a consecutive section of genes from a chromosome, and treats it as a NWJS subproblem. This subproblem is then transformed to an ATSP and solved. After that, a local search based on ATSP solution is implemented. The best sequence obtained by the local search is placed back into the chromosome. By doing this, new building blocks consisting of suboptimal schedules for subproblem is introduced. The performance of the HGA is evaluated by means of 58 benchmark problems obtained from a public library (Beasly, 1990). The computational experiments show that HGA converges more quickly and to a better solution than pure genetic algorithm does. This suggests that the incorporation of the problem-oriented knowledge can direct the search of GA to more profitable regions in the solution space.

2. Problem formulation

Suppose there are n jobs, $J_1, J_2, \ldots, J_n$ and m machines, $M_1, M_2, \ldots, M_m$ in the shop. Job i has N_i operations and the processing time of its jth operation O_{ij} is deterministic and prescribed in advance. For job i, its O_{ij+1} must be initiated right after the completion of O_{ij}. The objective is to minimize the elapsed time from the start of the first job to the completion of the last job.

2.1. Problem assumptions

The following assumptions are made throughout this paper:

(1) The route of all jobs through all machines is prescribed in advance. Each job has its own processing route and may be different from each other.
(2) The processing time for each job on each machine is deterministic and known in advance.
(3) A machine can process at most one job at a time.
(4) A job cannot be processed on more than one machine simultaneously and it cannot be interrupted during processing.
(5) All jobs are ready at the beginning of the processing.
(6) No job can wait in process.
(7) Jobs are not allowed to reenter previous machines.

2.2. Integer programming model

Brizuela et al. (2001) proposed an integer programming model for NWJS based on the classical job shop model of Baker (1974). In their model, a set of constraints is modified to meet the no-wait requirement. Before presenting their model, the following variables are introduced and defined.

Symbol definition:

J_i	job number i
M_k	machine number k
O_k^i	the operation of J_i to be processed on M_k
O_{ij}	operation number j of J_i
N_i	number of operations of J_i

Problem parameters

M	a very large positive number
n	number of jobs
m	number of machines in the shop
p_{ik}	the processing time of J_i on M_k
r_{ijk}	1 if O_{ij} requires M_k; 0 otherwise

Decision variables:

$C_{\max}$	maximum completion time or makespan of jobs
s_{ik}	the earliest start time of J_i on M_k
$Z_{ii'k}$	1 if J_i precedes $J_{i'}$ (not necessarily immediately) on M_k; 0 otherwise

The integer programming model for NWJS is as follows:

$$\text{Minimize} \quad C_{\max}$$

$$\text{Subject to} \quad \sum_{k=1}^{m} r_{ijk}(s_{ik}+p_{ik}) = \sum_{k=1}^{m} r_{i,j+1,k}s_{ik}$$
$$i = 1,2,\ldots,n,\ j = 1,2,\ldots,N_j-1, \tag{1}$$
$$s_{i'k} - s_{ik} \geq p_{ik} - M(1-z_{ii'k})$$
$$1 \leq i < i' \leq n;\ k = 1,2,\ldots,m, \tag{2}$$
$$s_{ik} - s_{i'k} \geq p_{i'k} - Mz_{ii'k}$$
$$1 \leq i < i' \leq n;\ k = 1,2,\ldots,m, \tag{3}$$
$$\sum_{k=1}^{m} r_{i,N_i,k}(s_{ik}+p_{ik}) \leq C_{\max} \quad i = 1,2,\ldots,n, \tag{4}$$
$$C_{\max} \geq 0, \quad s_{ik} \geq 0 \quad i = 1,2,\ldots n;\ k = 1,2,\ldots m,$$
$$z_{ii'k} = 0 \text{ or } 1 \quad 1 \leq i < i' \leq n;\ k = 1,2,\ldots,m. \tag{5}$$

Constraint set (1) restricts that M_k begins the processing of $O_{i,j+1}$ right after it finishes O_{ij} to ensure that the constraints of no wait in process are obeyed. Constraint sets (2) and (3) enforce the requirement that only one job may be processed on a machine at any time; if both O_k^i and $O_k^{i'}$ are processed on M_k, either $s_{i'k}-s_{ik} \geqslant p_{ik}$ or $s_{ik} - s_{i'k} \geqslant p_{i'k}$ must be satisfied. Constraint sets (2) and (3) together guarantee that one of the constraints must hold when the other is eliminated since $z_{ii'k}$ is a binary variable and M is a large enough positive number. Constraint set (4) defines $C_{\max}$ to be minimized in the objective function. Constraint set (5) specifies the non-negativity of $C_{\max}$ and s_{ik} and the binary restrictions of $z_{ii'k}$.

3. A hybrid genetic algorithm

Genetic algorithms (GAs) are stochastic search techniques based on analogy to Darwinian natural selection. Individuals who fit the environment best should have a better chance to propagate their offspring. By the same reason, solutions that have the best "fitness" should receive higher probability to search their "neighbors". This idea was first proposed by Holland (1975). The main advantage of GA lies in its powerful implicit parallelism. In Holland's theory, a GA implicitly evaluates a number of patterns larger than population size without additional computational time and memory. The five main components of a GA (Jones & Rabelo, 1998) are listed below:

(1) Parameters setting for the algorithm, the operators and so forth.
(2) A way of encoding solutions to the problem – fixed length string of symbols.
(3) A way of initializing the population of solutions.
(4) Operations that may be applied to parents such as reproduction, crossover, mutation, and other domain specific operators.
(5) An evaluation function that returns a rating for each solution.

When problem-specific mechanism is encapsulated in component (4), either a local search procedure or a heuristic tailored for the problem, it is usually called a hybrid genetic algorithm. The incorporation of local search heuristics serves as an extra operator and it works together with crossover and mutation as a part of the GA's loop in order to accelerate convergence towards a better solution space.