A hybrid particle swarm optimization approach for the job-shop scheduling problem

Wei-jun Xia · Zhi-ming Wu

Abstract A new approximation algorithm is proposed for the problem of finding the minimum makespan in the job-shop scheduling environment. The new algorithm is based on the principle of particle swarm optimization (PSO). PSO combines local search (by self-experience) and global search (by neighboring experience), and possesses high search efficiency. Simulated annealing (SA) employs certain probability to avoid becoming trapped in a local optimum and the search process can be controlled by the cooling schedule. By reasonably combining these two different search algorithms, we develop a general, fast and easily implemented hybrid optimization algorithm; we called the HPSO. The effectiveness and efficiency of the proposed PSO-based algorithm are demonstrated by applying it to some benchmark job-shop scheduling problems. Comparison with other results in the literature indicates that the PSO-based algorithm is a viable and effective approach for the job-shop scheduling problem.

Keywords Hybrid optimization · Job-shop scheduling · Particle swarm optimization · Simulated annealing

1 Introduction

Scheduling is concerned with allocating limited resources to tasks to optimize certain objective functions. In the last four decades, many papers have been published in the scheduling area. One of the most popular models in scheduling is that of the job-shop. The classic job-shop scheduling problem (JSP) can be described as follows: a set of \( m \) machines and a set of \( n \) jobs are given. Each job consists of a sequence of operations that must be executed in a specified order. Each operation has to be performed on a given machine for a given time. A schedule is an allocation of operations on machines in time, i.e. a sequence of operations on machines. The problem is to find the schedule that the makespan (the maximum of job completion times) or other cost function is minimal, subject to the following constraints: (i) the operation precedence constraints are respected for every job; (ii) each machine can process at most one operation at a time; and (iii) an operation cannot be interrupted if it initiates processing on a given machine.

It is well-known that the JSP is NP-hard \([1]\) and belongs to the most intractable problems considered. The minimum makespan problem of job-shop scheduling is a classical combinatorial optimization problem that has received considerable attention in the literature. Historically, JSP was treated via classical math optimization and approximation methods. The math optimization methods are based chiefly on the branch and bound (BB) method. BB algorithms use a dynamically-constructed tree structure as a means of representing the solution space of all feasible sequences. The principle of BB is the enumeration of all feasible solutions of JSP. Early work was performed by Brooks and White \([2]\), followed by Greenberg \([3]\). Further research included Balas \([4]\), Florian et al. \([5]\), Lageweg et al. \([6]\), and Carlier and Pinson \([7]\). In the last decade, job-shop researchers considering the math optimization methods include Applegate and Cook \([8]\), Brucker et al. \([9]\), Perregaard and Clausen \([10]\), and Martin \([11]\). Although the computational study indicates that improvements have been achieved by BB methods, these methods cannot be applied to large problems and their execution necessitates a very good understanding of the JSP. As such, the algorithm lost its attraction to practitioners. Many researchers have turned their attention to approximation methods.

Although approximation methods do not guarantee achieving optimal solutions, they are able to attain near-optimal solutions, even for difficult-to-solve problems, with moderate computing times. Therefore, approximation methods are more suitable for larger problems.

Approximation procedures applied to JSP were first developed on the basis of priority dispatching rules (PDRs). PDRs are probably the most frequently applied heuristics for solving JSP because of their ease of implementation and their substan-
2 PSO algorithm

PSO is an evolutionary computation technique proposed by Kennedy and Eberhart [24, 25]. The particle swarm concept was based on the premise of social behavior. The original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock. A PSO algorithm mimics the behavior of flying birds and their means of information exchange to solve optimization problems. PSO has been introduced as an optimization technique in real-number spaces. But many optimization problems are set in a space featuring discrete components. Typical examples include problems that require ordering and route planning, such as in scheduling and routing problems. In this paper, we introduce a method of converting the continuous domain to the discrete domain for PSO. PSO requires only primitive and simple mathematical operators, and is computationally inexpensive in terms of both memory requirements and time.

2.1 Standard PSO algorithm

PSO is similar to the evolutionary algorithm in that the system is initialized with a population (“swarm”) of random solutions. Each individual or potential solution, called a particle, flies in the D-dimensional problem space with a velocity that is dynamically adjusted according to the flying experience of the individual and its colleagues. In past years, researchers have explored several models of PSO algorithm. In this paper, we use the global model equations, which are described as follows [26]:

\[ V_{id} = W \cdot V_{id} + C_1 \cdot \text{Rand}( ) \cdot (P_{id} - X_{id}) + C_2 \cdot \text{rand}( ) \cdot (P_{gd} - X_{id}) \]  \hfill (1a)

\[ X_{id} = X_{id} + V_{id} \]  \hfill (1b)

where \( V_{id} \), the velocity for particle \( i \), represents the distance to be traveled by particle \( i \) from its current position. \( X_{id} \) represents the particle position, \( P_{id} \) – also called “pbest”, the local best solution – represents \( i \)th particle’s best previous position, and \( P_{gd} \) – also called “gbest”, the global best solution – represents the best position among all particles in the swarm. \( W \) is inertia weight. It regulates the trade-off between the global exploration and local exploitation abilities of the swarm. The acceleration constants \( C_1 \) and \( C_2 \) represent the weight of the stochastic acceleration terms that pull each particle toward “pbest” and “gbest” positions. Rand() and rand() are two random functions with range \([0, 1]\).

For Eq. 1a, the first part represents the inertia of previous velocity; the second part is the “cognition” part, which represents individuals thinking independently; and the third part is the “social” part, which represents cooperation among the particles [27]. The process of implementing the PSO algorithm is as follows:

1. Initialize a swarm of particles with random positions and velocities in the D-dimensional problem space.
2. For each particle, evaluate the desired optimization fitness function.