ON THE PERFORMANCE OF THE PARTICLE SWARM OPTIMIZATION ALGORITHM WITH VARIOUS INERTIA WEIGHT VARIANTS FOR COMPUTING OPTIMAL CONTROL OF A CLASS OF HYBRID SYSTEMS

M. SENTHIL ARUMUGAM AND M. V. C. RAO

Received 23 October 2005; Accepted 8 January 2006

This paper presents an alternative and efficient method for solving the optimal control of single-stage hybrid manufacturing systems which are composed with two different categories: continuous dynamics and discrete dynamics. Three different inertia weights, a constant inertia weight (CIW), time-varying inertia weight (TVIW), and global-local best inertia weight (GLbestIW), are considered with the particle swarm optimization (PSO) algorithm to analyze the impact of inertia weight on the performance of PSO algorithm. The PSO algorithm is simulated individually with the three inertia weights separately to compute the optimal control of the single-stage hybrid manufacturing system, and it is observed that the PSO with the proposed inertia weight yields better result in terms of both optimal solution and faster convergence. Added to this, the optimal control problem is also solved through real coded genetic algorithm (RCGA) and the results are compared with the PSO algorithms. A typical numerical example is also included in this paper to illustrate the efficacy and betterment of the proposed algorithm. Several statistical analyses are carried out from which can be concluded that the proposed method is superior to all the other methods considered in this paper.

1. Introduction

The hybrid systems of interest contain two different types of categories, subsystems with continuous dynamics and subsystems with discrete dynamics that interact with each other. Such hybrid system frameworks arise in varied contexts in manufacturing, communication networks, automotive engine design, computer synchronization, and chemical processes, among others. In hybrid manufacturing systems, the manufacturing framework is composed of the event-driven dynamics of the parts moving among different machines and the time-driven dynamics of the processes within particular machines.

Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2006, Article ID 79295, Pages 1–17
DOI 10.1155/DDNS/2006/79295

Frequently in hybrid systems, the event-driven dynamics are studied separately from the time-driven dynamics, the former via automata or Petri net models, PLC, and so forth, and the latter via differential or difference equations. Two categories of modelling framework have been proposed to study hybrid systems. The first category extends event-driven models to include time-driven dynamics and the second category extends the traditional time-driven models to include event-driven dynamics. The hybrid framework considered in this paper adopts the first category and it is motivated by the structure of many manufacturing systems.

In these systems, discrete entities that are often referred to as jobs are associated with *temporal states* and *physical states*. The temporal state of a job evolves according to event-driven dynamics and includes information such as the processing time or departure time of the job. The physical state evolves according to the time-driven dynamics and describes some measures of "quality" of the job such as temperature, weight, and chemical composition. The interaction of time-driven with event-driven dynamics leads to a natural tradeoff between temporal requirements on job completion times and physical requirements on the quality of the completed jobs (see Figure 2.1). Such modelling frameworks and optimal control problems have been considered in [2, 14, 18].

The task of solving these problems was simplified by exploiting structural properties of the optimal sample path. In particular, an optimal sample path is first decomposed into decoupled segments (*busy periods*) and then into *blocks* with *critical jobs*. The identification of such critical jobs is the crucial part of the analysis and the key to developing effective algorithms for solving the optimal control problems, which has been realized using nonsmooth optimization techniques [2, 4]. By this way few algorithms were developed for solving the optimal control problems, and they decompose the entire optimal control problem into a set of smaller convex optimization subproblems with linear constraints. The first is a backward recursive algorithm [17] which proceeds backward in time from the last job to the first job. The complexity of the problem was thus reduced from exponential N (the number of jobs processed) to a linear bounded one by $2N - 1$. The second is a forward algorithm whose complexity is simply N as shown in [3]. The third is an improved forward algorithm, which is an extension of forward algorithm. Instead of increasing the number of jobs by one at every step, this algorithm may increase the number of jobs by more than one [18].

The real coded genetic algorithm (RCGA) with different forms of selection methods and crossover methods is implemented to solve the optimal control problem [1]. The selection procedure comprises roulette wheel selection (RWS), tournament selection (TS), and the hybrid combination of both. Arithmetic crossover (AMXO) and dynamic mutation (DM) are the other two genetic operators considered.

In the PSO algorithm, initially a constant inertia weight was used for solving the optimization problem and later it was replaced with a monotonically or linearly decreasing inertia weight in order to improve the performance of the PSO algorithm. In this paper a new inertia weight is proposed in terms of the global best and personal best values of the objective function. The result obtained through this proposed algorithm is better than all the other methods.

This paper is divided into eight sections. Section 2 describes the formulation of the objective function. A description of PSO is presented in Section 3. In Section 4, review

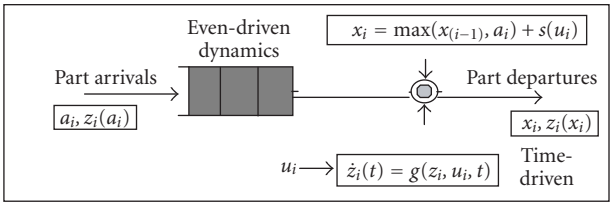


Figure 2.1. A hybrid model for a single-stage hybrid manufacturing system.

of RCGA is described. The experimental settings for both PSO and RCGA are given in Section 5. Examples and simulations are presented in Section 6. Section 7 depicts the discussions and comparison between the simulated results with PSO and RCGA, and conclusions are shown in Section 8.

2. Problem formulation of single-stage hybrid system

The hybrid model for a single-stage hybrid manufacturing system shown in Figure 2.1, receives a sequence of N number of jobs $(C_1, C_2, \ldots, C_N)$ with the known arrival times $0 \le a_1 \le \cdots \le a_N$ from an external source. The jobs are processed on first-come first-serve (FCFS) basis and the processing time $s(u_i)$ is a function of a control variable u_i, and $s(u_i) \ge 0$.

The time-driven dynamics of the hybrid framework which is given in (2.1) evolves the job C_i initially at some physical state ξ_i at time x_0, and the event-driven dynamics is given in the form of standard Lindley equation and shown in (2.2):

$$\dot{z}_i(t) = g(z_i, u_i, t), \qquad z_i(x_o) = \xi_i, \tag{2.1}$$

$$x_i = \max(x_{(i-1)}, a_i) + s(u_i), \quad i = 1, \ldots, N, \tag{2.2}$$

where x_i is the departure or completion time of ith job.

The presence of the control variable u_i in both the physical state $\dot{z}_i$ of the time-driven dynamics (2.1) and the next temporal state x_i of the event-driven dynamics (2.2) justifies the hybrid nature of the system [18]. The typical optimal path trajectory of the hybrid system is shown in Figure 2.2.

When the first job arrives at a_1, the physical state starts to evaluate the time-driven dynamics until it reaches the departure time x_1. Since the first job completes before the second job arrives, there is an idle period, in which the server has no jobs to process. The physical state again begins evolving the time-driven dynamics at time a_2 (arrival of second job) until the second job completes at x_2. But here the third job has arrived before the second job is completed. So the third job is forced to wait in the queue until time x_2. After the second job completes at x_2 the physical state begins to process the third job. As indicated in Figure 2.2, not only the arrival time and departure time cause switching in the time-driven dynamics according to (2.1), but the sequence in which these events occur is governed by the event-driven dynamics given in (2.2). In this framework, each job must be processed until it reaches a certain quality level denoted by Γ_i. That is, the

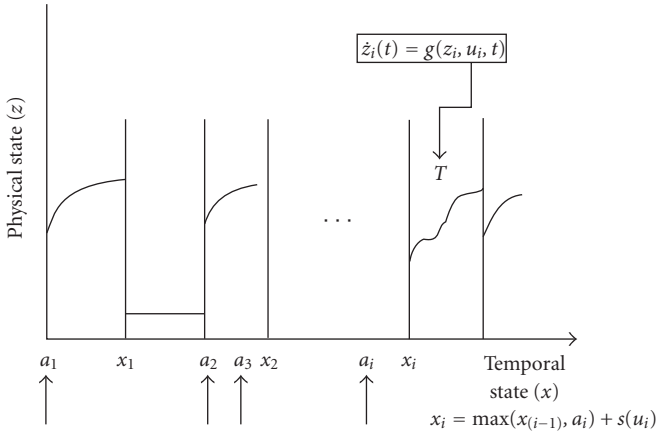


Figure 2.2. The optimal path trajectory of the hybrid system.

processing time for each job is chosen such that

$$s_i(u_i) = \min\left[t \geq 0;\ \mathrm{z_i}\,(t_0) = \int_{t_o}^{t_o+t} g_i(\tau, u_i, t)\,d\tau + z(t_0) \in \Gamma_i\right]. \tag{2.3}$$

For the above single-stage framework defined by (2.1) and (2.2), the optimal control objective is to choose a control sequence $\{u_1,\ldots,u_N\}$ to minimize an objective function of the form

$$J = \sum_{i=1}^{N} \{\theta_i(u_i) + \phi_i(x_i)\}. \tag{2.4}$$

Although, in general, the state variables $z_i,\ldots,z_N$ evolve continuously with time, minimizing (2.4) is an optimization problem in which the values of the state variables are considered only at the job completion times $x_1,\ldots,x_N$. Since the stopping criterion in (2.3) is used to obtain the service times, a cost on the physical state $z_i(x_i)$ is unnecessary because the physical state of each completed job satisfies the quality objectives, that is, $z_i(x_i) \in \Gamma_i$.

Generally speaking, u_i is a control variable affecting the processing time through $s_i = s(u_i)$ for extension to cases with time-varying controls $u_i(t)$ over a service time. By assuming $s_i(\cdot)$ is either monotone increasing or monotone decreasing, given a control u_i, service time s_i can be determined from $s_i = s(u_i)$ and vice versa.

For simplicity, let $s_i = u_i$, and the rest of the analysis is carried out with the notation u_i. Hence the optimal control problem, denoted by **P** is of the following form:

$$\begin{aligned} P:\ &\min_{u_1,\ldots,u_N}\left\{J = \sum_{i=1}^{N}\{\theta_i(u_i) + \phi_{\mathrm{i}}(x_{\mathrm{i}})\} : u \geq 0,\ i = 1,\ldots,N\right\} \\ &\text{subject to } x_i = \max\left(x_{(i-1)}, a_i\right) + s(u_i), \quad i = 1,\ldots,N. \end{aligned} \tag{2.5}$$

The optimal solution of P is denoted by u_i^* for $i = 1,\ldots,N$, and the corresponding departure time in (2.5) is denoted by x_i^* for $i = 1,\ldots,N$.

3. Particle swarm optimization

Dr. Kennedy and Dr. Eberhart introduced particle swarm optimization in [6] as an alternative to genetic algorithm (GA). The PSO technique has ever since turned out to be a competitor in the fields of numerical optimization.

The evolutionary algorithms, EAs, (GA and EP) are search algorithms based on the simulated evolutionary process of natural selection, variation, and genetics. The evolutionary algorithms are more flexible and robust than conventional calculus-based methods. Both GA and EP can provide a near global solution. However, the encoding and decoding schemes essential in the GA approach make it take longer time for convergence. EP differs from traditional GAs in two aspects: EP uses the control parameters (real values), but not their coding as in traditional GAs, and EP relies primarily on mutation and selection, but not crossover, as in traditional GAs. Hence, considerable computation time may be saved in EP. Although GA and EP seem to be good methods to solve optimization problems, when applied to problems consisting of high number of local minima, the solutions obtained from both methods are just near global optimum ones.

Particle swarm optimization (PSO) is one of the modern heuristic algorithms under the EAs and has gained a lot of attention in various power system applications [7]. PSO can be applied to nonlinear and noncontinuous optimization problems with continuous variables. It has been developed through simulation of simplified social models. PSO is similar to the other evolutionary algorithms in that the system is initialized with a population of random solutions. However, each potential solution is also assigned a randomized velocity, and the potential solutions, called agents, correspond to individuals. Each agent in PSO flies in the n-dimensional problem space with a velocity, which is dynamically adjusted according to the flying experiences of its own and its colleagues [6, 12]. Generally, the PSO is characterized as a simple heuristic of well-balanced mechanism with flexibility to enhance and adapt to both global and local exploration abilities. It is a stochastic search technique with reduced memory requirement, computationally effective, and easier to implement compared to other EAs. PSO developed by Dr. Kennedy and Dr. Eberhart shares some of the common features available in other EAs, except the selection procedure [11]. Also, PSO will not follow "survival of the fittest," the principle of other EAs. PSO when compared to EP has very fast converging characteristics; however, it has a slow fine-tuning ability of the solution. Also PSO has a more global searching ability at the beginning of the run and a local search near the end of the run. Therefore, while solving problems with more local optima, there are more possibilities for the PSO to explore local optima at the end of the run [6, 8].

The underlying motivation for the development of PSO algorithm was the social behavior of animals such as bird flocking, fish schooling, and swarm. Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variable, and acceleration in movement. PSO is similar to genetic algorithm (GA) in that the system is initialized with a population of random solutions [8]. However, in PSO, each individual of the population, called particle, has an adaptable velocity, according to which

it moves over the search space. Each particle keeps track of its coordinates in hyperspace, which are associated with the solution (fitness) it has achieved so far [15]. This value is called personal best and is denoted by "pbest." Additionally among these personal bests, there is only one, which has the best fitness. In a search space of D-dimensions, the ith particle can be represented by a vector $X_i = X_1, X_2, \ldots, X_D$. Similarly, the relevant velocity is represented by another D-dimensional vector $V_i = V_1, V_2, \ldots, V_D$. The best among pbest is called the global best and is denoted by "gbest" in (3.1).

$$V_i = wV_i + \rho_1 r_1 (\text{gbest} - X_i) + \rho_2 r_2 (\text{pbest} - X_i). \tag{3.1}$$

In (3.1), w is known as the inertia weight. The best-found position for the given particle is denoted by pbest and gbest is the best position known for all particles. The parameters ρ_1 and ρ_2 are set to constant values, which are normally given as 2, whereas r_1 and r_2 are two random values, uniformly distributed in [0,1]. The position of each particle is updated every generation. This is done by adding the velocity vector to the position vector, as described in (3.2) below:

$$X_i = X_i + V_i. \tag{3.2}$$

The choice of the PSO algorithm's parameters (such as the group's inertia) seems to be of utmost importance for the speed and efficiency of the algorithm. Inertia weight plays an important role in the convergence of the optimal solution to a best optimal value as well as the execution time of the simulation run. The inertia weight controls the local and global exploration capabilities of PSO [16]. Large inertia weight enables the PSO to explore globally, and small inertia weight enables it to explore locally. So the selection of inertia weight and maximum velocity allowed may be problem-dependent.

There are five basic principles of swarm intelligence. First is the proximity principle: the population should be able to carry out simple-space and time computations. Second is the quality principle: the population should be able to respond to quality factors in the environment. Third is the principle of diverse response: the population should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the population should not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the population must be able to change behavior mode when it is worth the computational price. Note that principles four and five are the opposite sides of the same coin. The particle swarm optimization concept and paradigm presented in this paper seem to adhere to all five principles. The population is responding to the quality factors pbest and gbest. The allocation of responses between pbest and gbest ensures a diversity of response. The population changes its state (mode of behavior) only when gbest changes, thus adhering to the principle of stability. The population is adaptive because it does change when gbest changes [15, 16].

4. Review of real coded genetic algorithm

The genetic algorithm (GA) is a search technique based on the mechanics of natural genetics and survival of the fittest. GA is an attractive and alternative tool for solving complex multimodal optimization problems [10, 13].

GA is unique as it operates from a rich database of many points simultaneously. Any carefully designed GA is only able to balance the exploration of the search effort, which means that an increase in the accuracy of a solution can only come at the sacrifice of convergent speed, and vice versa. It is unlikely that both of them can be improved simultaneously. Despite their superior search ability, GA still fails to meet the high expectations that theory predicts for the quality and efficiency of the solution. As widely accepted, a conventional GA is only capable of identifying the high performance region at an affordable time and displaying inherent difficulties in performing local search for numerical applications [5].

To improve the final local tuning capabilities of a binary coded genetic algorithm, which is a must for high precision problems, new genetic operators have been introduced [9]. The main objective behind real coded GA implementations is to move the genetic algorithm closer to the problem space. For most applications of GAs to constrained optimization problems, the real coding is used to represent a solution to a given problem. Such coding is also known as floating-point representation, real number representation.

GAs start searching the solution by initializing a population of random candidates to the solution. Every individual in the population undergoes genetic evolution through crossover and mutation. The selection procedure is conducted based on the fitness of each individual. In this paper, the roulette wheel selection (RWS), tournament selection (TS), and a hybrid of both selection procedures are adopted in conjunction with the elitist strategy. By using the elitist strategy, the best individual in each generation is ensured to be passed to the next generation. The selection operator creates a new population by selecting individuals from the old populations, biased towards the best. The chromosomes, which produce the best optimal fitness, are selected for next generations. Crossover is the main genetic operator, which swaps chromosome parts between individuals. Crossover is not performed on every pair of individuals; its frequency being controlled by a crossover probability (P_c). The probability should have a larger value, typically, $P_c = 0.8$. The last operator is mutation which changes a random part of string representing the individual. This operator must be used with some care, with low probability, typically P_m, ranges from 0.01 to 0.1 for normal populations. The algorithm is repeated for several generations until one of the individuals of population converges to an optimal value or the required number of generations (max_gen) is reached.

Michaelewicz indicates that for real valued numerical optimization problems, floating-point representations outperform binary representations because they are more consistent, more precise, and lead to faster execution. For most applications of GAs to optimization problems, the real coding technique is used to represent a solution to a given problem. Hence, we use GA with real values in both conventional and hybrid forms, for solving the optimal control problem.

5. Experimental parameter settings

5.1. Initial population. The initial populations are generated randomly, and the number of chromosomes generated per population is equal to the dimension of the optimal problem or equal to the number of jobs (N) involved in the main objective function. In

this paper the number of chromosomes generated per population (or the dimension of the optimal control problem) is varying from 5 to 25.

5.2. Selection. Three different types of selection methods are used in this paper: roulette wheel method (RWS), tournament selection method (TS), and the hybrid combinations (TS + RWS) with different proportions of roulette wheel and tournament selection methods.

5.2.1. Roulette wheel selection method. Each individual in the population is assigned a space on the roulette wheel, which is proportional to the individual relative fitness. Individuals with the largest portion on the wheel have the greatest probability to be selected as parent generation for the next generation.

5.2.2. Tournament selection method. In tournament selection, a number "Tour" of individuals is chosen randomly from the population and the best individual from this group is selected as a parent. This process is repeated as often as individuals to choose. These selected parents produce uniform offspring at random. The parameter for tournament selection is the tournament size Tour. Tour takes values ranging from 2-N_{ind} (number of individuals in population).

5.2.3. Hybrid selection method. The hybrid selection method consists of the combination of both RWS and TS. We designed two types of hybrid selections: single level and two level hybrid selection methods. In single level hybrid selection method, 50% of the population size adopts TS procedure where as the RWS procedure is used in the remaining 50% of the population size. The two level hybrid selection method consists of 25% of TS, then followed by 25% RWS and again 25% TS and 25% RWS.

5.3. Crossover. Crossover is the main genetic operator and consists of swapping chromosome parts between individuals. Crossover is not performed on every pair of individuals; its frequency being controlled by a crossover probability (P_c). There are several crossover methods available and here we use hybrid combination of arithmetic crossover method (AMXO), average convex crossover (ACXO), and direction-based crossover (DBXO).

5.3.1. Arithmetic crossover method (AMXO). The basic concept of this method is borrowed from the convex set theory [9, 13]. Simple arithmetic operators are defined as the combination of two vectors (chromosomes) as follows:

$$\begin{aligned} x' &= \lambda x + (1-\lambda)y, \\ x'' &= (1-\lambda)x + \lambda y, \end{aligned} \tag{5.1}$$

where λ is a uniformly distributed random variable between 0 and 1.

5.4. Population size. From the earlier research, done by Eberhart and Shi [8], it is proved that the performance of the standard algorithm is not sensitive to the population size but to the convergence rate. Based on these results the population size in the experiments was

fixed at 20 particles in order to keep the computational requirements low. The size of the population will affect the convergence of the solution. Hence the population size is set to 20 in this paper.

5.5. Search space. The range in which the algorithm computes the optimal control variables is called search space. The algorithm will search for the optimal solution in the search space which is between 0 and 1. When any of the optimal control values of any particle exceeds the searching space, the value will be reinitialized. In this paper the lower boundary is set to zero and the upper boundary to one.

5.6. Dimension. The dimension is the number of independent variables, which is identical to the number of jobs considered for processing in the hybrid system framework. In this paper, the dimension of the optimal control problem varies between 5 and 25.

5.7. Maximum generations. This refers to the maximum number of generations allowed for the fitness value to converge with the optimal solution. We set 1000 generations for the simulation.

5.8. Boundary. When any of the optimal control value of any particle exceeds the searching space, the value will be reinitialized.

Lower bound = 0, and upper bound = 1.

5.9. Time-varying inertia weight (TVIW). Eberhart and Shi [8] have found a significant improvement in the performance of PSO method with the linearly decreasing inertia weight over the generations, time-varying inertia weight (TVIW). The mathematical representation of TVIW is given in (5.2).

$$\text{Inertia weight } w = (w_i - w_f)\left(\frac{\max\text{iter} - \text{iter}}{\text{iter}}\right) + w_f, \tag{5.2}$$

where w_1 *and* w_2 are the initial and final values of the inertia weight, respectively, iter is the current iteration and max iter is the maximum number of allowable iterations.

5.10. Global local best inertia weight (GLbest IW). In this paper, a new inertia weight is proposed which is neither set to a constant value nor set as a linearly decreasing time-varying function. Instead it is defined as a function of pbest and gbest values of the objective function for each generation and given in (5.3):

$$\text{inertia weight } w_i = \left(1.1 - \frac{\text{gbest}_i}{(\text{pbest}_i)}\right), \tag{5.3}$$

where pbest is the local best value of the particles and gbest is the best among all the local best values in the swarm.

6. Examples and simulations

To test the efficacy of our proposed algorithm, we consider the optimal control problem from (2.5) with the following functions:

$$\theta_i(u_i) = \frac{1}{u_i}, \qquad \phi(x_i) = x_i^2. \tag{6.1}$$

Now (2.5) becomes

$$\min_{u_1,\ldots,u_N} \left\{ J = \sum_{i=1}^{N} \left(\frac{1}{\mathrm{u_i}} + x_i^2 \right) \right\} \tag{6.2}$$
$$\text{subject to } x_i = \max\left(x_{(i-1)}, a_i\right) + u_i.$$

The optimal controls (u_i) and cost or fitness (J) for the objective function given in (6.2) is computed with the following parameter settings. The dimension or the number of jobs involved in the objective function $N = (5, 10, 15, 20, 25)$. The maximum number of generations is set as 1000 with the population size of 20. The crossover probability, P_c, and mutation probability P_m are set to 0.8 and 0.1, respectively.

The arrival sequence (a_i for $i = 1$ to N) is $\{1, 1.2, 1.5, 1.8, 2, 2.2, 2.5, 2.8, 3, 3.2, 3.5, 3.8, 4, 4.2, 4.5, 4.8, 5, 5.2, 5.5, 5.8, 6, 6.2, 6.5, 6.8, 7\}$. The number of arrival times is taken according to the dimension of the problem, that is, the number of jobs considered for processing. In this paper, PSO algorithms with 3 different inertia weights are considered.

Added to these three PSO techniques, three real coded genetic algorithms are also considered in this paper for comparison. The genetic operators used in RCGAs are

(1) roulette-wheel selection (RWS),
(2) tournament selection (TS),
(3) hybrid selection (RWS + TS),
(4) arithmetic crossover (AMXO), and
(5) dynamic mutation (DM).

All the six methods which are listed in Table 6.1 are simulated 1000 times at different periods of time, and their statistical analyses are recorded. The *mean or average* and *standard deviation* (SD) are the basic statistical tests. From these two, the *coefficient of variance* (CV), which is the ratio of standard deviation to mean, is calculated. The fourth statistical test is average deviation (AVEDEV), which will give the average of the absolute deviation of the fitness values from their mean, which are taken in 1000 simulation runs. Added to these analyses, *hypothesis t test and analysis of variance* (ANOVA) test also were carried out to validate the efficacy of all the six methods. These statistics analyses are presented in Tables 6.2–6.6. The graphical analyses are done through box plot, which are shown in Figure 6.2.

A box plot, which is shown in Figure 6.1, provides an excellent visual summary of many important aspects of a distribution. The box stretches from the lower hinge (defined as the 25th percentile) to the upper hinge (the 75th percentile) and therefore contains the middle half of the scores in the distribution.

Table 6.1. Various optimization methods.

Method no.	Method name	Description
1	PSO-1	PSO with constant inertia weight (CIW) = 0.5
2	PSO-2	PSO with linearly decreasing inertia weight (TVIW)
3	PSO-3	PSO with proposed GLbest inertia weight (GLbestIW)
4	RCGA-1	Simple RCGA with TS, AMXO, and DM
5	RCGA-2	Simple RCGA with RWS, AMXO, and DM
6	RCGA-3	Hybrid RCGA with RWS & TS, AMXO, and DM

Table 6.2. Statistical analyses of fitness value for $N = 5$.

Stat. test	Average	SD	CV	AVEDEV	t test for $N = 5$		
PSO-1	34.81632	0.00001	0.00000	0.00000	Method nos.	P value	Best method
PSO-2	34.81631	0.00000	0.00000	0.00000	1 & 2	0.998074	2
PSO-3	34.81631	0.00000	0.00000	0.00000	2 & 3	0.999942	3
RCGA-1	34.84775	0.02246	0.00064	0.01730	3 & 4	0.000000	3
RCGA-2	34.92619	0.06024	0.00172	0.05081	3 & 5	0.000000	3
RCGA-3	34.81637	0.00013	0.00000	0.00098	3 & 6	0.000000	3

Table 6.3. Statistic analyses of fitness value for $N = 10$.

Stat. test	Average	SD	CV	AVEDEV	t test for $N = 10$		
PSO-1	102.10156	0.00476	0.00005	0.00272	Method nos.	P value	Best method
PSO-2	102.10725	0.01614	0.00016	0.00818	1 & 2	0.005011	1
PSO-3	102.09958	0.00058	0.00001	0.00049	1 & 3	0.999118	3
RCGA-1	103.02754	0.42894	0.00416	0.30942	3 & 4	0.000000	3
RCGA-2	103.07814	0.30204	0.00293	0.23756	3 & 5	0.000000	3
RCGA-3	102.10978	0.02283	0.00022	0.09394	3 & 6	0.000000	3

Table 6.4. Statistical analyses of fitness value for $N = 15$.

Stat. test	Average	SD	CV	AVEDEV	t test for $N = 15$		
PSO-1	216.15085	2.43051	0.01124	2.13195	Method nos.	P value	Best method
PSO-2	213.98033	0.05227	0.00025	0.03306	1 & 2	1.000000	2
PSO-3	213.85370	0.00323	0.00002	0.00271	2 & 3	1.000000	3
RCGA-1	215.02073	0.53207	0.00247	0.38085	3 & 4	0.000000	3
RCGA-2	216.36323	0.45778	0.00212	0.37181	3 & 5	0.000000	3
RCGA-3	213.99321	0.06611	0.00031	0.04170	3 & 6	0.000000	3

Table 6.5. Statistical analyses of fitness value for $N = 20$.

Stat. test	Average	SD	CV	AVEDEV	t test for $N = 20$		
PSO-1	461.09201	22.78419	0.04941	19.83020	Method nos.	P value	Best method
PSO-2	387.08515	0.05378	0.00014	0.04239	1 & 2	1.000000	2
PSO-3	387.00120	0.01241	0.00003	0.01039	2 & 3	1.000000	3
RCGA-1	389.21029	0.89738	0.00231	0.65196	3 & 4	0.000000	3
RCGA-2	391.23726	1.11092	0.00284	0.97646	3 & 5	0.000000	3
RCGA-3	387.29281	0.07721	0.00019	0.07281	3 & 6	0.000000	3

Table 6.6. Statistical analyses of fitness value for $N = 25$.

Stat. test	Average	SD	CV	AVEDEV	t test for $N = 25$		
PSO-1	882.39203	38.21507	0.04331	29.67120	Method nos.	P value	Best method
PSO-2	636.98189	0.13835	0.00022	0.08762	1 & 2	1.000000	2
PSO-3	636.59875	0.03011	0.00005	0.02530	2 & 3	0.999993	3
RCGA-1	639.90364	1.12836	0.00176	0.83489	3 & 4	0.000000	3
RCGA-2	642.32081	1.31279	0.00204	1.05927	3 & 5	0.000000	3
RCGA-3	637.12148	0.13012	0.00020	0.10393	3 & 6	0.000000	3

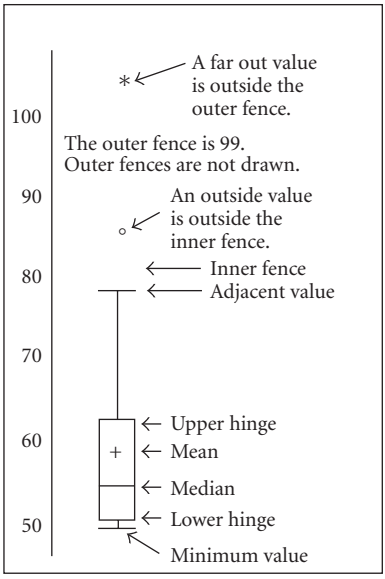
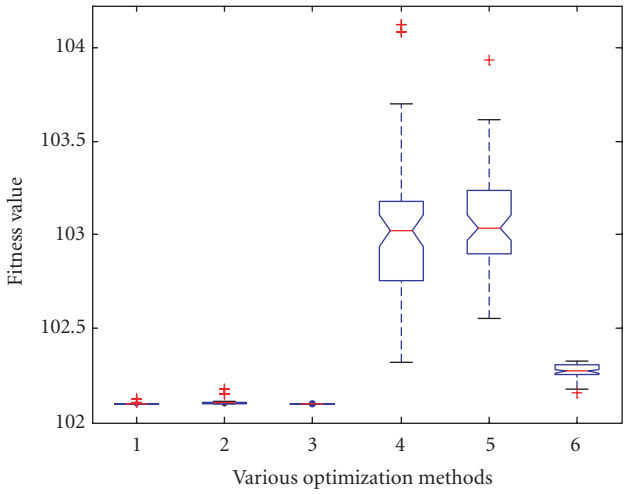
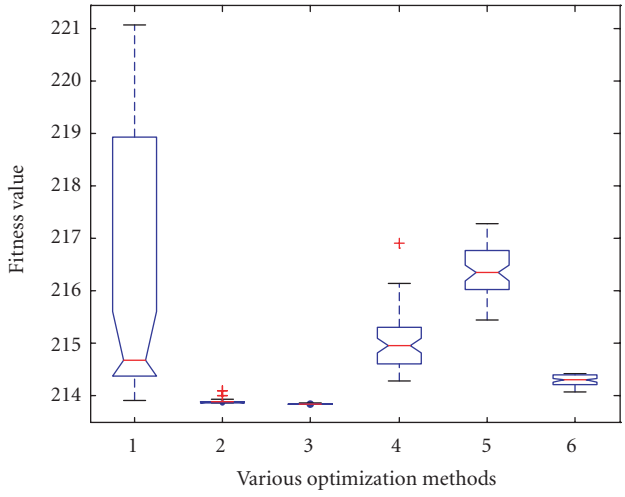


Figure 6.1. A simple box plot.

The median is shown as a line across the box. Therefore 1/4 of the distribution is between this line and the top of the box and 1/4 of the distribution is between this line and the bottom of the box.



(a)



(b)

Figure 6.2. ANOVA t test for all the six methods for (a) $N = 10$ and (b) $N = 15$.

The percentage of deviation of all other optimization methods discussed in this paper with the proposed method, PSO with GLbest inertia (PSO-3), is calculated using (6.3) and presented in Table 6.7. The negative deviation clearly proves the betterment of the proposed method compared to the other methods considered in this paper.

$$\text{Percentage of deviation of the fitness value} = \left(\frac{\text{Fitness of PSO-3} - \text{Fitness of other method}}{\text{Fitness of PSO-3}}\right) \times 100. \tag{6.3}$$

Table 6.7. Percentage of deviation of fitness values from the proposed PSO-3 algorithm.

Number of jobs	Percentage of deviation of fitness values from the proposed PSO-3 algorithm				
	PSO-1	PSO-2	RCGA-1	RCGA-2	RCGA-3
5	0.00000	0.0000	−0.0903	−0.3156	−0.0002
10	−0.00751	−0.0075	−0.9089	−0.9584	−0.0100
15	−0.05921	−0.0218	−0.5457	−1.1735	−0.0637
20	−0.02169	−0.0139	−0.5708	−1.0946	0.0754
25	−0.06019	−0.0131	−0.5191	−0.8988	−0.0821

Table 6.8. Execution times for the PSOAs and RCGAs.

Methods	Execution time (in seconds)				
	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 25$
PSO-1	2.000	2.500	3.000	3.500	4.000
PSO-2	3.000	4.000	4.000	5.000	5.000
PSO-3	3.000	4.000	4.500	4.500	5.500
RCGA-1	3.000	4.500	5.000	6.500	7.500
RCGA-2	4.000	5.000	6.000	7.000	8.500
RCGA-3	4.000	5.000	6.000	7.500	9.000

The execution times for all the evolutionary algorithms considered in this paper, are calculated for $N = 5, 10, 15, 20, 25$ and given in Table 6.8. PSO-1 converges faster than the other methods but delivers only premature results. At the same time, PSO-2 and PSO-3 take little longer than PSO-1 but provide better solutions. When comparing PSO-2 and PSO-3, the latter provides the best solution with almost same convergence speed as the former. Hence it can be concluded that the proposed GLbest IW improves the performance of PSO in all possible manner.

7. Discussions

This paper deals with the optimal control problem for a single-stage manufacturing systems in the hybrid framework. The control variables comprise the processing times of various jobs and the performance metrics involve measures of quality and of time delivery requirements of the completed jobs. Two important EAs (PSO and RCGA) are implemented to solve the optimal control problem with different parameter designs.

For any optimization search algorithm, generally, it is a good idea for the algorithm to possess more exploitation ability at the beginning to find a good seed, and then have the algorithm to have more exploration ability to fine search the local area around the seed. PSO is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions. Conceptually, PSO seems to lie somewhere between genetic

algorithms and evolutionary programming. PSO algorithm solves the optimal control problem by changing the position and velocity of each particle at each time step, to obtain the best optimal solution. The PSO technique provides the natural mechanism to make the particles search actively, which encourages employing the small inertia weight to obtain better optimal solution.

Number of simulations have been performed to illustrate the impact of inertia weight on the performance of PSO in computing the optimal control of a class of hybrid systems. From the simulation results it can be observed that the PSO with the proposed inertia weight, GLbestIW (PSO-3), provides better result compared to the other PSO methods with TVIW (PSO-2) and CIW (PSO-1) and as well as the RCGAs (RCGA 1, 2, and 3).

The experimental results illustrate that the PSO algorithm has the ability to converge faster than RCGAs. The impact of inertia weight on the performance of PSO is analyzed with three different inertia weight variants. When the inertia weight is constant (say I.W = 0.5, PSO-1), the PSO converges faster and yields the premature solution which is not a satisfactory result. But at the same time when the inertia weight is replaced with a linearly decreasing time variant inertia weight (TVIW, PSO-2), the PSO yields a better optimal solution but the convergence of the solution takes little more time. In order to get a compromise between optimal solution and convergence rate (or execution time), a new inertia weight is introduced with PSO (GLbestIW, PSO-3) which comprises global best and local best values of the fitness function. From the simulation results it is observed that PSO-3 provides a better optimal solution to the optimal control problem than any other method considered in this paper. The convergence rate (or the execution time) is also improved over PSO-2 but slightly higher than PSO-1. Since PSO-3 gives a compromise between the optimal solution and the execution time, it can be concluded that PSO-3 is the best method to compute the optimal control of single-stage hybrid manufacturing systems.

It is also proved through the statistical analyses, which are presented in Tables 6.2–6.6, for different dimensions of the optimal control problem. All the statistical parameters, such as mean, standard deviation, C.V, AVEDEV, taken from the 1000 simulated runs of all the six methods considered in this paper, are proven and the efficacy and betterment of the proposed method, PSO-3, is verified. The hypothesis *t* test results and the ANOVA box plots strengthen and justify the superiority of the proposed method. The results here hence indicate that the improved performance of the PSO can be obtained by carefully selecting the inertia weight.

Apart from the PSO, another popular optimization technique, GA, in the form of RCGA, is also considered to solve the optimal control problem in order to get an external comparison for the proposed PSO method. Three different RCGAs are considered and each varies from the other with the composition of genetic operators. Among the three RCGA methods, RCGA-3 which has hybrid composition of the genetic operators seems to be better. From the results listed in Tables 6.2–6.6, it can be observed that RCGAs are superior to PSO-1, especially for higher dimension problems, but fails to maintain the superiority with PSO-2 and PSO-3 where the latter is the proposed method. However, the comparison of the results with RCGAs, strengthen the validity and efficacy of the PSO methods and particularly the proposed method.

8. Conclusions

In summary, it can be concluded without any prejudice that PSO performs significantly well on high-dimensional optimal problems with great speed, reliability, and accuracy than RCGAs. Also, the impact of the inertia weight in improving the performance of the PSO towards obtaining the optimal solution for the optimal control of the single-stage hybrid system is clearly seen. The new concept of defining the inertia weight in terms of the personal and global best values helps the PSO to perform better in solving any high-dimensional optimal control problem with faster convergence and accuracy. The superiority of the proposed method is hence proved by all means. Further improvements in the performance of PSO algorithms will be investigated in future works.

References

[1] M. S. Arumugam and M. V. C. Rao, *Novel hybrid approaches for real coded genetic algorithm to compute the optimal control of a single stage hybrid manufacturing systems*, International Journal of Computational Intelligence **1** (2004), no. 3, 231–249.

[2] C. G. Cassandras, D. L. Pepyne, and Y. Wardi, *Optimal control of a class of hybrid systems*, IEEE Transactions on Automatic Control **46** (2001), no. 3, 398–415.

[3] Y. C. Cho, C. G. Cassandras, and D. L. Pepyne, *Forward decomposition algorithms for optimal control of a class of hybrid systems*, International Journal of Robust and Nonlinear Control **11** (2001), no. 5, 497–513.

[4] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Canadian Mathematical Society Series of Monographs and Advanced Texts, John Wiley & Sons, New York, 1983.

[5] L. Davis, *A Hand Book of Genetic Algorithms*, New York, 1990.

[6] R. C. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory*, Proceedings of the Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, New Jersey, 1995, pp. 39–43.

[7] R. C. Eberhart, J. Kennedy, and P. Simpson, *Computational Intelligence PC Tools*, Academic Press, Massachusetts, 1996.

[8] R. C. Eberhart and Y. Shi, *Comparison between genetic algorithms and particle swarm optimization*, Evolutionary Programming VII, Lecture Notes in Computer Science, vol. 1447, Springer, New York, 1998, pp. 611–616.

[9] M. Gen and R. Cheng, *Genetic Algorithms & Engineering Design*, John Wiley & Sons, New York, 1997.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Massachusetts, 1989.

[11] J. Kennedy, *The particle swarm: social adaptation of knowledge*, Proceedings of the IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, New Jersey, 1997, pp. 303–308.

[12] J. Kennedy and R. C. Eberhart, *Particle swarm optimization*, Proceedings of the IEEE Internation Conference on Neural Networks (Perth, Australia), vol. 4, IEEE Service Center, New Jersey, 1995, pp. 1942–1948.

[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed., Springer, Berlin, 1994.

[14] D. L. Pepyne and C. G. Cassandras, *Modeling, analysis, and optimal control of a class of hybrid systems*, Discrete Event Dynamic Systems: Theory and Applications **8** (1998), no. 2, 175–201.

[15] Y. Shi and R. C. Eberhart, *A modified particle swarm optimizer*, Proceedings of the IEEE International Conference on Evolutionary Computation, Alaska, May 1998.

[16] ———, *Parameter selection in particle swarm optimization*, Evolutionary Programming VII, Lecture Notes in Computer Science, vol. 1447, Springer, New York, 1998, pp. 591–600.

[17] Y. Wardi, C. G. Cassandras, and D. L. Pepyne, *A backward algorithm for computing optimal controls for single-stage hybrid manufacturing systems*, International Journal of Production Research **39** (2001), no. 2, 369–393.

[18] P. Zhang and C. G. Cassandras, *An improved forward algorithm for optimal control of a class of hybrid systems*, IEEE Transactions on Automatic Control **47** (2002), no. 10, 1735–1739.

M. Senthil Arumugam: Faculty of Engineering and Technology, Multimedia University,
Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
E-mail address: msenthil.arumugam@mmu.edu.my

M. V. C. Rao: Faculty of Engineering and Technology, Multimedia University,
Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
E-mail address: machavaram.venkata@mmu.edu.my