Applied Soft Computing 11 (2011) 605–613

Contents lists available at ScienceDirect

Applied Soft Computing

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

Simulation optimization using particle swarm optimization algorithm with application to assembly line design

R.J. Kuo [a,*], C.Y. Yang [b,1]

[a] Department of Industrial Management, National Taiwan University of Science and Technology, No. 43, Section 4, Kee-Lung Road, Taipei 106, Taiwan
[b] Department of Industrial Engineering and Management, National Taipei University of Technology, No. 1, Section 3, Chung-Hsiao East Road, Taipei 106, Taiwan

ARTICLE INFO

Article history:
Received 19 June 2009
Received in revised form
22 November 2009
Accepted 6 December 2009
Available online 22 December 2009

Keywords:
Simulation
Particle swarm optimization algorithm
Simulation optimization
Assembly line

ABSTRACT

Assembly line design is an important part of production system. Some processes need to undergo changes in order to increase in efficiency. Computer simulation has been applied on process design for many decades. Traditionally, simulation had to run all possible alternatives of assembly line and was not considered as an optimization technique. Thus, this study employs particle swarm optimization (PSO) algorithm which is with mutation based on similarity for simulation optimization in order to optimize the managerial parameters in production system. Through experimentation designs and statistics tests, the simulation results show that the proposed method is better than other algorithms, like genetic algorithm and conventional PSO algorithm for solving assembly line design problem.

1. Introduction

Configuration of assembly lines, sequence of processes and timing of placing orders are critical operations in the production system. Their purpose is to assign production resources at the appropriate timing while confronting pre-determined performance measures in order to achieve optimal production efficiency. Therefore it is essential to determine which operation to use at what timing, and to apply the best method in order to assign these operations to the appropriate machines. Consumers today prefer innovative and up-to-date products, and to satisfy the demands of modern consumers, the industry has shifted from the conventional mass-production environment to a fast-changed and competitive environment where production volume is decreased, while product variety is increased. Also, traditionally it was common to rely on experience or intuition to find solutions for problems in production. This approach, however, is also no longer applicable to the contemporary market. Production management calls for new adaptive solutions. Modification or reconfiguration of production operation is needed. For large-scale enterprises, such modification is likely to affect many other operations, as well as cost. Many enterprises have adapted the simulation concept for improving production system. Simulation makes it possible for enterprises to evaluate the outcomes for different alternatives, thus minimizing unnecessary errors and costs.

Simulation generates numerical and logical models based on real-world problems, and imitates different scenarios using computers to find solutions for problems. For complex problems with high risks or that are impossible for real-world testing, computer simulation technology provides an effective tool to help plan for solving, analyzing and evaluating different alternatives. Computer hardware and software are continually improved and updated, and computer simulation software is high in computation efficiency and accuracy. However, computer simulations sometimes had not been considered an ideal tool for problem solving because conventional computer simulations used the exhaustive method, in which all possible solutions need to be fed into the simulation system to compute the best parameters setting and resources combination. Not only is this method very time-consuming and ineffective, but it is also costly. As a result, recent years many researches have purposed that by integrating optimization algorithms, computer simulation can determine the best combination more rapidly and more efficiently. Common optimization algorithms in computer simulations are Tabu search, neural network, simulated annealing and genetic algorithms.

Unlike other evolutionary algorithms, particle swarm optimization (PSO) algorithm [1] has not been applied to the area of simulation optimization. Thus, the main objective of this study is to utilize computer simulation technology to construct production assembly line and obtain the makespan and waiting time of each

* Corresponding author. Tel.: +886 2 27376328; fax: +886 2 27376344.
E-mail addresses: rjkuo@mail.ntust.edu.tw (R.J. Kuo), george0918@msn.com (C.Y. Yang).
[1] Tel.: +886 2 27712171; fax: +886 2 27316328.

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.12.020

product, to use PSO algorithm in computer simulation system, and to use this simulation system as the fitness function of the algorithm. The PSO algorithm used is with mutation based on similarity (PSOMS) [2]. The concept of PSOMS is based on similarity between the particle and the current global best particle in the swarm. And the collectivity was used to randomly mutate the position of the particles, maintaining swarm diversity in the search space.

In order to verify the validity of PSOMS in simulation optimization, the problem from Fontanili et al. [3] is adopted. For the purpose of comparison, three other algorithms, conventional PSO algorithms and two different types of genetic algorithms [3] are also employed to solve the simulation model.

The rest of the paper is organized as follows: Section 2 presents the backgrounds of this study. Section 3 proposes a particle swarm-based simulation method for production line, while the simulation results are shown in Section 4. Finally, the concluding remarks are made in Section 5.

2. Background

Since this study intends to apply optimization method to simulation analysis, this section mainly presents related literatures including simulation, particle swarm optimization algorithm, and simulation with optimization.

2.1. Simulation

Simulation can observe the different operational patterns quickly and choose the appropriate plan to solve problems [4]. Generally, simulation is a set of techniques for computers to imitate, or simulate, the operations of various kinds of real-world facilities or processes. The process imitates a real phenomenon with a set of mathematical formulas. The computer is used to generate a numerical model of reality for the purposes of describing complex interaction among components of a system. In addition to imitating processes to see how they behave under different conditions, simulations are also used to test new theories. After creating a theory of causal relationships, the theorist can codify the relationships in the form of a computer program. If the program then behaves in the same way as the real process, there is a good chance that the proposed relationships are correct. The simulation result cannot solve the problem, but may actually display the problem clearly [5].

Pegden et al. [6] proposed that simulation can be classified into three parts by its characteristics. They are (1) static and dynamic, (2) deterministic and stochastic and (3) discrete and continuous.

2.2. Particle swarm optimization (PSO) algorithm

The PSO algorithm shares many similarities with evolutionary computation techniques such as GAs. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, the PSO algorithm has no evolutionary operators, such as crossover and mutation. In the PSO algorithm, the potential solutions, called particles, move through the problem space by following the current optimum particles.

The PSO algorithm was originally designed by Kennedy and Eberhart [1]. In the original PSO algorithm, particle i is represented as $X_i = (x_{i1}, x_{i2}, \ldots, X_{iD})$, which represents a potential solution to a problem in D-dimensional space. Each particle keeps a memory of its previous best position Pbest, and a velocity along each dimension, represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. At each iteration, the position of the particle with the best fitness value in the search space, designated as g, and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle [7,8]. The method could be divided into GBEST and LBEST versions [7], whose main difference is their definition of the best. In the GBEST version, the particle swarm optimizer keeps track of the overall best value, and its location, obtaining thus far by any particle in the population, which is called *gbest* (G_{bestid}). For the LBEST version, in addition to *gbest*, each particle keeps track of the best solution, called *lbest* (G_{bestid}), and it is attained within a local topological neighborhood of particles. However, the particle velocities in each dimension are held to a maximum velocity, v_{max}, and the velocity in that dimension is limited to v_{max}. The updating rule is as follows:

$$v_{id}^{new} = v_{id}^{old} + c_1 \times rand_1 \times (P_{bestid} - X_{id}) + c_2 \times rand_2 \times (G_{bestid} - X_{id}) \quad (1)$$

$$X_{id}^{new} = X_{id}^{old} + v_{id}^{new} \quad (2)$$

where c_1 and c_2 determine the relative influence of the social and cognition components (learning factors), while $rand_1$ and $rand_2$ denote two random numbers uniformly distributed in the interval [0,1].

In the above PSO algorithm, the maximum velocity v_{max} serves as a constraint to control the global exploration ability of a particle swarm. As stated above, a larger v_{max} facilitates global exploration, while a smaller v_{max} encourages local exploitation. The concept of an inertia weight was developed to better balance exploration and exploitation in order to eliminate the need for v_{max}. The inclusion of an inertia weight in the particle swarm optimization algorithm was first reported by Shi and Eberhart [9]. The authors proposed two definitions of inertia weight: The first one defines inertia weight as a positive constant; the second defines it as variable decreasing with time. This is because the larger inertia weight has more exploitation ability at the beginning to find a good seed, and the later small inertia weight has more exploration ability to search the local area around the seed. The experimental result shows that weight decreasing with time has more potential. The updating rule is as follows:

$$v_{id}^{new} = W \times v_{id}^{old} + c_1 \times rand_1 \times (P_{bestid} - X_{id}) + c_2 \times rand_2 \times (G_{bestid} - X_{id}) \quad (3)$$

$$X_{id}^{new} = X_{id}^{old} + v_{id}^{new} \quad (4)$$

where W is an inertia weight. Shi and Eberhart [9] subsequently pointed out the impact that the inertia weight and maximum velocity have on the performance of the particle swarm optimizer, and provided guidelines for selecting these two parameters.

However, it should be noted that Eqs. (3) and (4) may have problem of dimensional consistencies. Basically, the dimension of the c_1 and $rand_1$ terms in Eq. (3) could be taken as $(\text{time})^{-2}$. Thus, the second and the third terms in Eq. (3) assume the dimension of acceleration. Due to left hand side, it is necessary to multiply velocity by Δt, the time step, which becomes unity in the present case, denoting changes from iteration $i-1$ to i, in order to get the correct dimension of them. It is also similar for Eq. (4) that the second term assumes the correct dimension when taken as $v_{id}^{new}\Delta t$. However, the present form results through the implicit assumption that Δt equals 1 [10,11].

During the running of the algorithm, the particles become more and more similar, which make the swarm premature convergence around the local solution. To avoid that particle swarm optimization searches the local area. Xie et al. [12] proposed dissipative particle swarm optimization (DPSO) in 2002. At each iteration, DPSO sample the random particle to mutate. Mutation can lead to leave local area around seed. The mutative rule is as follows:

$$If(rand_1 < C_v) \quad then \quad v_{id} = rand_2 \times \frac{v_{max}}{C_m} \quad (5)$$