A Hybrid Global Optimization Algorithm: Particle Swarm Optimization in Association with a Genetic Algorithm

M. Andalib Sahnehsaraei, M.J. Mahmoodabadi, M. Taherkhorsandi, K.K. Castillo-Villar and S.M. Mortazavi Yazdi

Abstract The genetic algorithm (GA) is an evolutionary optimization algorithm operating based upon reproduction, crossover and mutation. On the other hand, particle swarm optimization (PSO) is a swarm intelligence algorithm functioning by means of inertia weight, learning factors and the mutation probability based upon fuzzy rules. In this paper, particle swarm optimization in association with genetic algorithm optimization is utilized to gain the unique benefits of each optimization algorithm. Therefore, the proposed hybrid algorithm makes use of the functions and operations of both algorithms such as mutation, traditional or classical crossover, multiple-crossover and the PSO formula. Selection of these operators is based on a fuzzy probability. The performance of the hybrid algorithm in the case of solving both single-objective and multi-objective optimization problems is evaluated by utilizing challenging prominent benchmark problems including FON, ZDT1, ZDT2, ZDT3, Sphere, Schwefel 2.22, Schwefel 1.2, Rosenbrock, Noise, Step, Rastrigin, Griewank, Ackley and especially the design of the parameters of linear feedback control for a parallel-double-inverted pendulum system which is a complicated, nonlinear and unstable system. Obtained numerical results in comparison to the outcomes of other optimization algorithms in the literature demonstrate the

M. Andalib Sahnehsaraei
Department of Mechanical Engineering, Faculty of Engineering,
The University of Guilan, Rasht, Iran
e-mail: morteza_andalib@yahoo.com

M.J. Mahmoodabadi · S.M. Mortazavi Yazdi
Department of Mechanical Engineering, Sirjan University of Technology, Sirjan, Iran
e-mail: mahmoodabadi@sirjantech.ac.ir

S.M. Mortazavi Yazdi
e-mail: m.mechanic88@yahoo.com

M. Taherkhorsandi (✉) · K.K. Castillo-Villar
Department of Mechanical Engineering, The University of Texas at San Antonio,
San Antonio, TX 78249, USA
e-mail: m.taherkhorsandi@gmail.com; milad.taherkhorsandi@utsa.edu

K.K. Castillo-Villar
e-mail: krystel.castillo@utsa.edu

efficiency of the hybrid of particle swarm optimization and genetic algorithm optimization with regard to addressing both single-objective and multi-objective optimization problems.

Keywords Particle swarm optimization · Genetic algorithm optimization · Single-objective problems · Multi-objective problems · State feedback control · Parallel-double-inverted pendulum system

1 Introduction

Optimization is the selection of the best element from some sets of variables with a long history dating back to the years when Euclid conducted research to gain the minimum distance between a point and a line. Today, optimization has an extensive application in different branches of science, e.g. aerodynamics (Song et al. 2012), robotics (Li et al. 2013; Cordella et al. 2012), energy consumption (Wang et al. 2014), supply chain modeling (Yang et al. 2014; Castillo-Villar et al. 2014) and control (Mahmoodabadi et al. 2014a; Wang and Liu 2012; Wibowo and Jeong 2013). Due to the necessity of addressing a variety of constrained and unconstrained optimization problems, many changes and novelties in optimization approaches and techniques have been proposed during the recent decade. In general, optimization algorithms are divided into two main classifications: deterministic and stochastic algorithms (Blake 1989). Due to employing the methods of successive search based upon the derivative of objective functions, deterministic optimization algorithms are appropriate for convex, continuous and differentiable objective functions. On the other hand, stochastic optimization techniques are applicable to address most of real optimization problems, which are heavily non-linear, complex and non-differentiable. In this regard, a great number of studies have recently been devoted to stochastic optimization algorithms, especially, genetic algorithm optimization and particle swarm optimization.

The genetic algorithm, which is a subclass of evolutionary algorithms, is an optimization technique inspired by natural evolution, that is, mutation, inheritance, selection and crossover to gain optimal solutions. Lately, it was enhanced by using a novel multi-parent crossover and a diversity operator instead of mutation in order to gain quick convergence (Elsayed et al. 2014), utilizing it in conjunction with several features selection techniques, involving principle components analysis, sequential floating, and correlation-based feature selection (Aziz et al. 2013), using the controlled elitism and dynamic crowding distance to present a general algorithm for the multi-objective optimization of wind turbines (Wang et al. 2011), and utilizing a real encoded crossover and mutation operator to gain the near global optimal solution of multimodal nonlinear optimization problems (Thakur 2014). Particle swarm optimization is a population-based optimization algorithm mimicking the behavior of social species such as flocking birds, swimming wasps,

school fish, among others. Recently, its performance was enhanced by using a multi-stage clustering procedure splitting the particles of the main swarm over a number of sub-swarms based upon the values of objective functions and the particles positions (Nickabadi et al. 2012), utilizing multiple ranking criteria to define three global bests of the swarm as well as employing fuzzy variables to evaluate the objective function and constraints of the problem (Wang and Zheng 2012), employing an innovative method to choose the global and personal best positions to enhance the rate of convergence and diversity of solutions (Mahmoodabadi et al. 2014b), and using a self-clustering algorithm to divide the particle swarm into multiple tribes and choosing appropriate evolution techniques to update each particle (Chen and Liao 2014).

Lately, researchers have utilized hybrid optimization algorithms to provide more robust optimization algorithms due to the fact that each algorithm has its own advantages and drawbacks and it is not feasible that an optimization algorithm can address all optimization problems. Particularly, Ahmadi et al. (2013) predicted the power in the solar stirling heat engine by using neural network based on the hybrid of genetic algorithm and particle swarm optimization. Elshazly et al. (2013) proposed a hybrid system which integrates rough set and the genetic algorithm for the efficient classification of medical data sets of different sizes and dimensionalities. Abdel-Kader (2010) proposed an improved PSO algorithm for efficient data clustering. Altun (2013) utilized a combination of genetic algorithm, particle swarm optimization and neural network for the palm-print recognition. Zhou et al. (2012) designed a remanufacturing closed-loop supply chain network based on the genetic particle swarm optimization algorithm. Jeong et al. (2009) developed a hybrid algorithm based on genetic algorithm and particle swarm optimization and applied it for a real-world optimization problem. Mavaddaty and Ebrahimzadeh (2011) used the genetic algorithm and particle swarm optimization based on mutual information for blind signals separation. Samarghandi and ElMekkawy (2012) applied the genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and make-span criterion. Deb and Padhye (2013) enhanced the performance of particle swarm optimization through an algorithmic link with genetic algorithms. Valdez et al. (2009) combined particle swarm optimization and genetic algorithms using fuzzy logic for decision making. Premalatha and Natarajan (2009) applied discrete particle swarm optimization with genetic algorithm operators for document clustering. Dhadwal et al. (2014) advanced particle swarm assisted genetic algorithm for constrained optimization problems. Bhuvaneswari et al. (2009) combined the genetic algorithm and particle swarm optimization for alternator design. Jamili et al. (2011) proposed a hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem. Joeng et al. (2009) proposed a sophisticated hybrid of particle swarm optimization and the genetic algorithm which shows robust search ability regardless of the selection of the initial population and compared its capability to a simple hybrid of particle swarm optimization and the genetic algorithm and pure particle swarm optimization and pure the genetic algorithm. Castillo-Villar et al. (2012) used genetic algorithm optimization and simulated annealing for a

model of supply-chain design regarding the cost of quality and the traditional manufacturing and distribution costs. Talatahari and Kaveh (2007) used a hybrid of particle swarm optimization and ant colony optimization for the design of frame structures. Thangaraj et al. (2011) presented a comprehensive list of hybrid algorithms of particle swarm optimization with other evolutionary algorithms e.g. the genetic algorithm. Valdez et al. (2011) utilized the fuzzy logic to combine the results of the particle swarm optimization and the genetic algorithm. This method has been performed on some single-objective test functions for four different dimensions contrasted to the genetic algorithm and particle swarm optimization, separately.

For the optimum design of traditional controllers, the evolutionary optimization techniques are appropriate approaches to be used. To this end, Fleming and Purshouse (2002) is an appropriate reference to overview the application of the evolutionary algorithms in the field of the design of controllers. In particular, the design of controllers in Fonseca and Fleming (1994) and Sanchez et al. (2007) was formulated as a multi-objective optimization problem and solved using genetic algorithms. Furthermore, in Ker-Wei and Shang-Chang (2006), the sliding mode control configurations were designed for an alternating current servo motor while a particle swarm optimization algorithm was used to select the parameters of the controller. Also, PSO was applied to tune the linear control gains in Gaing (2004) and Qiao et al. (2006). In Chen et al. (2009), three parameters associated with the control law of the sliding mode controller for the inverted pendulum system were properly chosen by a modified PSO algorithm. Wai et al. (2007) proposed a total sliding-model-based particle swarm optimization to design a controller for the linear induction motor. More recently, in Gosh et al. (2011), an ecologically inspired direct search method was applied to solve the optimal control problems with Bezier parameterization. Moreover, in Tang et al. (2011), a controllable probabilistic particle swarm optimization (CPPSO) algorithm was applied to design a memory-less feedback controller. McGookin et al. (2000) optimized a tanker autopilot control system using genetic algorithms. Gaing (2004) used particle swarm optimization to tune linear gains of the proportional-integral-derivative (PID) controller for an AVR system. Qiao et al. (2006) tuned the proportional-integral (PI) controller parameters for doubly fed induction generators driven by wind turbines using PSO. Zhao and Yi (2006) proposed a GA-based control method to swing up an acrobot with limited torque. Wang et al. (2006) designed a PI/PD controller for the non-minimum phase system and used PSO to tune the controller gains. Sanchez et al. (2007) formulated a classical observer-based feedback controller as a multi-objective optimization problem and solved it using genetic algorithm. Mohammadi et al. (2011) applied an evolutionary tuning technique for a type-2 fuzzy logic controller and state feedback tracking control of a biped robot (Mahmoodabadi et al. 2014c). Zargari et al. (2012) designed a fuzzy sliding mode controller with a Kalman estimator for a small hydro-power plant based on particle swarm optimization. More recently, a two-stage hybrid optimization algorithm, which involves the combination of PSO and a pattern search based method, is used to tune a PI controller (Puri and Ghosh 2013).

In this chapter, a hybrid of particle swarm optimization and the genetic algorithm developed previously by authors (Mahmoodabadi et al. 2013) is described and used to design the parameters of the linear state feedback controller for the system of a parallel-double-inverted pendulum. In elaboration, the used operators of the hybrid algorithm include mutation, crossover of the genetic algorithm and particle swarm optimization formula. The classical crossover and the multiple-crossover are two parts of the crossover operator. A fuzzy probability is used to choose the particle swarm optimization and genetic algorithm operators at each iteration for each particle or chromosome. The optimization algorithm is based on the non-dominated sorting concept. Moreover, a leader selection method based upon particles density and a dynamic elimination method which confines the numbers of non-dominated solutions are utilized to present a high convergence and uniform spread. Single and multi-objective problems are utilized to assess the capabilities of the optimization algorithm. By using the same benchmarks, the results of simulation are contrasted to the results of other optimization algorithms. The structure of this chapter is as follows. Section 2 presents the genetic algorithm and its details including the crossover operator and the mutation operator. Particle swarm optimization and its details involving inertia weight and learning factors are provided in Sect. 3. Section 4 states the mutation probabilities at each iteration which is based on fuzzy rules. Section 5 includes the architecture, the pseudo code, the parameter settings, and the flow chart of the single-objective and multi-objective hybrid optimization algorithms. Furthermore, the test functions and the evolutionary trajectory for the algorithms are provided in Sect. 5. State feedback control for linear systems is presented in Sect. 6. Section 7 presents the state space representation, the block diagram, and the Pareto front of optimal state feedback control of a parallel-double-inverted pendulum. Finally, conclusions are presented in Sect. 8.

2 Genetic Algorithm

The genetic algorithm inspired from Darwin's theory is a stochastic algorithm based upon the survival fittest introduced in 1975 (Holland 1975).

Genetic algorithms offer several attractive features, as follows:

- An easy-to-understand approach that can be applied to a wide range of problems with little or no modification. Other approaches have required substantial alteration to be successfully used in applications. For example, the dynamic programming was applied to select the number, location and power of the lamps along a hallway in such a way that the electrical power needed to produce the required illuminance will be minimized. In this method, significant alternation is needed since the choice of the location and power of a lamp affect the decisions made about previous lamps (Gero and Radford 1978).
- Genetic algorithm codes are publicly available which reduces set-up time.
- The inherent capability to work with complex simulation programs. Simulation does not need to be simplified to accommodate optimization.

- Proven effectiveness in solving complex problems that cannot be readily solved with other optimization methods. The mapping of the objective function for a day lighting design problem showed the existence of local minima that would potentially trap a gradient-based method (Chutarat 2001). Building optimization problems may include a mixture of a large number of integer and continuous variables, non-linear inequality and equality constraints, a discontinuous objective function and variables embedded in constraints. Such characteristics make gradient-based optimization methods inappropriate and restrict the applicability of direct-search methods (Wright and Farmani 2001). The calculation time of mixed integer programming (MIP), which was used to optimize the operation of a district heating and cooling plant, increases exponentially with the number of integer variables. It was shown that it takes about two times longer than a genetic algorithm for a 14 h optimization window and 12 times longer for a 24 h period (Sakamoto et al. 1999), although the time required by MIP was sufficiently fast for a relatively simple plant to make on-line use feasible.
- Methods to allow genetic algorithms to handle constraints that would make some solutions unattractive or entirely infeasible.

Performing on a set of solutions instead of one solution is one of notable abilities of stochastic algorithms. Thus, at first, initial population consisting of a random set of solutions is generated by the genetic algorithm. Each solution in a population is named an individual or a chromosome. The size of population (N) is the number of chromosomes in a population. The genetic algorithm has the capability of performing with coded variables. In fact, the binary coding is the most popular approach of encoding the genetic algorithm. When the initial population is generated, the genetic algorithm has to encode the whole parameters as binary digits. Hence, while performing over a set of binary solutions, the genetic algorithm must decode all the solutions to report the optimal solutions. To this end, a real-coded genetic algorithm is utilized in this study (Mahmoodabadi et al. 2013). In the real coded genetic algorithm, the solutions are applied as real values. Thus, the genetic algorithm does not have to devote a great deal of time to coding and decoding the values (Arumugam et al. 2005). Fitness which is a value assigned to each chromosome is used in the genetic algorithm to provide the ability of evaluating the new population with respect to the previous population at any iteration. To gain the fitness value of each chromosome, the same chromosome is used to obtain the value of the function which must be optimized. This function is the objective function. Three operators, that is, reproduction, crossover and mutation are employed in the genetic algorithm to generate a new population in comparison to the previous population. Each chromosome in new and previous populations is named offspring and parent, correspondingly. This process of the genetic algorithm is iterated until the stopping criterion is satisfied and the chromosome with the best fitness in the last generation is proposed as the optimal solution. In the present study, crossover and mutation are hybridized with the formula of particle swarm optimization (Mahmoodabadi et al. 2013). The details of these genetic operators are elaborated in the following sections.

2.1 The Crossover Operator

The role of crossover operator is to generate new individuals, that is, offspring from parents in the mating pool. Afterward, two offspring are generated based upon the selected parents and will be put in the place of the parents. Moreover, this operator is used for a number of pair of parents to mate (Chang 2007). This number is calculated by using the formula as $\frac{P_{tc} \times N}{2}$, where P_{tc} and N denote the probability of traditional crossover and population size, correspondingly. By regarding $\overrightarrow{x_i}(t)$ and $\overrightarrow{x_j}(t)$ as two random selected chromosomes in such a way that $\overrightarrow{x_i}(t)$ has a smaller fitness value than $\overrightarrow{x_j}(t)$, the traditional crossover formula is as follows

$$
\begin{aligned}
\overrightarrow{x_i}(t+1) &= \overrightarrow{x_i}(t) + \gamma_1(\overrightarrow{x_i}(t) - \overrightarrow{x_j}(t)) \\
\overrightarrow{x_j}(t+1) &= \overrightarrow{x_j}(t) + \gamma_2(\overrightarrow{x_i}(t) - \overrightarrow{x_j}(t))
\end{aligned}
\tag{1}
$$

where γ_1 and $\gamma_2 \in [0, 1]$ represent random values. When Eq. (1) is calculated, between $\vec{x}(t)$ and $\vec{x}(t+1)$, whichever has the fewer fitness should be chosen. Another crossover operator called multiple-crossover operator is employed in this paper (Mahmoodabadi et al. 2013). This operator was presented in (Ker-Wei and Shang-Chang 2006) for the first time. The multiple-crossover operator consists of three chromosomes. The number of $\frac{P_{mc} \times N}{3}$ chromosomes is chosen for adjusting in which P_{mc} denotes the probability of multiple-crossover. Furthermore, $\overrightarrow{x_i}(t)$, $\overrightarrow{x_j}(t)$ and $\overrightarrow{x_k}(t)$ denote three random chosen chromosomes in which $\overrightarrow{x_i}(t)$ has the smallest fitness value among these chromosomes. Multiple-crossover is computed as follows

$$
\begin{aligned}
\overrightarrow{x_i}(t+1) &= \overrightarrow{x_i}(t) + \lambda_1(2\overrightarrow{x_i}(t) - \overrightarrow{x_j}(t) - \overrightarrow{x_k}(t)) \\
\overrightarrow{x_j}(t+1) &= \overrightarrow{x_j}(t) + \lambda_2(2\overrightarrow{x_i}(t) - \overrightarrow{x_j}(t) - \overrightarrow{x_k}(t)) \\
\overrightarrow{x_k}(t+1) &= \overrightarrow{x_k}(t) + \lambda_3(2\overrightarrow{x_i}(t) - \overrightarrow{x_j}(t) - \overrightarrow{x_k}(t))
\end{aligned}
\tag{2}
$$

where λ_1, λ_2, and $\lambda_3 \in [0, 1]$ represent random values. When Eq. (2) is computed, between $\vec{x}(t)$ and $\overrightarrow{x}(t+1)$, whichever has the fewer fitness should be selected.

2.2 The Mutation Operator

According to the searching behavior of GA, falling into the local minimum points is unavoidable when the chromosomes are trying to find the global optimum solution. In fact, after several generations, chromosomes will gather in several areas or even just in one area. In this state, the population will stop progressing and it will become unable to generate new solutions. This behavior could lead to the whole population being trapped in the local minima. Here, in order to allow the chromosomes' exploration in the area to produce more potential solutions and to explore new regions of the parameter space, the mutation operator is applied. The role of this

operator is to change the value of the number of chromosomes in the population. This number is calculated via $P_m \times N$, in which P_m and N are the probability of mutation and population size, correspondingly. In this regard, a variety in population and a decrease in the possibility of convergence toward local optima are gained through this operation. By regarding a randomly chosen chromosome, the mutation formula is obtained as (Mahmoodabadi et al. 2013):

$$\overrightarrow{x_i}(t+1) = \vec{x}_{\min}(t) + \upsilon(\vec{x}_{\max}(t) - \vec{x}_{\min}(t)) \tag{3}$$

in which $\overrightarrow{x_i}(t)$, $\vec{x}_{\max}(t)$ and $\vec{x}_{\min}(t)$ present the randomly chosen chromosome, upper bound and lower bound with regard to search domain, correspondingly and $\upsilon \in [0, 1]$ is a random value. When Eq. (3) is calculated, between $\vec{x}(t)$ and $\vec{x}(t+1)$, whichever has the fewer fitness should be chosen.

The second optimization algorithm used for the hybrid algorithm is particle swarm optimization and this algorithm and its details involving the inertia weight and learning factors will be presented in the following section.

3 Particle Swarm Optimization (PSO)

Particle swarm optimization introduced by Kennedy and Eberhart (1995) is a population-based search algorithm based upon the simulation of the social behavior of flocks of birds. While this algorithm was first used to balance the weights in neural networks (Eberhart et al. 1996), it is now a very popular global optimization algorithm for problems where the decision variables are real numbers (Engelbrecht 2002, 2005).

In particle swarm optimization, particles are flying through hyper-dimensional search space and the changes in their way are based upon the social-psychological tendency of individuals to mimic the success of other individuals. Here, the PSO operator adjusted the value of positions of particles which are not chosen for genetic operators (Mahmoodabadi et al. 2013). In fact, the positions of these particles are adjusted based upon their own and neighbors' experience. $\overrightarrow{x_i}(t)$ represents the position of a particle and it is adjusted through adding a velocity $\overrightarrow{v_i}(t)$ to it, that is to say:

$$\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t+1) \tag{4}$$

The socially exchanged information is presented by a velocity vector defined as follows:

$$\overrightarrow{v_i}(t+1) = W\overrightarrow{v_i}(t) + C_1 r_1(\vec{x}_{pbest_i} - \overrightarrow{x_i}(t)) + C_2 r_2(\vec{x}_{gbest} - \overrightarrow{x_i}(t)) \tag{5}$$

where C_1 represents the cognitive learning factor and denotes the attraction that a particle has toward its own success. C_2 is the social learning factor and represents the attraction that a particle has toward the success of the entire swarm. W is the

inertia weight utilized to control the impact of the previous history of velocities on the current velocity of a given particle. $\vec{x}_{pbest_i}$ denotes the personal best position of the particle i. $\vec{x}_{gbest}$ represents the position of the best particle of the entire swarm. $r_1, r_2 \in [0, 1]$ are random values. Moreover, in this paper, a uniform probability distribution is assumed for all the random parameters (Mahmoodabadi et al. 2013). The trade-off between the global and local search abilities of the swarm is adjusted by using the parameter W. An appropriate value of the inertia weight balances between global and local search abilities by regarding the fact that a large inertia weight helps the global search and a small one helps the local search. Based upon experimental results, linearly decreasing the inertia weight over iterations enhances the efficiency of particle swarm optimization (Eberhart and Kennedy 1995). The particles approach to the best particle of the entire swarm ($\vec{x}_{gbest}$) via using a small value of C_1 and a large value of C_2. On the other hand, the particles converge into their personal best position ($\vec{x}_{pbest_i}$) through employing a large value of C_1 and a small value of C_2. Furthermore, it was obtained that the best solutions were gained via a linearly decreasing C_1 and a linearly enhancing C_2 over iterations (Ratnaweera et al. 2004). Thus, the following linear formulation of inertia weight and learning factors are utilized as follows:

$$W_1 = W_1 - (W_1 - W_2) \times \left(\frac{t}{maximum\ iteration}\right) \tag{6}$$

$$C_1 = C_{1i} - (C_{1i} - C_{1f}) \times \left(\frac{t}{maximum\ iteration}\right) \tag{7}$$

$$C_2 = C_{2i} - (C_{2i} - C_{2f}) \times \left(\frac{t}{maximum\ iteration}\right) \tag{8}$$

in which W_1 and W_2 represent the initial and final values of the inertia weight, correspondingly. C_{1i} and C_{2i} denote the initial values of the learning factors C_1 and C_2, correspondingly. C_{1f} and C_{2f} represent the final values of the learning factors C_1 and C_2, respectively. t is the current number of iteration and *maximum iteration* is the maximum number of allowable iterations. The mutation probabilities at each iteration which is based on fuzzy rules will be presented in the next section.

4 The Mutation Probabilities Based on Fuzzy Rules

The mutation probability at each iteration is calculated via using the following equation:

$$P_m = \zeta_m \times Limit \tag{9}$$

in which ζ_m is a positive constant. *Limit* represents the maximum number of iteration restraining the changes in positions of the particles or chromosomes of the entire swarm or population. Equations (10) and (11) are utilized to compute other probabilities, as follows:

$$P_{tc} = \xi_{tc} \times P_g \tag{10}$$

$$P_{mc} = \xi_{mc} \times P_g \tag{11}$$

in which ξ_{tc} and ξ_{mc} are positive constants. P_g denotes a fuzzy variable and its membership functions and fuzzy rules are presented in Fig. 1 and Table 1.

The inference result P_g of the consequent variable can be computed via employing the min-max-gravity method, or the simplified inference method, or the product-sum-gravity method (Mizumoto 1996).

Single objective and multi-objective hybrid algorithms based on particle swarm optimization and the genetic algorithm and the details including the pseudo code, the parameter settings will be presented in the following section. These algorithms will be evaluated with many challenging test functions.

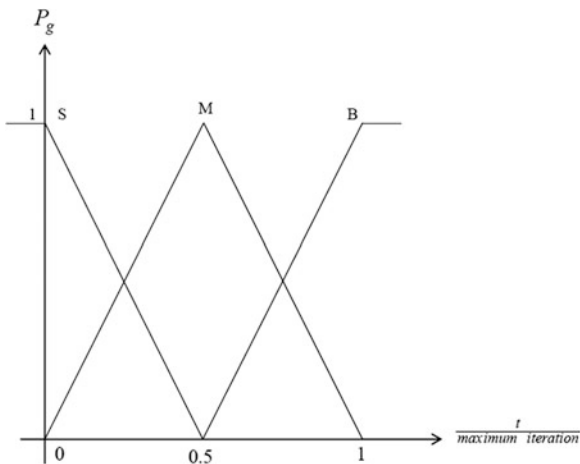

Fig. 1 Membership functions of fuzzy variable P_g

Table 1 Fuzzy rules of fuzzy variable P_g

Antecedent variable	Consequence variable
S	0.0
M	0.5
B	1.0

5 Optimization

Optimization is mathematical and numerical approaches to gain and identify the best candidate among a range of alternatives without having to explicitly enumerate and evaluate all possible alternatives (Ravindran et al. 2006). While maximum or minimum solutions of objective functions are the optimal solutions of an optimization problem, optimization algorithms are usually trying to address a minimization problem. In this regard, the goal of optimization is to gain the optimal solutions which are the points minimizing the objective functions. Based upon the number of objective functions, an optimization problem is classified as single-objective and multi-objective problems. This study uses both single-objective and multi-objective optimization algorithms to evaluate the capabilities of the hybrid of particle swarm optimization and the genetic algorithm (Mahmoodabadi et al. 2013). To this end, challenging benchmarks of the field of optimization are chosen to evaluate the optimization algorithm. The hybrid of particle swarm optimization and the genetic algorithm is applied to these benchmarks and the obtained results are compared to the obtained results of running a number of similar algorithms on the same benchmark problems.

5.1 Single-Objective Optimization

5.1.1 Definition of Single-Objective Optimization Problem

A single-objective optimization problem involves just one objective function as there are many engineering problems where designers combine several objective functions into one. Each objective function can include one or more variables. A single-objective optimization problem can be defined as follow:

$$\text{Find } \overrightarrow{x}^* = [x_1^*, x_2^*, \ldots, x_n^*] \in R^n$$
$$\text{To minimize } f(\vec{x})$$

By regarding p equality constraints $g_i(\vec{x}) = 0$, $i = 1, 2, \ldots, p$; and q inequality constraints $h_j(\vec{x}) \leq 0$, $j = 1, 2, \ldots, q$, where $\vec{x}$ represents the vector of decision variables and $f(\vec{x})$ denotes the objective function.

5.1.2 The Architecture of the Algorithm of Single-Objective Optimization

In this section, a single-objective optimization algorithm is used which is based on a hybrid of genetic operators and PSO formula to update the chromosomes and particle positions (Mahmoodabadi et al. 2013). In elaboration, the initial population

is randomly produced. At each iteration, the inertia weight, the learning factors, and the operators probabilities are computed. Afterward, $\overrightarrow{x}_{pbest_i}$ and $\overrightarrow{x}_{gbest}$ are specified when the fitness values of all members are evaluated. The genetic algorithm operators, that is, traditional crossover, multiple-crossover and mutation operators are used to adjust the chromosomes (which are chosen randomly). Each chromosome is a particle and the group of chromosome is regarded as a swarm. Furthermore, the chromosomes which are not chosen for genetic operations will be appointed to particles and improved via PSO. Until the user-defined stopping criterion is satisfied, this cycle is repeated. Figures 2 and 3 illustrate the pseudo code and flow chart of the technique respectively.

5.1.3 The Results of Single-Objective Optimization

To evaluate the performance of the hybrid of particle swarm optimization and the genetic algorithm, nine prominent benchmark problems are utilized regarding a single-objective optimization problem. Essential information about these functions is summarized in Table 2 (Yao et al. 1999). Some of these functions are unimodal and the others are multimodal. Unimodal functions have only one optimal point while multimodal functions have some local optimal points in addition to one global optimal point.

The hybrid of particle swarm optimization and the genetic algorithm is applied to all the test functions with 30 dimensions ($n = 30$) (Mahmoodabadi et al. 2013). The mean and standard deviation of the best solution for thirty runs are presented in Table 4. In this regard, the results are contrasted to the results of three other algorithms [i.e., GA with traditional crossover (Chang 2007), GA with multiple-crossover (Chang 2007; Chen and Chang 2009), standard PSO (Kennedy and Eberhart 1995)]. Table 3 illustrates the list of essential parameters to run these

Initialize population and determine the algorithm configuration of the hybrid method.
While stopping criterion is satisfied

      Determine P_{tc}, P_{mc}, P_m based on the hybrid formulations.

      Find fitness values of each member and store $\overrightarrow{x}_{pbest_i}$ and $\overrightarrow{x}_{gbest}$

      If $rand < P_{tc}$

         Select randomly two chromosomes from population and update them using traditional crossover operator;

      Elseif $rand < P_{mc}$

         Select randomly three chromosomes from population and update them using multiple-crossover operator;

        Elseif $rand < P_m$

         Select randomly a chromosome from population and update it using mutation operator;

        Else

         Select randomly a particle from swarm and update its position based on PSO formula;

End.

Fig. 2 The pseudo code of the hybrid algorithm for single-objective optimization

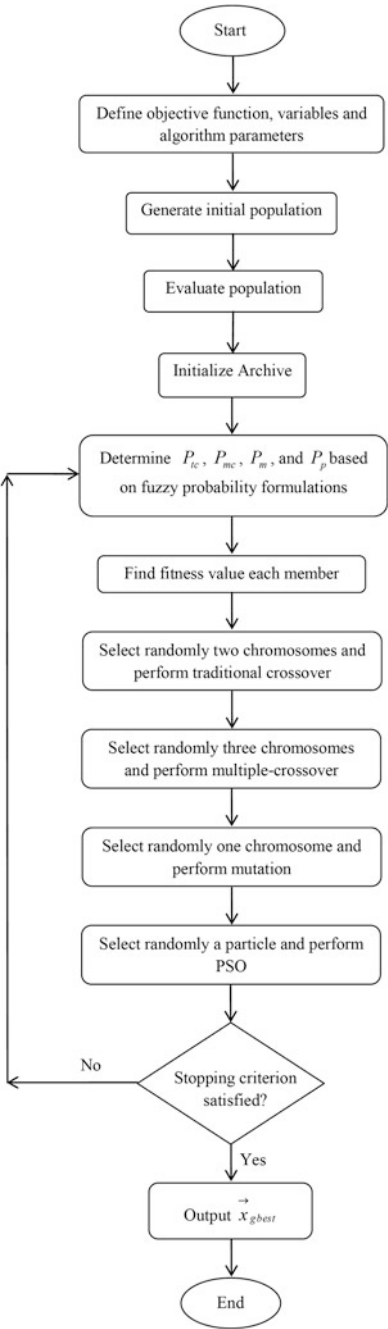**Fig. 3** The flow chart of the hybrid algorithm for single-objective optimization

Table 2 Single-objective test functions

Name (comment)	Formula: $f(x)$	Search domain
Sphere (unimodal)	$\sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
Schwefel 2.22 (unimodal)	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
Schwefel 1.2 (unimodal)	$\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
Rosenbrock (unimodal)	$\sum_{i=1}^{n-1} \left[100\left(x_{i+1}-x_i^2\right)^2 + \left(x_i-1\right)^2 \right]$	$[-30, 30]^n$
Noise (unimodal)	$\sum_{i=1}^{n} i x_i^4 + random[0,1)$	$[-1.28, 1.28]^n$
Step (unimodal)	$\sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	$[-100, 100]^n$
Rastrigin (multimodal)	$\sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$	$[-5.12, 5.12]^n$
Griewank (multimodal)	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$
Ackley (multimodal)	$20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$ $- \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos\left(2\pi x_i\right)\right)$	$[-32, 32]^n$

Table 3 The parameter settings of optimization algorithms

Algorithm	Parameter
GA (traditional crossover)	$P_r = 0.2$, $P_c = 0.4$, $P_m = 0.1$, $S = 0.05$, tournament method for selection
GA (multiple-crossover)	$P_r = 0.2$, $P_c = 0.4$, $P_m = 0.1$, $S = 0.05$, tournament method for selection
Standard PSO	$W = 0.9$, $C_1 = C_2 = 2$
The hybrid algorithm	$W_1 = 0.9$, $W_2 = 0.4$, $C_{1i} = C_{2f} = 2.5$, $C_{1f} = C_{2i} = 0.5$, $\zeta_m = 0.001, \xi_{tc} = \xi_{mc} = 0.2$

Table 4 The comparison results among single-objective optimization algorithms for the Sphere function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	3.28×10^{-14}	2.56×10^{-18}	2.25×10^{-98}	1.58×10^{-119}
Standard deviation	4.58×10^{-14}	1.34×10^{-17}	6.54×10^{-98}	8.58×10^{-119}

algorithms. The population size and maximum iteration are set at 20 and 10,000, accordingly.

By contrasting the results of GA with traditional crossover, GA with multiple-crossover, standard PSO, and the hybrid of particle swarm optimization and the genetic algorithm (Tables 4, 5, 6, 7, 8, 9, 10, 11 and 12), it can be found that the hybrid algorithm has a superior performance with respect to other optimization algorithms. Moreover, the hybrid algorithm presents the best solutions in all test functions except Schwefel 2.22, in which the PSO algorithm has the best solution

Table 5 The comparison results among single-objective optimization algorithms for the Schwefel 2.22 function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	1.83×10^{-6}	8.16×10^{0}	3.15×10^{-26}	9.11×10^{-25}
Standard deviation	4.32×10^{-6}	$3.13 \times 10^{+1}$	1.73×10^{-25}	4.99×10^{-24}

Table 6 The comparison results among single-objective optimization algorithms for the Schwefel 1.2 function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	$7.74 \times 10^{+2}$	$3.14 \times 10^{+2}$	6.73×10^{-5}	2.14×10^{-11}
Standard deviation	$4.16 \times 10^{+2}$	$1.99 \times 10^{+2}$	9.40×10^{-5}	6.91×10^{-11}

Table 7 The comparison results among single-objective optimization algorithms for the Rosenbrock function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	$1.33 \times 10^{+2}$	$8.25 \times 10^{+1}$	$2.04 \times 10^{+1}$	5.34×10^{-1}
Standard deviation	$1.32 \times 10^{+2}$	$5.51 \times 10^{+1}$	$2.53 \times 10^{+1}$	1.38×10^{0}

Table 8 The comparison results among single-objective optimization algorithms for the Noise function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	5.10×10^{-2}	8.64×10^{-2}	1.06×10^{0}	3.38×10^{-3}
Standard deviation	1.88×10^{-2}	2.48×10^{-2}	3.12×10^{-1}	1.47×10^{-3}

Table 9 The comparison results among single-objective optimization algorithms for the Step function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	$2.10 \times 10^{+2}$	$3.40 \times 10^{+1}$	1.00×10^{-1}	0
Standard deviation	$3.27 \times 10^{+2}$	$1.32 \times 10^{+2}$	3.05×10^{-1}	0

Table 10 The comparison results among single-objective optimization algorithms for the Rastringin function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	$7.54 \times 10^{+1}$	$1.52 \times 10^{+2}$	$4.11 \times 10^{+1}$	2.15×10^{-3}
Standard deviation	$1.90 \times 10^{+1}$	$4.37 \times 10^{+1}$	9.19×10^{0}	1.18×10^{-2}

Table 11 The comparison results among single-objective optimization algorithms for the Griewank function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	7.16×10^{-1}	4.86×10^{-1}	2.15×10^{-2}	2.03×10^{-2}
Standard deviation	2.87×10^{0}	1.4478×10^{0}	2.31×10^{-2}	2.21×10^{-2}

Table 12 The comparison results among single-objective optimization algorithms for the Ackley function

	GA (traditional crossover)	GA (multiple-crossover)	PSO (standard)	The hybrid algorithm
Mean	$1.57 \times 10^{+1}$	$1.80 \times 10^{+1}$	2.81×10^{-1}	4.30×10^{-8}
Standard deviation	1.13×10^{0}	6.23×10^{-1}	5.92×10^{-1}	9.90×10^{-8}

but the result is very close to the results of the hybrid method. Figures 4, 5, 6, 7, 8, 9 and 10 illustrate the evolutionary traces of some test functions of Table 2. In these figures, the mean best values are gained for thirty runs. The maximum iteration, population size, and dimension are set at 1,000, 10 and 50, respectively. In these figures, the vertical axis is the value of the best function found after each iteration of the algorithms and the horizontal axis is the iteration. By comparing these figures, it is obtained that the combination of the traditional crossover, multiple-crossover and mutation operator can enhance the performance of particle swarm optimization.

5.2 Multi-objective Optimization

5.2.1 Definition of Multi-objective Optimization Problem

In most of real problems, there is more than one objective function required to be optimized. Furthermore, most of these functions are in conflict with each other. Hence, there is not just one solution for the problem and there are some optimal
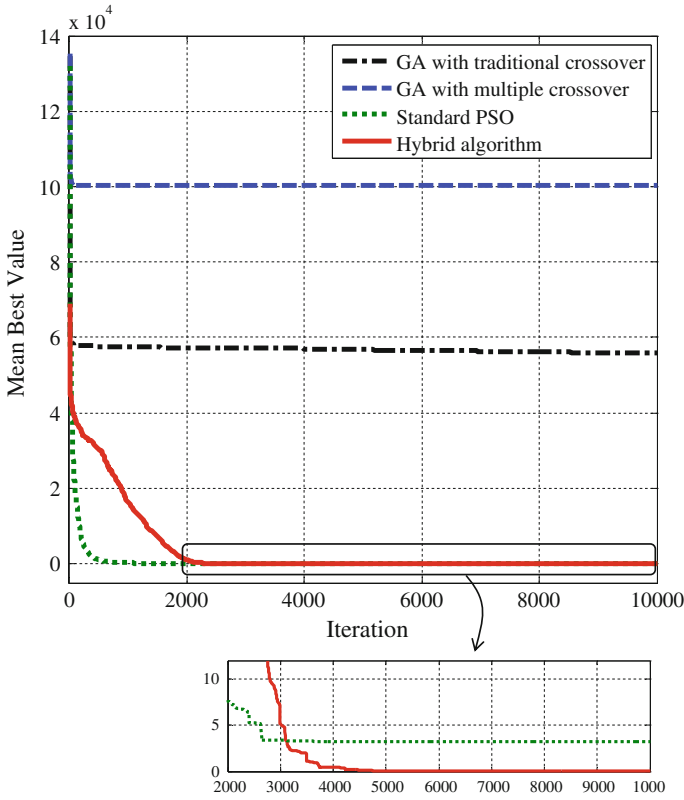
Fig. 4 The evolutionary trajectory of the single-objective optimization algorithm on the Sphere test function

solutions which are non-dominated with respect to each other and the designer can use each of them based upon the design criteria.

$$\text{Find } \vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*] \in R^n$$
$$\text{To minimize } \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})] \in R^m$$

By regarding p equality constraints $g_i(\vec{x}) = 0, \quad i = 1, 2, \ldots, p$ and q inequality constraints $h_j(\vec{x}) \leq 0, \quad j = 1, 2, \ldots, q$, where $\vec{x}$ represents the vector of decision variables and $\vec{f}(\vec{x})$ denotes the vector of objective functions.

As it is mentioned earlier, there is not one unique optimal solution for multi-objective problems. There exists a set of optimal solutions called Pareto-optimal solutions. The following definitions are needed to describe the concept of optimality (Deb et al. 2002).
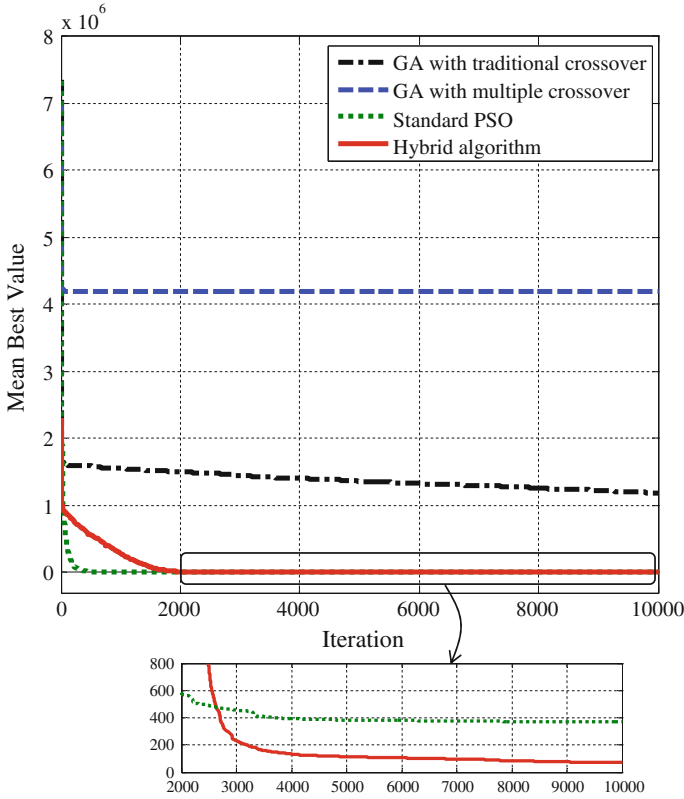
Fig. 5 The evolutionary trajectory of the single-objective optimization algorithm on the Rosenbrock test function

Definition 1 Pareto Dominance: It says that the vector $\vec{u} = [u_1, u_2, \ldots, u_n]$ dominates the vector $\vec{v} = [v_1, v_2, \ldots, v_n]$ and it illustrates $\vec{u} < \vec{v}$, if and only if:

$$\forall i \in \{1, 2, \ldots, n\} : u_i \leq v_i \wedge \exists\ j \in \{1, 2, \ldots n\} : u_j < v_j$$

Definition 2 Non-dominated: A vector of decision variables $\vec{x} \in X \subset R^n$ is non-dominated, if there is not another $\vec{x}' \in X$ which dominates $\vec{x}$. That is to say that $\forall \vec{x} \in X,\ \nexists \vec{x}' \in X, \vec{x} \neq \vec{x}' : \vec{f}(\vec{x}') < \vec{f}(\vec{x})$, where $\vec{f} = \{f_1, f_2, \ldots, f_m\}$ denotes the vector of objective functions.

Definition 3 Pareto-optimal: the vector of decision variables $\vec{x}^* \in X \subset R^n$, where X is the design feasible region, is Pareto-optimal if this vector is non-dominated in X.

Definition 4 Pareto-optimal set: In multi-objective problems, a Pareto-optimal set or in a more straightforward expression, a Pareto set denoted by P^* consists of all Pareto-optimal vectors, namely:
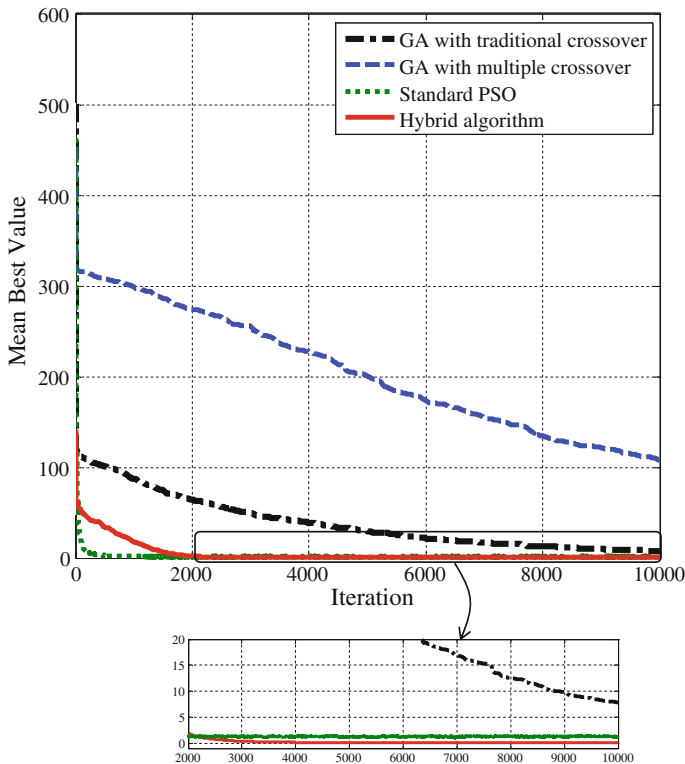
Fig. 6 The evolutionary trajectory of the single-objective optimization algorithm on the Noise test function

$$P^* = \{\vec{x} \in X \,|\, \vec{x} \text{ is Pareto-optimal}\}$$

Definition 5 Pareto-optimal front: The Pareto-optimal front or in a more straight-forward expression, Pareto front PF^* is defined as:

$$PF^* = \{\vec{f}(\vec{x}) \in R^m | \vec{x} \in P^*\}.$$

5.2.2 The Structure of the Hybrid Algorithm for Multi-objective Optimization

It is necessary to make modifications to the original scheme of PSO in finding the optimal solutions for multi-objective problems. In the single-objective algorithm of PSO, the best particle of the entire swarm ($\vec{x}_{gbest}$) is utilized as a leader. In the multi-objective algorithm, each particle has a set of different leaders that one of them is chosen as a leader. In this book paper, a leader selection method based upon density
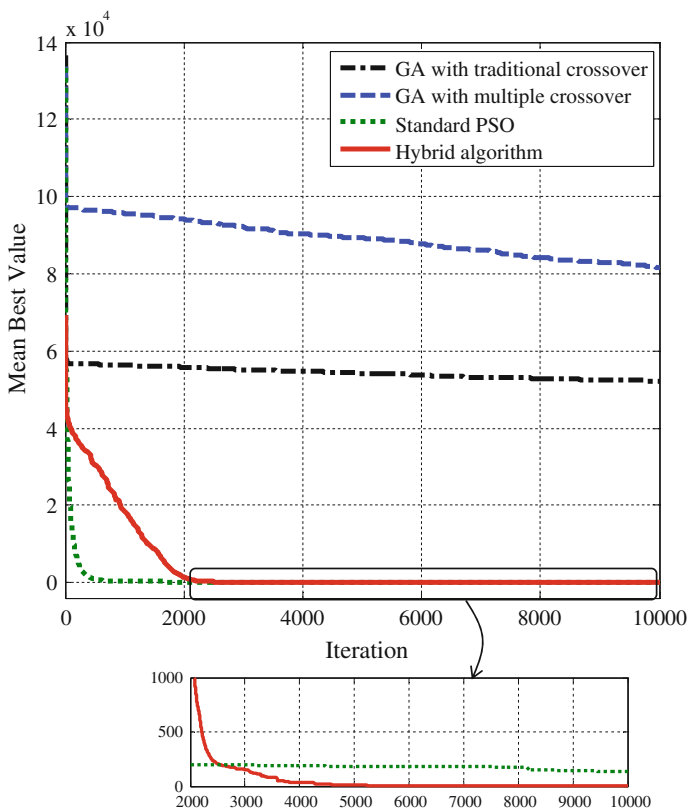
Fig. 7 The evolutionary trajectory of the single-objective optimization algorithm on the Step test function

measures is used (Mahmoodabadi et al. 2013). To this end, a neighborhood radius $R_{neighborhood}$ is defined for the whole non-dominated solutions. Two non-dominated solutions are regarded neighbors in case the Euclidean distance of them is less than $R_{neighborhood}$. Based upon this radius, the number of neighbors of each non-dominated solution is computed in the objective function domain and the particle having fewer neighbors is chosen as leaders. Furthermore, for particle i, the nearest member of the archive is devoted to $\overrightarrow{x}_{pbest_i}$. At this stage, a multi-objective optimization algorithm using the hybridization of genetic operators and PSO formula can be presented (Mahmoodabadi et al. 2013). In elaboration, the population is randomly generated. Once the fitness values of all members are computed, the first archive can be produced. The inertia weight, the learning factors and operator's probabilities are computed at each iteration. The genetic operators, that is, mutation operators, traditional crossover and multiple-crossover are utilized to change some chromosomes selected randomly. Each chromosome corresponds to a particle in it and the group of chromosome can be regarded as a swarm. On the other hand, the
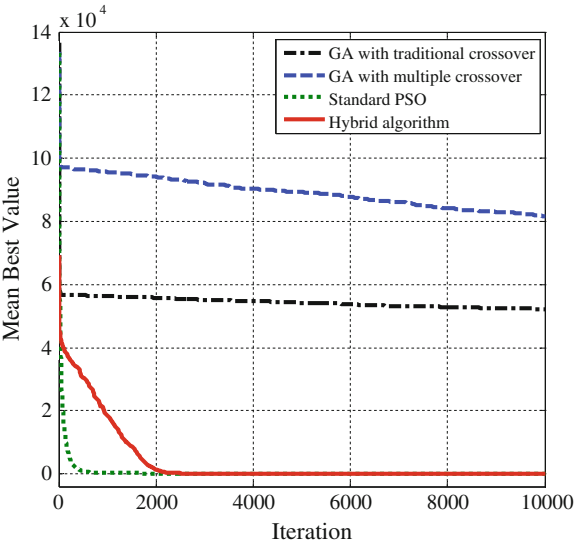
Fig. 8 The evolutionary trajectory of the single-objective optimization algorithm on the Rastringin test function
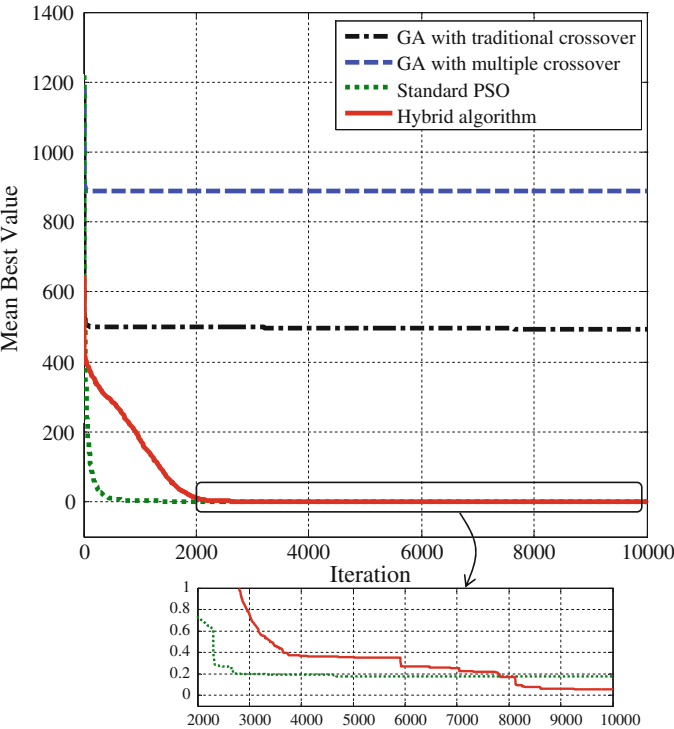

Fig. 9 The evolutionary trajectory of the single-objective optimization algorithm on the Griewank test function
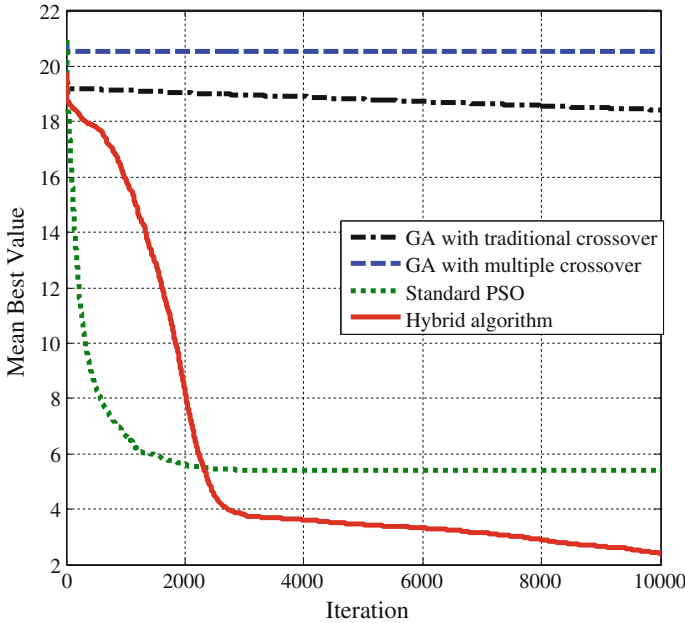
Fig. 10 The evolutionary trajectory of the single-objective optimization algorithm on the Ackley test function

chromosomes which are not chosen for genetic operations are enhanced via particle swarm optimization. Then, the archive is pruned and updated. This cycle is repeated until the user-defined stopping criterion is met. Figure 11 illustrates the flow chart of this algorithm.

The set of non-dominated solutions is saved in a different location named archive. If all of the non-dominated solutions are saved in the archive, the size of archive enhances rapidly. On the other hand, since the archive must be updated at each iteration, the size of archive will expand significantly. In this respect, a supplementary criterion is needed that resulted in saving a bounded number of non-dominated solutions. To this end, the dynamic elimination approach is utilized here to prune the archive (Mahmoodabadi et al. 2013). In this method, if the Euclidean distance between two particles is less than $R_{\text{elimination}}$ which is the elimination radius of each particle, then one of them will be eliminated. As an example, it is illustrated in Fig. 12. To gain the value of $R_{\text{elimination}}$, the following equation is utilized:

$$R_{elimination} = \begin{cases} \frac{t}{\alpha \times maximum\ iteration} & if\ \left(\frac{t}{\beta}\right) = fix\left(\frac{t}{\beta}\right) \\ 0 & else \end{cases} \tag{12}$$

Fig. 11 The flow chart of the hybrid algorithm for multi-objective optimization problems
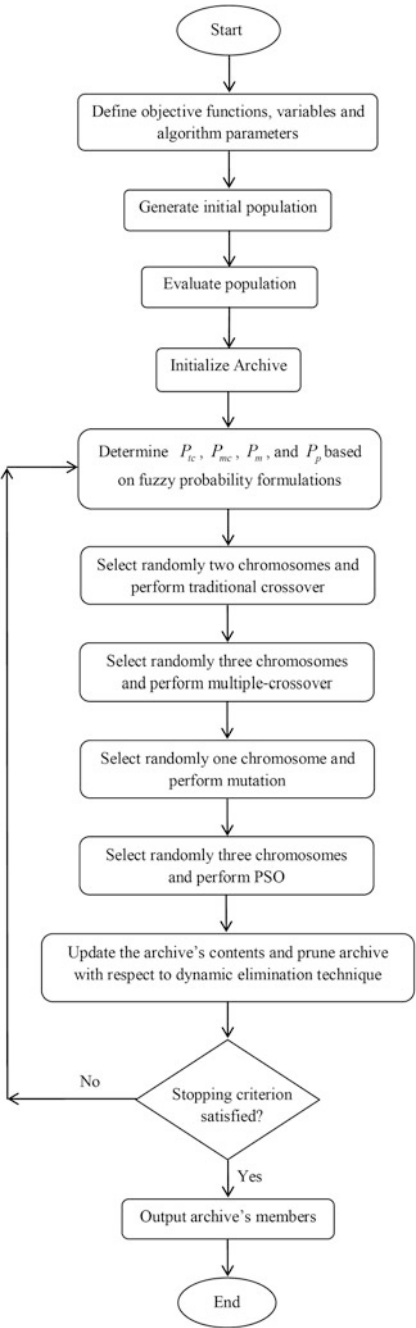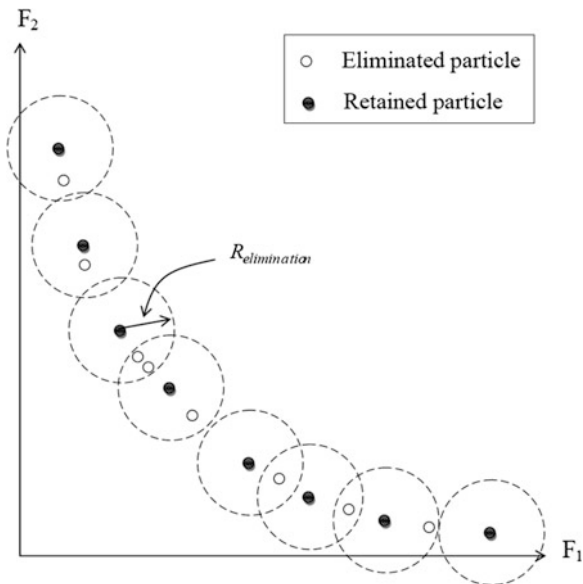
Fig. 12 The particles located in another particle's $R_{elimination}$ will be removed using the dynamic elimination technique

In which, t stands for the current iteration number and *maximum iteration* is the maximum number of allowable iterations. α and β are positive constants regarding as $\alpha = 100$ and $\beta = 10$.

5.2.3 Results for Multi-objective Optimization

Five multi-objective benchmark problems are regarded which have similar features such as the bounds of variables, the number of variables, the nature of Pareto-optimal front and the true Pareto optimal solutions. These problems which are unconstrained have two objective functions. The whole features of these algorithms are illustrated in Table 13. The contrast of the true Pareto optimal solutions and the results of the hybrid algorithm is illustrated in Figs. 13, 14, 15, 16 and 17. As it is obtained, the hybrid algorithm can present a proper result in terms of converging to the true Pareto optimal and gaining advantages of a diverse solution set.

In this comparison, the capability of the hybrid algorithm is contrasted to three prominent optimization algorithms, that is, NSGA-II (Deb et al. 2002), SPEA (Zitzler and Theile 1999) and PAES (Knowles and Corne 1999) with respect to the same test functions. Two crucial facts considered here are the diversity solution of the solutions with respect to the Pareto optimal front and the capability to gain the Pareto optimal set. Regarding these two facts, two performance metrics are utilized in evaluating each of the above-mentioned facts.

Table 13 Multi-objective test functions

Name	Variable bounds (dimension n)	Formula	Optimal solutions	Comments
SCH	$[-10^3, 10^3]n\ (n=1)$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$x \in [0, 2]$	Convex
FON	$[-4, 4]^n\ (n=3)$	$f_1(x) = 1-\exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(x) = 1-\exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$	$x_1 = x_2 = x_3$ $\in \left[-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]$	Nonconvex
ZDT1	$[0, 1]^n\ (n=30)$	$f_1(x) = x_1, f_2(x) = g(x)[1-\sqrt{x_1/g(x)}]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)\big/(n-1)$	$x_1 \in [0, 1], x_i = 0$ $i = 2,\ldots,\text{n}$	Convex
ZDT2	$[0, 1]^n\ (n=30)$	$f_1(x) = x_1, f_2(x) = g(x)[1-(x_1/g(x))^2]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)\big/(n-1)$	$x_1 \in [0, 1], x_i = 0$ $i = 2,\ldots,\text{n}$	Nonconvex
ZDT3	$[0, 1]^n\ (n=30)$	$f_1(x) = x_1, f_2(x) = g(x)[1-\sqrt{x_1/g(x)} - \frac{x_1}{g(x)}\sin(10\pi x_1)]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)\big/(n-1)$	$x_1 \in [0, 1], x_i = 0$ $i = 2,\ldots,\text{n}$	Convex, disconnected

Fig. 13 The non-dominated solutions of the hybrid method for the SCH test function
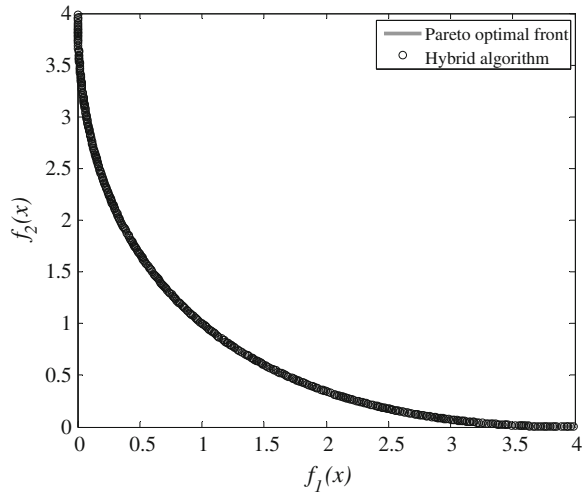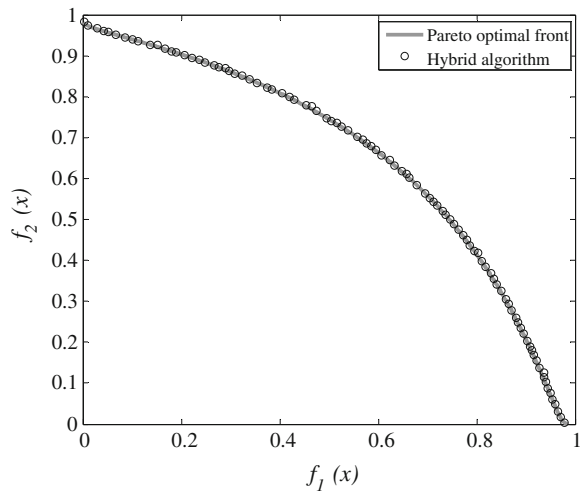

Fig. 14 The non-dominated solutions of the hybrid method for the FON test function

(1) A proper indication of the gap between the non-dominated solution members and the Pareto optimal front is gained by means of the metric of distance (Υ) (Deb et al. 2002) as follows:

$$\Upsilon = \sum_{i=1}^{n} d_i^2 \tag{13}$$

where n is the number of members in the set of non-dominated solutions and d_i is the least Euclidean distance between the member i in the set of non-dominated

Fig. 15 The non-dominated
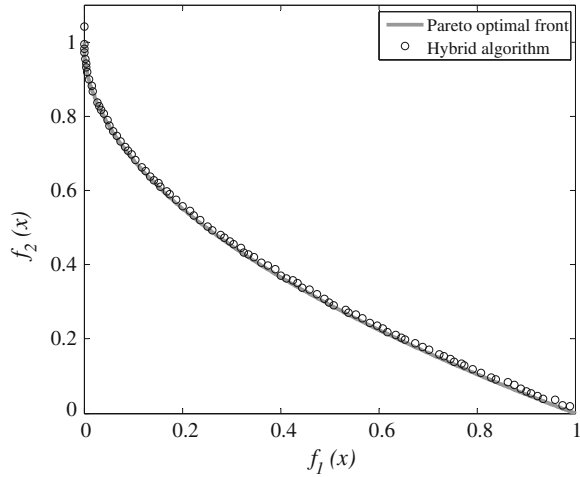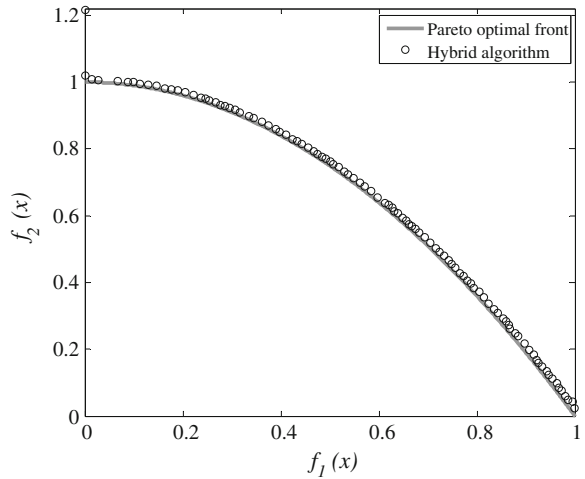solutions of the hybrid
method for the ZDT1 test
function

Fig. 16 The non-dominated
solutions of the hybrid
method for the ZDT2 test
function

solutions and Pareto optimal front. If all members in the set of non-dominated
solutions are in Pareto optimal front then $\Upsilon = 0$.

(2) The metric of diversity (Δ) (Deb et al. 2002) measures the extension of spread
achieved among non-dominated solutions, which is given as

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (n-1)\bar{d}} \tag{14}$$

In this formula, d_f and d_l denote the Euclidean distance between the boundary
solutions and the extreme solutions of the non-dominated set, n stands for the

Fig. 17 The non-dominated solutions of the hybrid method for the ZDT3 test function
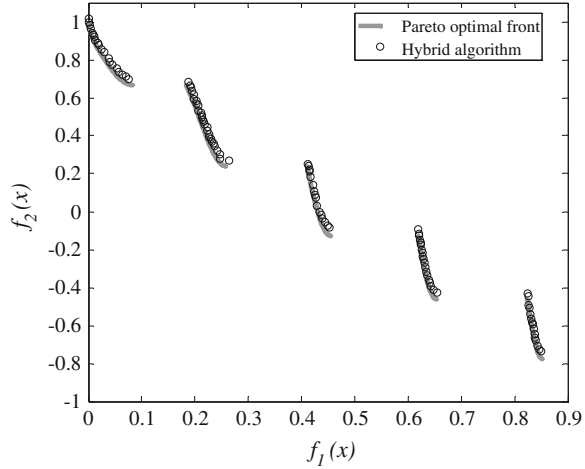
number of members in the set of non-dominated solutions, d_i is the Euclidean distance between consecutive solutions in the gained non-dominated set, and $\bar{d} = \frac{\sum_{i=1}^{n-1} d_i}{n-1}$.

For the most extent spread set of non-dominated solutions $\Delta = 0$

The performance of the hybrid algorithm comparing to NSGA-II (Deb et al. 2002), SPEA (Zitzler and Theile 1999), and PAES (Knowles and Corne 1999) algorithms is illustrated in Tables 14, 15, 16, 17 and 18.

Based on the results of Tables 14, 15, 16, 17 and 18, the hybrid algorithm has very proper Δ values for all test functions excluding ZDT2. While NSGA-II presents proper Δ results for all test functions except ZDT3, the approaches SPEA and PAES do not illustrate proper performance in the diversity metric. The hybrid algorithm presents acceptable results for the convergence metric in all test functions. On the other hand, NSGA-II ZDT3 function, SPEA for FON function, and PAES for FON and ZDT2 functions do not show proper performance.

The hybrid optimization algorithm is used to design the parameters of state feedback control for linear systems. In the following section, state space representation and the control input of state feedback control for linear systems will be presented.

Table 14 The results of the comparison of multi-objective optimization algorithms for the SCH test function

Metrics		NSGA-II	SPEA	PAES	The hybrid algorithm
Δ	Mean	4.77×10^{-1}	1.02×10^{0}	1.06×10^{0}	6.00×10^{-1}
	Standard deviation	3.47×10^{-3}	4.37×10^{-3}	2.86×10^{-3}	1.81×10^{-2}
Υ	Mean	3.39×10^{-3}	3.40×10^{-3}	1.31×10^{-3}	3.22×10^{-3}
	Standard deviation	0	0	3.00×10^{-6}	1.35×10^{-4}

Table 15 The results of the comparison of multi-objective optimization algorithms for the FON test function

Metrics		NSGA-II	SPEA	PAES	The hybrid algorithm
Δ	Mean	3.78×10^{-1}	7.92×10^{-1}	1.16×10^{0}	5.90×10^{-1}
	Standard deviation	6.39×10^{-4}	5.54×10^{-3}	8.94×10^{-3}	3.60×10^{-2}
Υ	Mean	1.93×10^{-3}	1.25×10^{-1}	1.51×10^{-1}	1.56×10^{-3}
	Standard deviation	0	3.80×10^{-5}	9.05×10^{-4}	1.71×10^{-4}

Table 16 The results of the comparison of multi-objective optimization algorithms for the ZDT1 test function

Metrics		NSGA-II	SPEA	PAES	The hybrid algorithm
Δ	Mean	3.90×10^{-1}	7.84×10^{-1}	1.22×10^{0}	6.55×10^{-1}
	Standard deviation	1.87×10^{-3}	4.44×10^{-3}	4.83×10^{-3}	4.91×10^{-2}
Υ	Mean	3.34×10^{-2}	1.79×10^{-3}	8.20×10^{-2}	8.16×10^{-3}
	Standard deviation	4.75×10^{-3}	1.00×10^{-6}	8.67×10^{-3}	2.73×10^{-3}

Table 17 The results of the comparison of multi-objective optimization algorithms for the ZDT2 test function

Metrics		NSGA-II	SPEA	PAES	The hybrid algorithm
Δ	Mean	4.30×10^{-1}	7.55×10^{-1}	1.16×10^{0}	9.57×10^{-1}
	Standard deviation	4.72×10^{-3}	4.52×10^{-3}	7.68×10^{-3}	3.20×10^{-2}
Υ	Mean	7.23×10^{-2}	1.33×10^{-3}	1.26×10^{-1}	3.04×10^{-2}
	Standard deviation	3.16×10^{-2}	0	3.68×10^{-2}	1.84×10^{-2}

Table 18 The results of the comparison of multi-objective optimization algorithms for the ZDT3 test function

Metrics		NSGA-II	SPEA	PAES	The hybrid algorithm
Δ	Mean	7.38×10^{-1}	6.72×10^{-1}	7.89×10^{-1}	6.28×10^{-1}
	Standard deviation	1.97×10^{-2}	3.58×10^{-3}	1.65×10^{-3}	5.30×10^{-2}
Υ	Mean	1.14×10^{-1}	4.75×10^{-2}	2.38×10^{-2}	8.88×10^{-3}
	Standard deviation	7.94×10^{-3}	4.70×10^{-5}	1.00×10^{-5}	6.97×10^{-3}

6 State Feedback Control for Linear Systems

The vector state equation can be utilized for a continuous time system, as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{B}_v\boldsymbol{v}(t) \tag{15}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ stands for the state vector, $\dot{\boldsymbol{x}}(t)$ denotes the time derivative of state vector and $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the input vector. The disturbance $\boldsymbol{v}(t)$ is assumed to be a

deterministic nature. Furthermore, $A \in \mathbb{R}^{n \times n}$ is the system or dynamic matrix, $B \in \mathbb{R}^{n \times m}$ is the input matrix and B_v is the disturbance matrix. Measurements are made on this system which can be either the states themselves or linear combinations of them:

$$y(t) = Cx(t) + w(t) \tag{16}$$

where $y(t) \in \mathbb{R}^r$ is the output vector and $C \in \mathbb{R}^{r \times n}$ is the output matrix. The vector $w(t)$ stands for the measurement disturbance.

In order to establish linear state feedback around the above system, a linear feedback law can be applied as follows:

$$u(t) = -Kx(t) + r(t) \tag{17}$$

In this formula, $K \in \mathbb{R}^{m \times n}$ stands for a feedback matrix (or a gain matrix). $r(t)$ denotes the reference input vector of the system having dimensions the same as the input vector $u(t)$. The resulting feedback system is a full state feedback system due to measuring all of the states. To design the state feedback controller with an optimal control input and minimum error, the hybrid optimization algorithm is applied and the optimal Pareto front of the controller is shown in the following section.

7 Pareto Optimal State Feedback Control of a Parallel-Double-Inverted Pendulum

The model of a parallel-double-inverted pendulum system is presented in this section. The work deals with the stabilization control of a system which is a complicated nonlinear and unstable high-order system. Figure 18 illustrates the mechanical structure of the inverted pendulum. According to the figure, the cart is moving on a track with two pendulums hinged and balanced upward by means of a DC motor. In addition, the cart has to track a (varying) reference position. This system includes two pendulums and one cart. The pendulums are attached to the cart. While the cart is moving, the system has to be controlled in such a way that pendulums are placed in desired angels. The dynamic equations of the system are as follows:

$$I_1 \ddot{\varphi} + C_1 \dot{\varphi} - a_3 \sin \varphi + a_1 \ddot{x} \cos \varphi = 0 \tag{18}$$

$$I_2 \ddot{\alpha} + C_2 \dot{\alpha} - a_4 \sin \alpha + a_2 \ddot{x} \cos \alpha = 0 \tag{19}$$

$$M\ddot{x} + f_r \dot{x} + a_1 \left(\ddot{\varphi} \cos \varphi - \dot{\varphi}^2 \sin \varphi \right) + a_2 \left(\ddot{\alpha} \cos \alpha - \dot{\alpha}^2 \sin \alpha \right) = u \tag{20}$$

Fig. 18 The system of a parallel-double-inverted pendulum
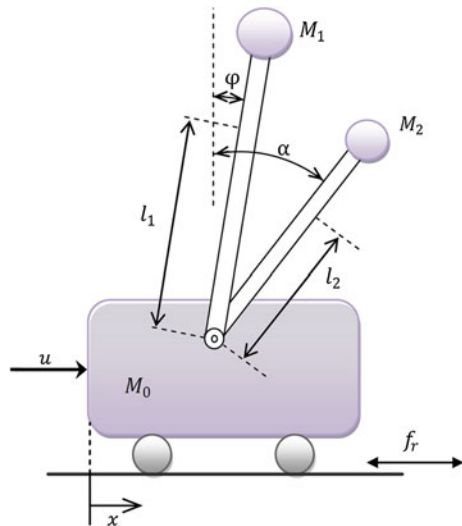
where

$$I_1 = I_1' + M_1 l_1^2, \, I_2 = I_2' + M_2 l_2^2, \tag{21}$$

$$M = M_0 + M_1 + M_2 \tag{22}$$

$$a_1 = M_1 l_1, \, a_2 = M_2 l_2, \, a_3 = M_1 l_1 g, \, a_4 = M_2 l_2 g \tag{23}$$

where x is the position of the cart, $\dot{x}$ is the velocity of the cart, φ stands for the angular velocity of the first pendulum with respect to the vertical line, $\dot{\varphi}$ is the angular velocity of the first pendulum, α is the angular position of the second pendulum, $\dot{\alpha}$ represents the angular velocity of the second pendulum, M_1 is the mass of the first pendulum, M_2 is the mass of second pendulum, M_0 is the mass of the cart, l_1 denotes the length of the first pendulum with respect to its center, l_2 stands for the length of the second pendulum with respect to its center, f_r is the friction coefficient of the cart with ground, I_1' is the inertia moment of the first pendulum with respect to its center, I_2' represents the inertia moment of the second pendulum with respect to its center, C_1 is the angular frictional coefficient of the first pendulum, C_2 stands for the angular frictional coefficient of the second pendulum, and u is the control effort.

To obtain the state space representations of the dynamic equations, the state space variables are defined as $x = [x_1, x_2, x_3, x_4, x_5, x_6]^T$. This vector includes the position of the cart, the velocity of the cart, the angular position and velocity of the first pendulum, the angular position and velocity of the second pendulum. After linearization around the equilibrium point $x_a = [x_1, 0, 0, 0, 0, 0]^T$, the state space representation is obtained according to Eq. (25).

$$
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & \frac{f_r I_1 I_2}{P_2} & -\frac{a_1 a_3 I_2}{P_2} & -\frac{a_1 C_1 I_2}{P_2} & -\frac{a_2 a_4 I_1}{P_2} & -\frac{a_2 C_2 I_1}{P_2} \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & -\frac{a_1 f_r I_2}{P_1} & \frac{a_3(a_2^2 - m I_2)}{P_1} & -\frac{c_1(a_2^2 - m I_2)}{P_1} & -\frac{a_1 a_2 a_4}{P_1} & -\frac{a_1 a_2 a_4}{P_1} \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & -\frac{a_2 f_r I_1}{P_2} & \frac{-a_1^2 a_2 a_3 I_2}{P_3} & \frac{a_1 a_2 C_1}{P_2} & \frac{a_1^3 a_4 I_2 - a_1 a_4 m I_1 I_2}{P_3} & \frac{-a_1^3 C_2 I_2 + a_1 C_2 m I_1 I_2}{P_3}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}
$$

$$
+ \begin{bmatrix} 0 \\ -\frac{I_1 I_2}{P_2} \\ 0 \\ \frac{a_1 I_2}{P_1} \\ 0 \\ \frac{a_2 I_1}{P_2} \end{bmatrix} u
\tag{25}
$$

where

$$
P_1 = a_1^2 I_2 - I_1(-a_2^2 + m I_2) \tag{26}
$$

$$
P_2 = a_2^2 I_1 + a_1^2 I_2 - m I_1 I_2 \tag{27}
$$

$$
P_3 = a_1 I_2(a_2^2 I_1 + a_1^2 I_2 - m I_1 I_2) \tag{28}
$$

The block diagram of the linear state feedback controller to control the parallel-double-inverted pendulum is illustrated in Fig. 19. The control effort of the state feedback controller is obtained as follows

$$
\begin{aligned}
u = {}& K_1(x_1 - x_{1,d}) + K_2(x_2 - x_{2,d}) + K_3(x_3 - x_{3,d}) \\
& + K_4(x_4 - x_{4,d}) + K_5(x_5 - x_{5,d}) + K_6(x_6 - x_{6,d})
\end{aligned}
\tag{29}
$$

where $x_d = [x_{1,d}, x_{2,d}, x_{3,d}, x_{4,d}, x_{5,d}, x_{6,d}]^T$ is the vector of the desired states and $K = [K_1, K_2, K_3, K_4, K_5, K_6]$ is the vector of design variables obtained via the optimization algorithm. The boundaries of the system are:

The boundary of the control effort is $|u| \leq 20 \ [N]$

The boundary of the length of x_1, x_3 and x_5 are $|x_1| \leq 0.5 \ [m]$, $|x_3| \leq 0.174 \ [rad]$, $|x_5| \leq 0.174 \ [rad]$.

The initial state vector, final state vector, and the boundaries of design variables are as follows. Furthermore, the values of the parameters of the system of a parallel-double-inverted pendulum are presented in Table 19.
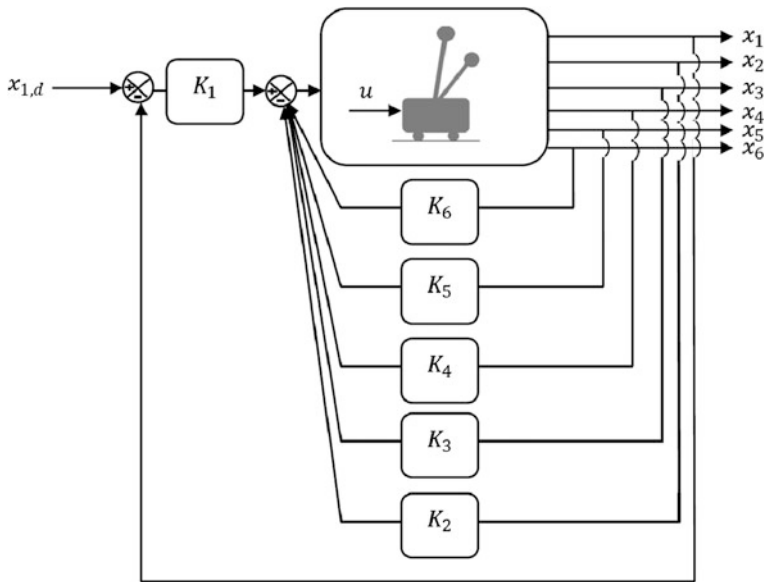
Fig. 19 The block diagram of the linear state feedback controller for a parallel-double-inverted pendulum for $x_d = [x_{1,d}, 0, 0, 0, 0, 0]^T$

Table 19 The values of the parameters of the system of a parallel-double-inverted pendulum

m_0	4.2774 kg
m_1	0.3211 kg
m_2	0.2355 kg
l_1	0.3533 m
l_2	0.0963 m
Θ_1	0.072 kg m^2
Θ_2	0.0044 kg m^2
F_r	10 Kg/s
C_1	0.023 Kg m^2/s
C_2	0.00145 Kg m^2/s

$$x_0 = [0, 0, 0, 0, 0, 0]^T \tag{30}$$

$$x_d = [0.2, 0, 0, 0, 0, 0]^T \tag{31}$$

$$50 \le K_1 \le 150 \tag{32}$$

$$150 \le K_2 \le 250 \tag{33}$$

$$14,900 \leq K_3 \leq 15,700 \tag{34}$$

$$3,000 \leq K_4 \leq 4,000 \tag{35}$$

$$-14,000 \leq K_5 \leq -12,000 \tag{36}$$

$$-3,000 \leq K_6 \leq -1,500 \tag{37}$$

In this problem, the objective functions of the multi-objective optimization algorithm are

F_1 = the sum of settling time and overshoot of the cart;
F_2 = the sum of settling time and overshoot of the first pendulum + the sum of settling time and overshoot of the second pendulum;

These objective functions have to be minimized simultaneously. The Pareto front of the control of the system of the parallel-double-inverted pendulum obtained via multi-objective hybrid of particle swarm optimization and the genetic algorithm is shown in Fig. 20. In Fig. 20, points A and C stand for the best sum of settling time and overshoot of the cart and the sum of settling time and overshoot of the first and second pendulums, respectively. It is clear from this figure that all the optimum design points in the Pareto front are non-dominated and could be chosen by a designer as optimum linear state feedback controllers. It is also clear that choosing a better value for any objective function in a Pareto front would cause a worse value for another objective. The corresponding decision variables (vector of linear state
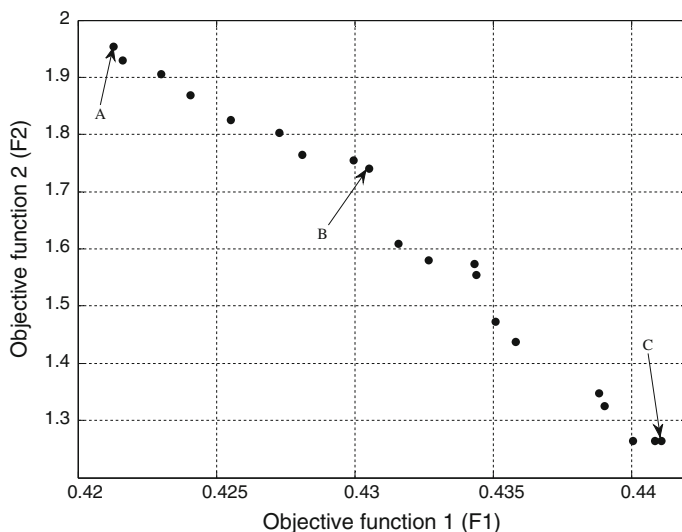

Fig. 20 Pareto front of multi-objective hybrid of particle swarm optimization and the genetic algorithm for the control of the system of the parallel-double-inverted pendulum

Table 20 The values of the parameters of the system of the parallel-double-inverted pendulum

	K_1	K_2	K_3	K_4	K_5	K_6	F_1	F_2
Point A	99.8110	224.5012	14,984.2210	3,967.5551	−12,480.5041	−2,502.5520	0.4213	1.9578
Point B	100.3566	225.5012	14,975.5521	3,968.2250	−12,404.6652	−2,493.3324	0.4305	1.7422
Point C	99.0315	224.5646	14,999.1835	3,966.4309	−12,391.4606	−2,491.0947	0.4411	1.2622

feedback controllers) of the Pareto front shown in Fig. 20 are the best possible design points. Moreover, if any other set of decision variables is selected, the corresponding values of the pair of those objective functions will locate a point inferior to that Pareto front. Indeed, such inferior area in the space of two objectives is top/right side of Fig. 20. Thus, the Pareto optimum design method causes to find important optimal design facts between these two objective functions. From Fig. 20, point B is the point which demonstrates such important optimal design facts. This point could be the trade-off optimum choice when considering minimum values of both sum of settling time and overshoot of the cart and sum of settling time and overshoot of the first and second pendulums. The values of the design variables obtained for three design points are illustrated in Table 20. The control effort, the angle of the first pendulum, the angle of the second pendulum, and the position of the cart are illustrated in Figs. 21, 22, 23 and 24. By regarding these figures, it can be concluded that the point A has the best time response (overshoot plus settling time) of the cart and the worst time responses (overshoot plus settling time) of the pendulums while point C has the best time responses of pendulums and the worst time response of the cart.
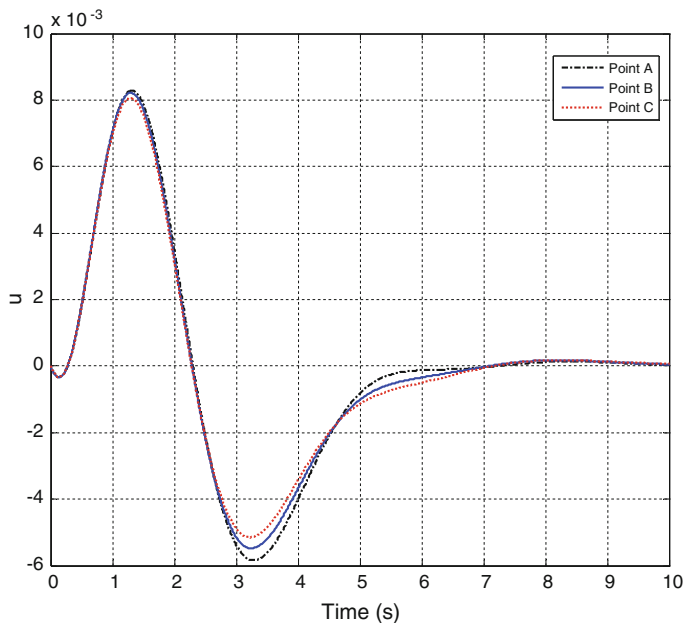

Fig. 21 The control effort of the system of the parallel-double-inverted pendulum for design points of the Pareto front of multi-objective hybrid of particle swarm optimization and the genetic algorithm
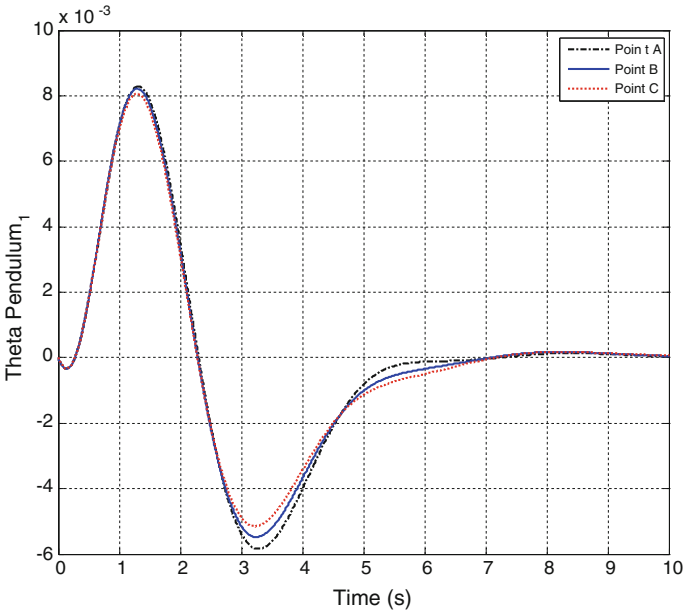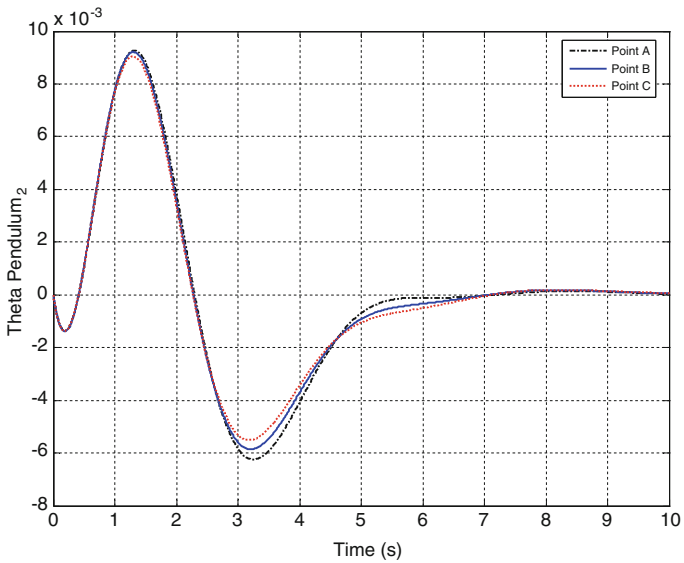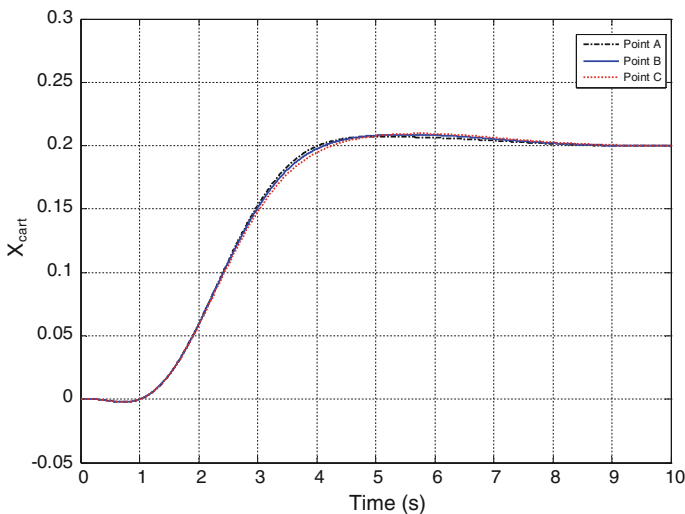
Fig. 22 The angle of the first pendulum of the system of the parallel-double-inverted pendulum for design points of the Pareto front of multi-objective hybrid of particle swarm optimization and the genetic algorithm

Fig. 23 The angle of the second pendulum of the system of the parallel-double-inverted pendulum for design points of the Pareto front of multi-objective hybrid of particle swarm optimization and the genetic algorithm

Fig. 24 The position of the cart of the system of the parallel-double-inverted pendulum for design points of the Pareto front of multi-objective hybrid of particle swarm optimization and the genetic algorithm

8 Conclusions

In this work, a hybrid algorithm using GA operators and PSO formula was presented via using effectual operators, for example, traditional and multiple-crossover, mutation and PSO formula. The traditional and multiple-crossover probabilities were based upon fuzzy relations. Five prominent multi-objective test functions and nine single-objective test functions were used to evaluate the capabilities of the hybrid algorithm. Contrasting the results of the hybrid algorithm with other algorithms demonstrates the superiority of the hybrid algorithm with regard to single and multi-objective optimization problems. Moreover, the hybrid optimization algorithm was used to obtain the Pareto front of non-commensurable objective functions in designing parameters of linear state feedback control for a parallel-double-inverted pendulum system. The conflicting objective functions of this problem were the sum of settling time and overshoot of the cart and the sum of settling time and overshoot of the first and second pendulums. The hybrid algorithm could design the parameters of the controller appropriately in order to minimize both objective functions simultaneously.

References

Abdel-Kader, R. F. (2010). Generically improved PSO algorithm for efficient data clustering. In *The 2010 Second International Conference on Machine Learning and Computing (ICMLC)*, February 9–11, 2010, Bangalore (pp. 71–75). doi:10.1109/ICMLC.2010.19.

Ahmadi, M. H., Aghaj, S. S. G., & Nazeri, A. (2013). Prediction of power in solar stirling heat engine by using neural network based on hybrid genetic algorithm and particle swarm optimization. *Neural Computing and Applications, 22*(6), 1141–1150.

Altun, A. A. (2013). A combination of genetic algorithm, particle swarm optimization and neural network for palmprint recognition. *Neural Computing and Applications, 22*(1), 27–33.

Aziz, A. S. A., Azar, A. T., Salama, M. A., Hassanien, A. E., & Hanafy, S. E. O. (2013). In *The 2013 Federated Conference on Computer Science and Information Systems (FedCSIS)*, September 8-11, 2013, Kraków (pp. 769–774).

Bhuvaneswari, R., Sakthivel, V. P., Subramanian, S., & Bellarmine, G. T. (2009). Hybrid approach using GA and PSO for alternator design. In *The 2009. SOUTHEASTCON '09. IEEE Southeastcon*, March 5–8, 2009, Atlanta (pp. 169–174). doi:10.1109/SECON.2009.5174070.

Blake, A. (1989). Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(1), 2–12.

Castillo-Villar, K. K., Smith, N. R., & Herbert-Acero, J. F. (2014). Design and optimization of capacitated supply chain networks including quality measures. *Mathematical Problems in Engineering*, *2014*, 17, Article ID 218913. doi:10.1155/2014/218913.

Castillo-Villar, K. K., Smith, N. R., & Simonton, J. L. (2012). The impact of the cost of quality on serial supply-chain network design. *International Journal of Production Research, 50*(19), 5544–5566.

Chang, W. D. (2007). A multi-crossover genetic approach to multivariable PID controllers tuning. *Expert Systems with Applications, 33*(3), 620–626.

Chen, J. L., & Chang, W. D. (2009). Feedback linearization control of a two link robot using a multi-crossover genetic algorithm. *Expert Systems with Applications, 36*(2), 4154–4159.

Chen, C.-H., & Liao, Y.- Y. (2014). Tribal particle swarm optimization for neurofuzzy inference systems and its prediction applications. *Communications in Nonlinear Science and Numerical Simulation, 19*(4), 914–929.

Chen, Z., Meng, W., Zhang, J., & Zeng, J. (2009). Scheme of sliding mode control based on modified particle swarm optimization. *Systems Engineering-Theory & Practice, 29*(5), 137–141.

Chutarat, A. (2001). Experience of light: The use of an inverse method and a genetic algorithm in day lighting design. *Ph.D. Thesis*, Department of Architecture, MIT, Massachusetts, USA.

Cordella, F., Zollo, L., Guglielmelli, E., & Siciliano, B. (2012). A bio-inspired grasp optimization algorithm for an anthropomorphic robotic hand. *International Journal on Interactive Design and Manufacturing (IJIDeM), 6*(2), 113–122.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Deb, K., & Padhye, N. (2013). Enhancing performance of particle swarm optimization through an algorithmic link with genetic algorithms. *Computational Optimization and Applications, 57*(3), 761–794.

Dhadwal, M. K., Jung, S. N., & Kim, C. J. (2014). Advanced particle swarm assisted genetic algorithm for constrained optimization problems. *Computational Optimization and Applications, 58*(3), 781–806.

Eberhart, R., Simpson, P., & Dobbins, R. (1996). *Computational intelligence PC tools*. Massachusetts: Academic Press Professional Inc.

Eberhart, R. C., Kennedy, J. (1995). A new optimizer using particle swarm theory. In The Proceedings of the Sixth International Symposium on Micro Machine and Human Science, October 4–6, 1995, Nagoya (pp. 39–43). doi:10.1109/MHS.1995.494215.

Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2014). A new genetic algorithm for solving optimization problems. *Engineering Applications of Artificial Intelligence, 27*, 57–69.

Elshazly, H. I., Azar, A. T., Hassanien, A. E., & Elkorany, A. M. (2013). Hybrid system based on rough sets and genetic algorithms for medical data classifications. *International Journal of Fuzzy System Applications (IJFSA), 3*(4), 31–46.

Engelbrecht, A. P. (2002). *Computational intelligence: An introduction*. New York: Wiley.

Engelbrecht, A. P. (2005). *Fundamentals of computational swarm intelligence*. New York: Wiley.

Fleming, P. J., & Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering: A survey. *Control Engineering Practice, 10*(11), 1223–1241.

Fonseca, C. M., & Fleming, P. J. (1994). Multiobjective optimal controller design with genetic algorithms. In The International Conference on Control, March 21–24, 1994, Coventry (pp. 745–749). doi:10.1049/cp:19940225.

Gaing, Z. L. (2004). A particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE Transactions on Energy Conversion, 19*(2), 384–391.

Gero, J., & Radford, A. (1978). A dynamic programming approach to the optimum lighting problem. *Engineering Optimization, 3*, 71–82.

Gosh, A., Das, S., Chowdhury, A., & Giri, R. (2011). An ecologically inspired direct search method for solving optimal control problems with Bezier parameterization. *Engineering Applications of Artificial Intelligence, 24*(7), 1195–1203.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor, Michigan: University of Michigan Press.

Jamili, A., Shafia, M. A., & Tavakkoli-Moghaddam, R. (2011). A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 54*(1–4), 309–322.

Jeong, S., Hasegawa, S., Shimoyama, K., & Obayashi, A. (2009). Development and investigation of efficient GA/PSO-hybrid algorithm applicable to real-world design optimization. *IEEE Computational Intelligence Magazine, 4*(3), 36–44.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In The IEEE International Conference on Neural Networks, November/December, 1995, Perth (pp. 1942–1948). doi:10.1109/ICNN.1995.488968.

Ker-Wei, Y., & Shang-Chang, H. (2006). An application of AC servo motor by using particle swarm optimization based sliding mode controller. In The IEEE International Conference on Systems, Man and Cybernetics, October 8-11, 2006, Taipei (pp. 4146–4150). doi:10.1109/ICSMC.2006.384784.

Knowles, J., & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In The Proceedings of the 1999 Congress on Evolutionary Computation, July, 1999, Washington (pp. 98–105). doi:10.1109/CEC.1999.781913.

Li, Z., Yang, K., Bogdan, S., & Xu, B. (2013). On motion optimization of robotic manipulators with strong nonlinear dynamic coupling using support area level set algorithm. *International Journal of Control, Automation and Systems, 11*(6), 1266–1275.

Mahmoodabadi, M. J., Safaie, A. A., Bagheri, A., & Nariman-zadeh, N. (2013). A novel combination of particle swarm optimization and genetic algorithm for pareto optimal design of a five-degree of freedom vehicle vibration model. *Applied Soft Computing, 13*(5), 2577–2591.

Mahmoodabadi, M. J., Bagheri, A., Arabani Mostaghim, S., & Bisheban, M. (2011). Simulation of stability using Java application for Pareto design of controllers based on a new multi-objective particle swarm optimization. *Mathematical and Computer Modelling, 54*(5–6), 1584–1607.

Mahmoodabadi, M. J., Momennejad, S., & Bagheri, A. (2014a). Online optimal decoupled sliding mode control based on moving least squares and particle swarm optimization. *Information Sciences, 268*, 342–356.

Mahmoodabadi, M. J., Taherkhorsandi, M., & Bagheri, A. (2014b). Optimal robust sliding mode tracking control of a biped robot based on ingenious multi-objective PSO. *Neurocomputing, 124*, 194–209.

Mahmoodabadi, M. J., Taherkhorsandi, M., & Bagheri, A. (2014c). Pareto design of state feedback tracking control of a biped robot via multiobjective PSO in comparison with sigma method and genetic algorithms: Modified NSGAII and MATLAB's toolbox. *The Scientific World Journal, 2014*, 8, Article ID 303101.

Mavaddaty, S., & Ebrahimzadeh, A. (2011). Blind signals separation with genetic algorithm and particle swarm optimization based on mutual information. *Radioelectronics and Communications Systems, 54*(6), 315–324.

McGookin, E. W., Murray-Smith, D. J., Li, Y., & Fossen, T. I. (2000). The optimization of a tanker autopilot control system using genetic algorithms. *Transactions of the Institute of Measurement and Control, 22*(2), 141–178.

Mizumoto, M. (1996). Product-sum-gravity method = fuzzy singleton-type reasoning method = simplified fuzzy reasoning method. In The Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, September 8–11, 1996, New Orleans (pp. 2098–2102). doi:10.1109/FUZZY.1996.552786.

Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2012). A competitive clustering particle swarm optimizer for dynamic optimization problems. *Swarm Intelligence, 6*(3), 177–206.

Premalatha, K., & Natarajan, A. M. (2009). Discrete PSO with GA operators for document clustering. *International Journal of Recent Trends in Engineering, 1*(1), 20–24.

Puri, P., & Ghosh, S. (2013). A hybrid optimization approach for PI controller tuning based on gain and phase margin specifications. *Swarm and Evolutionary Computation, 8*, 69–78.

Qiao, W., Venayagamoorthy, G. K., & Harley, R. G. (2006). Design of optimal PI controllers for doubly fed induction generators driven by wind turbines using particle swarm optimization. In The International Joint Conference on Neural Networks, Vancouver (pp. 1982–1987). doi:10.1109/IJCNN.2006.246944.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficient computation. *IEEE Transactions on Evolutionary Computation, 8*(3), 240–255.

Ravindran, A., Ragsdell, K. M., & Reklaitis, G. V. (2006). *Engineering optimization: Method and applications* (2nd ed.). New Jersey: Wiley.

Sakamoto, Y., Nagaiwa, A., Kobayasi, S., & Shinozaki, T. (1999). An optimization method of district heating and cooling plant operation based on genetic algorithm. *ASHRAE Transaction, 105*, 104–115.

Samarghandi, H., & ElMekkawy, T. Y. (2012). A genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and makespan criterion. *The International Journal of Advanced Manufacturing Technology, 61*(9–12), 1101–1114.

Sanchez, G., Villasana, M., & Strefezza, M. (2007). Multi-objective pole placement with evolutionary algorithms. *Lecture Notes in Computer Science, 4403*, 417–427. doi:10.1007/978-3-540-70928-2_33.

Arumugam, M. S., Rao, M. V. C., & Palaniappan, R. (2005). New hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems. *Applied Soft Computing, 6*(1), 38–52.

Song, K. S., Kang, S. O., Jun, S. O., Park, H. I., Kee, J. D., Kim, K. H., et al. (2012). Aerodynamic design optimization of rear body shapes of a sedan for drag reduction. *International Journal of Automotive Technology, 13*(6), 905–914.

Talatahari, S., & Kaveh, A. (2007). A discrete particle swarm ant colony optimization for design of steel frames. *Asian Journal of Civil Engineering (Building and Housing), 9*(6), 563–575.

Tang, Y., Wang, Z., & Fang, J. (2011). Controller design for synchronization of an array of delayed neural networks using a controllable probabilistic PSO. *Information Sciences, 181*(20), 4715–4732.

Thakur, M. (2014). A new genetic algorithm for global optimization of multimodal continuous functions. *Journal of Computational Science, 5*(2), 298–311.

Thangaraj, R., Pant, M., Abraham, A., & Bouvry, P. (2011). Particle swarm optimization: Hybridization perspectives and experimental illustrations. *Applied Mathematics and Computation, 217*(12), 5208–5226.

Valdez, F., Melin, P., & Castillo, O. (2011). An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithm. *Applied Soft Computing, 11*(2), 2625–2632.

Valdez, F., Melin, P., & Castillo, O. (2009). Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making. In The IEEE International Conference on Fuzzy Systems, August 20–24, 2009, Jeju Island (pp. 2114–2119). doi:10.1109/FUZZY.2009.5277165.

Wai, R. J., Chuang, K. L., & Lee, J. D. (2007). Total sliding-model-based particle swarm optimization controller design for linear induction motor. In The IEEE Congress on Evolutionary Computation, September 25–28, 2007, Singapore (pp. 4729–4734). doi:10.1109/CEC.2007.4425092.

Wang, H.-B., & Liu, M. (2012). Design of robotic visual servo control based on neural network and genetic algorithm. *International Journal of Automation and Computing, 9*(1), 24–29.

Wang, J. S., Zhang, Y., & Wang, W. (2006). Optimal design of PI/PD controller for non-minimum phase system. *Transactions of the Institute of Measurement and Control, 28*(1), 27–35.

Wang, K., & Zheng, Y. J. (2012). A new particle swarm optimization algorithm for fuzzy optimization of armored vehicle scheme design. *Applied Intelligence, 37*(4), 520–526.

Wang, L., Wang, T.-G., & Luo, Y. (2011). Improved non-dominated sorting genetic algorithm (NSGA)-II in multi-objective optimization studies of wind turbine blades. *Applied Mathematics and Mechanics, 32*(6), 739–748.

Wang, Q., Liu, F., & Wang, X. (2014). Multi-objective optimization of machining parameters considering energy consumption. *The International Journal of Advanced Manufacturing Technology, 71*(5–8), 1133–1142.

Wibowo, W. K., & Jeong, S.-K. (2013). Genetic algorithm tuned PI controller on PMSM simplified vector control. *Journal of Central South University, 20*(11), 3042–3048.

Wright, J., & Farmani, R. (2001). The simultaneous optimization of building fabric construction, HVAC system size, and the plant control strategy. In The Proceedings of the 7th IBPSA Conference: Building Simulation, Rio de Janeiro, August, 2001 (Vol. 2, pp. 865–872).

Yang, Y., Wang, L., Wang, Y., Bi, Z., Xu, Y., & Pan, S. (2014). Modeling and optimization of two-stage procurement in dual-channel supply chain. *Information Technology and Management, 15*(2), 109–118.

Yao, X., Lin, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*(2), 82–102.

Zargari, A., Hooshmand, R., & Ataei, M. (2012). A new control system design for a small hydro-power plant based on particle swarm optimization-fuzzy sliding mode controller with Kalman estimator. *Transactions of the Institute of Measurement and Control, 34*(4), 388–400.

Zhao, D., & Yi, J. (2006). GA-based control to swing up an acrobot with limited torque. *Transactions of the Institute of Measurement and Control, 28*(1), 3–13.

Zhou, X. C., Zhao, Z. X., Zhou, K. J., & He, C. H. (2012). Remanufacturing closed-loop supply chain network design based on genetic particle swarm optimization algorithm. *Journal of Central South University, 19*(2), 482–487.

Zitzler, E., & Thiele, L. (1999). Multi-objective evolutionary algorithms: A comparative case study. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.