Constrained Multiple-Swarm Particle Swarm Optimization Within a Cultural Framework

Moayed Daneshyari, *Member, IEEE*, and Gary G. Yen, *Fellow, IEEE*

***Abstract*—Particle swarm optimization (PSO) has been recently adopted to solve constrained optimization problems. In this paper, a cultural-based constrained PSO is proposed to incorporate the information of the objective function and constraint violation into four sections of the belief space, specifically normative knowledge, spatial knowledge, situational knowledge, and temporal knowledge. The archived information facilitates communication among swarms in the population space and assists in selecting the leading particles in three different levels: personal, swarm, and global levels. Comprehensive comparison of the proposed heuristics over a number of benchmark problems with selected state-of-the-art constraint-handling techniques demonstrates that the proposed cultural framework helps the multiple-swarm PSO to perform competitively with respect to selected designs.**

***Index Terms*—Constrained optimization, constrained particle swarm optimization (CPSO), cultural algorithm (CA), PSO.**

I. Introduction

POPULATION-BASED paradigms to solve constrained optimization problems have attracted much attention during the most recent years. Genetic-based algorithms and swarm-based paradigms are two popular population-based heuristics introduced for solving constrained optimization problems [1]–[3]. Particle swarm optimization (PSO) [4]–[10] is a swarm intelligence design based upon mimicking the behavior of social species such as flocking birds, schooling fish, swarming wasps, and so forth. Constrained PSO (CPSO) is a relatively new approach to tackle constrained optimization problems [11]–[24]. What constitute the challenges of constrained optimization problems are various limits on decision variables, the types of constraints involved, the interference among constraints, and the interrelationship between the constraints and the objective functions. In general, constrained optimization problems can be formulated as

$$\text{Optimize} \quad f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) \tag{1}$$

subject to inequality constraints

$$g_k(\mathbf{x}) \leq 0, \qquad k = 1, 2, \ldots, L \tag{2}$$

and equality constraints

$$h_k(\mathbf{x}) = 0, \qquad k = L+1, \ldots, m. \tag{3}$$

We should note that, in this paper, minimization problems are considered without loss of generality (due to duality principle). Individuals that satisfy all of the constraints are called feasible individuals, while individuals that do not satisfy at least one of the constraints are called infeasible individuals. Active constraints are defined as inequality constraints that satisfy $g_k(\mathbf{x}) \leq 0$ $(k = 1, 2, \ldots, L)$ at the global optimum solution, so all equality constraints, i.e., $h_k(\mathbf{x}) = 0$ $(k = L+1, \ldots, m)$, are active constraints.

Although there are a few research works on PSO proposed to solve constrained optimization problems, none of these studies fully explores the information from all particles in order to perform communication within PSO. Therefore, due to lack of communication, the particles will not be able to act synchronously. When particles share their information through communication with each other, they will be able to efficiently handle the constraints and optimize the objective function. In order to construct the environment needed to share information, we need to build the groundwork to enable us to employ this information as needed. In this paper, the foundation is the belief space under cultural framework [25], [26]. Cultural algorithm (CA) has alone shown its own ability to solve engineering problems [26]–[42], particularly some constrained optimization ones [36], [39]–[41]. The information-sharing process can be improved by migration of data within swarms.

From a sociological point of view, a study has shown that human societies will migrate from one place to another in order to counter their own life constraints and limitations as well as to reach a better economical, social, or political life [43]. People living in different societies migrate in spite of the different value systems and cultural distinctions. Indeed, cultural belief is an important factor affecting the issues underlying the migration phenomena [44]. As CA is one of the newly emerging nature-inspired computational paradigms, the inspiration from sociology and migration analogy places the proposed design in perspective.

On the other hand, information sharing will effectively reduce the computational complexity. Indeed, finding the appropriate information for communication within swarms can be computationally expensive if it is not performed in a systematic and categorical manner. One computational aspect is the difficulties of finding the appropriate information to communicate within PSO in order to be able to simultaneously handle the constraints and optimize the objective function. Using many

Manuscript received May 13, 2010; revised November 1, 2010 and March 14, 2011; accepted April 30, 2011. Date of publication August 12, 2011; date of current version February 17, 2012.

M. Daneshyari is with the Department of Technology, Elizabeth City State University, Elizabeth City, NC 27909 USA (e-mail: mdaneshyari@mail.ecsu.edu).

G. G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: gyen@okstate.edu).

Digital Object Identifier 10.1109/TSMCA.2011.2162498

concepts inspired from CA, such as normative knowledge, situational knowledge, spatial knowledge, and temporal knowledge, we will be able to efficiently and effectively organize the knowledge acquired from the evolutionary process to facilitate PSO's updating mechanism as well as swarm communications. The interswarm communication for the constrained optimization problems using PSO is an important duty that cannot be solved unless we have access to the knowledge throughout the search process, given CA as the computational framework.

In this paper, we have proposed a computational framework based on CA adopting the knowledge stored in the belief space in order to assist the interswarm communication, to search for the leading particles in the personal, swarm, and global levels. Every particle in CPSO will fly through a three-level flight. Then, particles cluster into several swarms, and interswarm communication takes place to share the information. The remaining sections complete the presentation of this paper as follows. In Section II, we briefly review principles of CA and related works in CPSO performed in this area. In Section III, the proposed cultural CPSO is elaborated. In Section IV, simulation results are evaluated on the benchmark test problems in comparison with the state-of-the-art constraint-handling models. Finally, Section V summarizes the concluding remarks and future directions of this paper.

II. Literature Survey

A. Related Works in Constrained PSO

Relevant works of CPSO algorithms are briefly reviewed in this subsection to motivate the proposed ideas. PSO [4]–[10] has shown its promise to solve constrained optimization problems. Hu and Eberhart simply generated particles in PSO for constrained optimization problems until they are located in the feasible region and then used these particles in the feasible region for finding the best personal and global particles [11]. Parsopoulos and Vrahatis used a dynamic multistage penalty function for constraint handling [12]. The penalty function is the weighted sum of all constraint violations, with each constraint having a dynamic exponent and a multistage dynamic coefficient. Coath and Halgamuge presented a comparison of two constraint-handling methods based upon preserving feasible solutions [11] and dynamic penalty function [12] to solve constrained nonlinear optimization problems using PSO [13]. It demonstrated that the convergence rate for penalty-function-based PSO was faster than that of the feasible solution method.

Paquet and Engelbrecht proposed a modified PSO to solve linearly constrained optimization problems [14]. An essential characteristic of their modified PSO is that the movement of the particles in the vector space is mathematically guaranteed by the velocity and position update mechanism of PSO. They proved that their modified PSO is always assured to find at least a local optimum for linearly constrained optimization problems. Takahama and Sakai, in their □-constrained PSO, proposed an algorithm in which particles that satisfy the constraints move to optimize the objective function while particles that violate the constraints progress to satisfy the constraints [15]. In order to adaptively control the maximum velocity of particles, particles are divided into some groups, and their movement in those groups is compared.

Krohling and Coelho adopted Gaussian distribution instead of uniform distribution for the personal- and global-term random weights of the PSO mechanism to solve constrained optimization problems formulated as min–max problems [16]. They used two populations simultaneously. First, PSO focuses on evolving the variable vector while the vector of Lagrangian multiplier is kept frozen, and the second PSO is to concentrate on evolving the Lagrangian multiplier while the first population is stayed frozen. The use of normal distribution for the stochastic parameters of PSO seems to provide a good compromise between the probability of having a large number of small amplitudes around the current points and the small probability of having large amplitudes, which may cause particles to move away from the current points and escape from the local optima.

Yang *et al.* [17] proposed a master–slave PSO in which the master swarm is responsible for optimizing the objective function while the slave swarm is focused on constraint feasibility. Particles in the master swarm only fly toward the current better particles in the feasible region. The slave swarm is responsible for searching feasible particles by scouting through the infeasible region. The feasible/infeasible leaders from a swarm will then communicate to lead the other swarm. By exchanging flight information between swarms, the algorithm can explore a wider solution space.

Zheng *et al.* [18] adopted an approach that congregates the neighboring particles in PSO to form multiple swarms in order to explore an isolated, long, and narrow feasible space. They also applied a mutation operator with dynamic mutation rate to encourage flight of particles to the feasible region more frequently. For constraint handling, a penalty function has been adopted as to how far the infeasible particle is located from the feasible region. Saber *et al.* [19] introduced a version of PSO for constrained optimization problems. In their version of PSO, the velocity update mechanism uses a sufficient number of promising vectors to reduce randomness for better convergence. The velocity coefficient in the positional update equation is a dynamic rate depending on the error and iteration. They also reinitialized the idle particles if there are not improving for some iterations.

Li *et al.* [20] proposed dual PSO with stochastic ranking to handle the constraints. One regular PSO evolves simultaneously along with a genetic PSO which is a discrete version of PSO including a reproduction operator. The better of the two positions generated by these two PSOs is then selected as the updated position. Flores-Mendoza and Mezura-Montes [21] used the Pareto dominance concept for constraint handling on a biobjective space, with one objective being the sum of inequality constraint violations and the second objective being the sum of equality constraint violations, in order to promote a better approach to the feasible region. They also adopted a decaying parameter to control the constriction factor and global acceleration of PSO to prevent premature convergence and to advance the exploration of the search space. Ting *et al.* [22] introduced a hybrid heuristic consisting of PSO and genetic algorithm to tackle the constraint optimization problem of load flow problems. They adopted two-point crossover, mutation,

and roulette-wheel selection from genetic algorithms along with the regular PSO to generate the new population space. Liu *et al.* [23] incorporated discrete genetic PSO with differential evolution (DE) to enhance the search process in which both genetic PSO and DE update the position of the individual at every generation. The better position will then be selected.

Yen and Leong [24] embedded the constraint-handling techniques into the flight mechanism of PSO, including separate procedures to update the infeasible and feasible personal bests in order to guide the infeasible individuals toward the feasible regions while promoting the search for optimal solutions. Additionally, storing infeasible nondominated solutions along with the best feasible solutions in the global best archive is to assist the search for feasible regions and a better solution. The adjustment of accelerated constants is based on the number of feasible personal bests and the constraint violations of personal bests and global best. The simulation study shows that the proposed design is able to obtain a quality solution in a very efficient manner.

In the context of information sharing, all existing algorithms are short of a data feedback process which we have solved in our proposed algorithm by adopting a cultural framework. In the proposed algorithm, the population and belief spaces will communicate through feedbacklike communication channels. This feedbacklike procedure will increase the efficiency and effectiveness of the PSO mechanism in the search process.

B. Related Work in CA on Constrained Optimization

For the completeness of the presentation, the basic principle of CA is briefly outlined hereinafter. CA, which is originated by Reynolds [25], [26], is a dual-inheritance system where information exists at two different spaces, namely, population and belief spaces, and can pass along to the next generation. CA, an adaptive evolutionary computation method derived from cultural evolution, consists of evolving agents whose experiences are gathered into the belief space consisting of various forms of symbolic knowledge. CA has shown its ability to solve different types of problems [26]–[42].

Researchers have identified five basic sections of knowledge stored in the belief space based upon the literature in cognitive science and semiotics: situational knowledge, normative knowledge, spatial or topographical knowledge [29], domain knowledge, and temporal or history knowledge [34]. Situational knowledge is a set of exemplary individuals useful for experiences of all individuals. Situational knowledge guides all individuals to move toward the exemplar individuals. Normative knowledge consists of a set of promising ranges. Normative knowledge provides a standard guiding principle within which individual adjustments can be made. Individuals jump into the good range using normative knowledge. Topographical knowledge keeps track of the best individuals found so far in the promising region. Topographical knowledge leads all individuals toward the best performing areas in the search space. Domain knowledge incorporates information from the problem domain to lead the search. Domain knowledge about landscape contour and its related parameters guides the search process. History knowledge keeps track of the history of the search process and records key events in the search. It might be either a considerable move in the search space or a discovery of landscape change. Individuals use history knowledge for guidance in selecting a move direction. Domain knowledge and history knowledge are useful on dynamic landscape problems [45].

Becerra and Coello Coello proposed a cultured DE for constrained optimization [35]. The population space in their study was DE, while the belief space consisted of situational, topographical, normative, and history knowledge. The variation operator in DE was influenced by the knowledge source of the belief space. Yuan *et al.* introduced chaotic hybrid CA for constrained optimization in which the population space was DE and the belief space included normative and situational knowledge [40]. They incorporated a logistic map function for better convergence of DE. Tang and Li proposed a cultured genetic algorithm for constrained optimization problems by introducing a triple-space CA [41]. The triple space includes belief and population spaces in addition to an anticulture population consisting of individuals disobeying the guidance of the belief space and going away from the belief-space-guided individuals. The effect of disobeying enhanced by some mutation operations appreciably makes the algorithm faster and less risky for premature convergence by awarding the most successful individuals and punishing the unsuccessful population. Zhao and Gao [42] introduced a cultural-based PSO to design a low-pass finite-impulse response digital filter. The key idea behind their algorithm is to acquire problem-solving knowledge (i.e., beliefs) from the evolving population and, in turn, use that knowledge to guide the evolution process. They adopt the structure of CA including belief space, acceptance, and influence functions along with particle swarm for designing the filter.

III. Cultural Constrained Optimization Using Multiple-Swarm PSO

The pseudocode of the proposed design is shown in Fig. 1, and the block diagram depicting the operations of the proposed algorithm is also shown in Fig. 2, which is based upon the general structure of CA consisting of two spaces (belief and population spaces) and communication channels (influence function from belief space to population space and acceptance function from population space to belief space). The population space (PSO) will be initialized and then clustered into several swarms based upon the proximity of the particles in the decision space. The correspondent belief space (BLF) will then be initialized. We evaluate the population space using fitness values. Acceptance function is applied to select particles, which will be used for the belief space. The belief space consists of four sections: normative, spatial (topographical), situational, and temporal (history) knowledge. This cultural framework plays a key role in the algorithm. Influence function is then applied to the belief space to adjust the key parameters of PSO for the next iteration, i.e., personal best, swarm best, and global best. After a predefined iteration, influence function manipulates the belief space to perform communication among swarms, which is done by preparing two sets of particles for each swarm to share with

Initialize PSO at t=0.
Initialize BLF at t=0.
Repeat
- Evaluate PSO(t).
- Cluster PSO(t) into several swarms using k-means.
- Apply ACCEPTANCE function to PSO(t) to select particles which affect BLF(t).
- Adjust BLF(t) including Normative, Spatial, Situational, and Temporal Knowledge.
- Apply INFLUENCE function to BLF(t) to select pbest(t), sbest(t), and gbest(t) of PSO(t).
- If t=$T_{migration}$, perform cultural-based inter-swarm communication.
- t=t+1.
- Perform flight in PSO(t) using new pbest(t), sbest(t), and gbest(t).

Until Termination Criteria are met.
End

Fig. 1. Pseudocode of the cultural constrained optimization using multiple-swarm PSO.

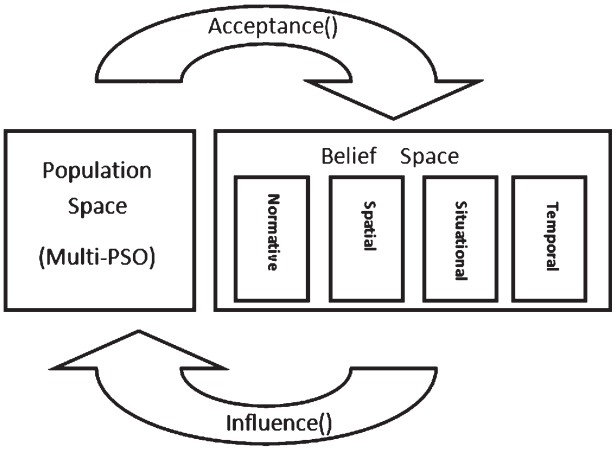

Fig. 2. Schema of the cultural framework adopted, where belief space consists of normative, spatial (topographical), situational, and temporal (history) knowledge, and population space is a multiple-swarm PSO.

the other swarms. Afterward, particles in the population space fly using newly computed personal, swarm, and global best. This process continues until the stopping criteria are met.

In the remainder of this section, we explain the multiswarm population space, acceptance function, different parts of belief space, influence functions, and interswarm communication strategy, respectively.

A. Multiswarm Population Space

The population space in this paper consists of multiple swarms, with each swarm performing a PSO paradigm. The particles are clustered into a predefined number of swarms using k-means clustering in the decision space. In this paper, the number of swarms, P, is chosen roughly 10% of the population size, N

$$P = \lfloor 0.1N \rfloor \tag{4}$$

where $\lfloor \cdot \rfloor$ refers to a rounding operator. This multiple-swarm PSO is a modified version of the algorithm introduced by Yen and Daneshyari [46], [47]. To overcome the premature convergence problem of PSO and to promote the particles in a swarm sharing information among themselves, a three-level flight for a PSO mechanism has been adopted. In the personal level, the particle will follow its best experienced behavior in its history. In the swarm level, the particle will simultaneously follow the best behaving particle in its swarm to achieve a synchronal behavior among the neighboring particles. Finally, in the global level, the entire population will follow the best known particle seeking a global goal. This modified paradigm of PSO is formulated as

$$\begin{aligned} v_i^d(t+1) &= wv_i^d(t) + c_1 r_1 \left(pbest_i^d(t) - x_i^d(t)\right) \\ &\quad + c_2 r_2 \left(sbest_{i,j}^d(t) - x_i^d(t)\right) \\ &\quad + c_3 r_3 \left(gbest^d(t) - x_i^d(t)\right) \\ x_i^d(t+1) &= x_i^d(t) + v_i^d(t+1) \end{aligned} \tag{5}$$

where $v_i^d(t)$ is the dth dimension of the velocity of the ith particle at time t; $x_i^d(t)$ is the dth dimension of the position of the ith particle at time t; $pbest_i^d(t)$ is the dth dimension of the best past position of the ith particle at time t; $sbest_{i,j}^d(t)$ is the dth dimension of the best particle from swarm j in which particle i belongs; $gbest^d(t)$ is the dth dimension of the best particle of a population at time t; r_1, r_2, and r_3 are uniformly generated random numbers in the range of (0, 1); c_1, c_2, and c_3 are constant parameters representing the weights for personal, swarm, and global behaviors; and w is the momentum for the previous velocity.

B. Acceptance Function

The belief space should be affected by a selection of best individuals. Therefore, we select all particles which are located in the feasible space, and also $p\%$ of the infeasible particles that have the least violation of constraints, where p is a predefined value. This allows the infeasible individuals with minimum constraint violations to portray feasibility landscape.

C. Belief Space

The belief space in this paradigm consists of four sections: normative, spatial, situational, and temporal knowledge. Since the constrained optimization problems of our interest have static landscapes, we only implement these four sections because domain knowledge, the fifth element, is mainly useful when fitness landscape is dynamic. We will briefly explain the type of information, the way to represent the knowledge, and how to update the knowledge for each section of the belief space.

1) Normative Knowledge: Normative knowledge represents the best area with respect to the objective function values. It is represented as Fig. 3(a), where $\mathbf{f}(t) = [\mathbf{f}_1(t)\mathbf{f}_2(t)\ldots\mathbf{f}_N(t)]$ and $V(t) = [v_1(t)v_2(t)\ldots v_N(t)]$ (N is the number of particles). $\mathbf{f}_j(t)$ is a normalized objective function defined as follows:

$$\mathbf{f}_j(t) = \frac{f(\mathbf{x}_j) - f_j^{\min}(t)}{f_j^{\max}(t) - f_j^{\min}(t)}, \qquad j = 1, 2, \ldots, N \tag{6}$$

where $f(\mathbf{x}_j)$ is the objective function value for particle $\mathbf{x}_j$, $f_j^{\min}(t) = \min_{x \in X}(f(\mathbf{x}_j))$ is the lower bound of the objective function value on the jth particle at time t, and

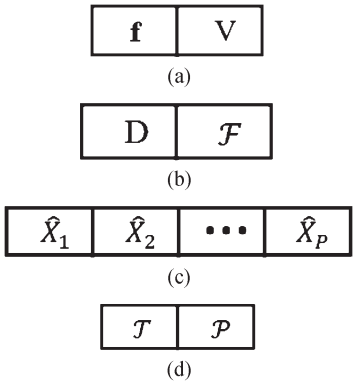

Fig. 3. Representation of (a) normative knowledge, (b) spatial knowledge, (c) situational knowledge, and (d) temporal knowledge.

$f_j^{\max}(t) = \max_{x \in X}(f(\mathbf{x}_j))$ is the upper bound of the objective function value on the jth particle at time t. X is referred to as the current population at time t. At every iteration, the lower and upper bounds will be calculated in the algorithm based upon the current population. They vary by iteration and are not fixed. In (6), the range of objective function values had been normalized. The reason for the normalization is that, throughout the algorithm, we do not deal with scaling. For cases with large values of $f_j^{\max}(t)$, the algorithm still works properly, because $f_j^{\max}(t)$ and $f_j^{\min}(t)$ come from the current population (not some fixed values), so the value of $f(\mathbf{x}_j) - f_j^{\min}(t)$ in the numerator of (6) will not be statistically very small compared to the denominator $f_j^{\max}(t) - f_j^{\min}(t)$. As a result, $\mathbf{f}_j(t)$ will still be a proper measure to handle objective functions with very large maximum values.

$v_j(t)$ is a measure of violation of all constraints for particle $\mathbf{x}_j$, which is defined as follows:

$$v_j(t) = \frac{1}{m}\sum_{k=1}^{m}\frac{c_k(\mathbf{x}_j)}{c_k^{\max}}, \qquad j = 1, 2, \ldots, N \tag{7}$$

where m is the number of constraints and $c_k(\mathbf{x}_j)$ is related to the kth constraint violation evaluated at particle $\mathbf{x}_j$ as follows:

$$c_k(\mathbf{x}_j) = \begin{cases} \max\left(0, g_k(\mathbf{x}_j)\right), & k = 1, 2, \ldots, L \\ \max\left(0, |h_k(\mathbf{x}_j)| - \delta\right), & k = L+1, \ldots, m \end{cases} \tag{8}$$

$$c_k^{\max} = \max_{\mathbf{x} \in X}\left(c_k(\mathbf{x}_j)\right). \tag{9}$$

In order to update normative knowledge, new objective function values will be normalized using (6), and constraint violation measures will be updated by the new position of the particles using (7). The information in normative knowledge is used to assemble the framework for spatial knowledge.

2) Spatial Knowledge: In order to represent spatial or topographical knowledge, we adopt normative knowledge. The method adopted in this section is similar to the penalty function method to handle constraints introduced by Tessema and Yen [48]. The normalized objective functions, $\mathbf{f}$, and violation measures, V, are set as the axes of a 2-D space, as shown in Fig. 4. Two particles are mapped in this space for visualization. Fig. 3(b) shows the spatial knowledge stored for every particle located in the $\mathbf{f}-\mathrm{V}$ space where $D(t) = [D_1(t)D_2(t)\ldots D_N(t)]$ and $\mathcal{F}(t) = [\mathcal{F}_1(t)\mathcal{F}_2(t)\ldots\mathcal{F}_N(t)]$ (N is the number of particles). $D_j(t)$ is the Euclidean distance from the origin of the $\mathbf{f}-V$ space, which is defined as

$$D_j(t) = \left(v_j(t)^2 + \mathbf{f}_j(t)^2\right)^{1/2}, \qquad j = 1, 2, \ldots, N \tag{10}$$

and $\mathcal{F}_j$ is the modified objective function value to handle constraints computed as a weighted sum of three spatial distances D, v, and $\mathbf{f}$, i.e.,

$$\mathcal{F}_j(t) = \begin{cases} D_j(t) + (1 - r(t))\, v_j(t) \\ \quad +r(t)\mathbf{f}_j(t), & r(t) \neq 0, \mathbf{x}_j \text{ infeasile} \\ v_j(t), & r(t) = 0, \mathbf{x}_j \text{ infeasile} \\ \mathbf{f}_j(t), & \mathbf{x}_j \text{ feasile,} \end{cases} \qquad j = 1, 2, \ldots, N \tag{11}$$

where $r(t)$ is the ratio of the number of feasible particles over the population size, and $\mathbf{f}_j(t)$ and $v_j(t)$ are defined in (6) and (7), respectively. If $0 \ll r(t) < 1$, then $v_j(t)$ will be more important than $\mathbf{f}_j(t)$ in (11); consequently, $\mathcal{F}_1(t) > \mathcal{F}_2(t)$ in the schema shown in Fig. 4, which means that particle 2 outperforms particle 1 for a minimization problem. However, if $0 < r(t) \ll 1$, then $\mathbf{f}_j(t)$ will be more important than $v_j(t)$ in (11); consequently, $\mathcal{F}_1(t) < \mathcal{F}_2(t)$ shown in Fig. 4, which implies that particle 1 outperforms particle 2. For the sake of completeness and to demonstrate that the modified objective function works well, we will elaborate all possible cases.

1) In case 1 of (11), i.e., $r(t) \neq 0$ and infeasible particle, there is at least one feasible particle. In this case, the modified objective function for the infeasible particle is formulated. It should incorporate both the violation factor and the normalized objective value. For the following two infeasible particles:
   a) If r (percentage of feasible particles) is too large, then objective function is more important than constraint violation in computing the modified objective value. The particle with the higher (worse in the minimization problem) objective function will have a higher (worse) modified objective value.
   b) If r is too small, then constraint violation is more important than objective function value. The particle with the higher constraint violation will have a higher (worse) modified objective value.
2) In case 2 of (11), i.e., $r(t) = 0$ and infeasible particle, there is no feasible particle. Therefore, the only important factor will be constraint violation. The particle with the higher constraint violation will have a higher (worse) modified objective value.
3) In case 3 of (11), i.e., feasible particles, there is at least one feasible particle. For two feasible particles, the constraint violation factor does not play a role, and only the objective function value is important. The particle with the higher objective function value will have a higher (worse) modified objective value.
4) When comparing two particles, i.e., one feasible and another infeasible, we should adopt cases 1 and 3 (since there is at least one feasible particle, r cannot be zero). In this comparison, using the geometrical graphs in Fig. 4,

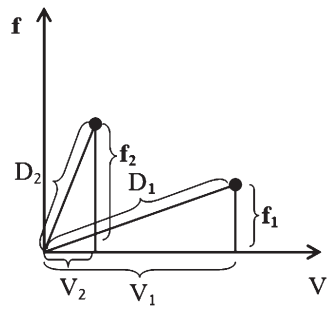

Fig. 4. Schema to represent how spatial knowledge is computed.

it can be observed that always the infeasible particle will have a higher (worse) modified objective value.

5) There is no comparison between cases 2 and 3. In addition, there will be no comparison between cases 1 and 2, because case 2 is when there is no feasible particle at all, and case 1 or 3 is when there exists at least one feasible particle.

At every iteration, spatial knowledge will be updated. To do so, the updated normative knowledge will be used to rebuild the spatial distance for every particle using (10) and (11). Spatial knowledge will be used later to find the global best particle in the population space and to build a communication strategy among swarms.

3) Situational Knowledge: This part of the belief space is used to keep the good exemplar particles for each swarm. Its representation is shown in Fig. 3(c). $\hat{X}_i(t)$ $[i = 1, 2, \ldots, P$, where P is the number of swarms defined in (4)] is the best particle in the ith swarm based upon the information received from spatial knowledge in accordance with both objective function value and constraint violation. Assume that at an arbitrary iteration, the ith swarm consists of N_i particles as $\Omega_i = \{z_1, z_2, \ldots, z_{N_i}\}$ and that $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{N_i}\}$ is a set consisting of the modified objective values extracted from spatial knowledge corresponding to $z_1, \boldsymbol{z}_2, \ldots, z_{N_i}$. $\hat{X}_i(t) \in \Omega_i$ is defined such that

$$\mathcal{F}\left(\hat{X}_i(t)\right) = \min_{1<l<N_i} \mathcal{F}_l, \qquad i = 1, 2, \ldots, P \tag{12}$$

where $\mathcal{F}(\hat{X}_i(t))$ is the modified objective function value for the particle $\hat{X}_i(t)$. In order to update situational knowledge, the updated position of the particles will be used to evaluate (6)–(11) to compute the updated modified objective function values, and then, the particle corresponding to the least value in each swarm will be stored in situational knowledge. Situational knowledge will be used later to compute the swarm best particles and to facilitate communication among swarms.

4) Temporal Knowledge: This part of the belief space is used to keep the history of the individual's behavior. Its representation is shown in Fig. 3(d), where $\mathcal{T}(t) = \{\mathcal{T}_1(t), \mathcal{T}_2(t), \ldots, \mathcal{T}_N(t)\}$ and $\mathcal{P}(t) = \{\mathcal{P}_1(t), \mathcal{P}_2(t), \ldots, \mathcal{P}_N(t)\}$ (N is the number of particles in the population space). $\mathcal{T}_j(t)$ is a set of past and current temporal patterns of the jth particle, which are collected at every time step from part of spatial knowledge, $\mathcal{F}_j(t)$, and is defined as follows:

$$\mathcal{T}_j(t) = \{\mathcal{F}_j(1), \mathcal{F}_j(2), \ldots, \mathcal{F}_j(t)\}, \quad j = 1, 2, \ldots, N \tag{13}$$

where $F_j(1), F_j(2), \ldots, F_j(t)$ are the modified objective function values defined in (11) for time steps $1, 2, \ldots t$, respectively. $\mathcal{P}_j(t)$ is a set of all past and current positions of the jth particle in the whole population, which is defined as $\mathcal{P}_j(t) = \{\mathbf{x}_j(1), \mathbf{x}_j(2), \ldots, \mathbf{x}_j(t)\}$, $j = 1, 2, \ldots, N$. Temporal knowledge will be updated at every iteration. To do so, the updated spatial knowledge, the updated position of the particle, and the previously stored temporal knowledge will be adopted as follows:

$$\begin{aligned} \mathcal{T}_j(t+1) &= \mathcal{T}_j(t) \cup \{\mathcal{F}_j(t+1)\}, \qquad j = 1, 2, \ldots, N \\ \mathcal{P}_j(t+1) &= \mathcal{P}_j(t) \cup \{\mathbf{x}_j(t+1)\}, \qquad j = 1, 2, \ldots, N. \end{aligned} \tag{14}$$

Temporal knowledge will later be used to compute the personal best for every particle in the population space.

D. Influence Functions

After the belief space is updated, correspondent knowledge should be used to influence the flight of particles in PSO. We propose to use the knowledge in the belief space to select the personal best, swarm best, and global best for the PSO flight mechanism. Furthermore, we propose to adopt the information in the belief space to perform a communication strategy among swarms.

1) pbest Selection: In order to select the personal best, we exploit the information in the temporal knowledge section of the belief space. The best behaving particle's past history should be selected as follows:

$$pbest_i(t) = \left\{\exists \mathbf{x}_i(\hat{t}) \in \mathcal{P}_i(t), |\mathcal{F}_i(\hat{t}) = \min_t\left(\mathcal{T}_i(t)\right)\right\}, \qquad i = 1, 2, \ldots, N \tag{15}$$

where $\mathcal{P}_i(t) = \{\mathbf{x}_i(1), \mathbf{x}_i(2), \ldots, \mathbf{x}_i(t)\}$ is a set of all past positions of the ith particle and $\mathcal{T}_i(t) = \{\mathcal{F}_i(1), \mathcal{F}_i(2), \ldots, \mathcal{F}_i(t)\}$ denotes the corresponding modified objective values for the past history of the ith particle, both extracted from the temporal knowledge section of the belief space.

2) sbest Selection: In order to select the swarm best particle, situational knowledge is adopted. The information stored in the situational knowledge section of the belief space is simply copied into the swarm best particles

$$sbest_i(t) = \hat{X}_i(t), \qquad i = 1, 2, \ldots, P \tag{16}$$

where P is the number of swarms and $\hat{X}_i(t)$ is the representation of the situational knowledge of the belief space.

3) gbest Selection: We use the spatial knowledge stored in the belief space to compute $gbest(t)$ at each iteration. The

global best particle is found as follows:

$$gbest(t) = \{\exists \mathbf{x}_j(t) \in \mathbb{P}(t), 1 \leq j \leq N | \mathcal{F}_j(t) = \min(\mathbb{F}(t))\} \tag{17}$$

where $\mathbb{P}(t) = \{\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_N(t)\}$ is the entire population of particles at time t and $\mathbb{F}(t) = \{\mathcal{F}_1(t), \mathcal{F}_2(t), \ldots, \mathcal{F}_N(t)\}$ is a set consisting of the modified objective function values for all particles at time t.

4) Interswarm Communication Strategy: After some predefined iterations, $T_{\text{migration}}$, swarms will perform information exchange. Each swarm prepares a list of sending particles to be sent to the next swarm and also assembles a list of replacement particles to be replaced by particles coming from other swarms. This communication strategy is a modified version of the algorithm adopted in [46]. We use the information stored in the belief space to perform communication among swarms. To do so, each swarm prepares two lists of particles, i.e., $\mathbb{S}_i$ and $\mathbb{R}_i$, $i = 1, 2, \ldots, P$, where P is the fixed number of swarms defined in (4). $\mathbb{S}_i$ is the list of particles in the ith swarm to be sent to the next swarm, and $\mathbb{R}_i$ is the list of particles in the ith swarm to be replaced by particles coming from another swarm. The interswarm communication strategy is based upon the particles' locations in the swarm and their modified objective value which is stored in the belief space. The sending list for the swarm is prepared in the following order.

1) The highest priority in the selection of particles is given to a particle that has the least average Hamming distance from others. The average Hamming distance between every particle and all other particles will be computed. To do so, the Hamming distance between two particles is calculated as a sum of absolute differences between the positions of two particles in different dimensions. The particle with the lowest value of the average Hamming distance will then be selected as the representative of the swarm.
2) The second priority is given to the closest particles to the representative particle whose modified objective value stored in the spatial knowledge of the belief space is greater than that of the representative.
3) The third priority is given to the closest particles to the representative particle whose modified objective value extracted from the belief space is less than that of the representative.

Note that depending on the predefined fixed value for the allowable number of the sending list, $N_{\text{migration}}$, the sending list will be filled in each swarm using the aforementioned priorities.

There will also be a replacement list that each swarm prepares, based upon the similar positional information of particles in the swarm. When swarms are approaching local optima, many particles' locations are the same. Each swarm will remove this excess information through its replacement list. The replacement list in each swarm is assembled in the following order.

1) The first priority is given to the particles with identical decision space information in the order of their modified objective values extracted from the belief space, with the least modified objective values being replaced first.
2) The second and last priority is given to the particles with the lowest modified objective values if all particles of the first priority have already been placed in the replacement list.

This information exchange among swarms happens in a ring sequential order between each pair of swarms. Each swarm accepts the sending list from the other swarm and will replace it with its own replacement list.

IV. Comparative Study

In this section, the performance of the cultural CPSO is evaluated against those of the selected state-of-the-art constrained optimization heuristics.

A. Parameter Settings

The parameters of the cultural CPSO are set as follows. The tolerance for equality constraints in (8), i.e., δ, is set as 0.0001, as suggested in the literature. In the flight mechanism, the momentum, w, is randomly selected from the uniform distribution of (0.5, 1), and the personal, swarm, and global accelerations, i.e., c_1, c_2, and c_3, are all selected as 1.5. Please note that the selection of parameters of the flight mechanism, such as momentum and accelerations, follows the same guidelines depicted in [38]. The population size is fixed at 100 particles. The maximum velocity for the particles in a specific dimension, $v^d_{\max}$, is set at half of the range of the particle's position in that dimension

$$v^d_{\max} = \left(x^d_{\max} - x^d_{\min}\right)/2. \tag{18}$$

The rate for information exchange among swarms affects how much swarms communicate with each other. A higher rate corresponds to more communication and better overall performance of the algorithm, but it does incur higher computational complexity, while a lower rate imposes less computational complexity and results in relatively poorer performance. The heuristic choice is set at 30%. The allowable number of migrating particles among swarms is set as 5% of the population size, which is $N_{\text{migration}} = 0.05N = 5$.

B. Benchmark Test Functions

The proposed cultural CPSO has been tested on 24 benchmark functions [49] to verify its performance. The characteristics on these test functions are summarized in Table I. These problems include various types of objective functions such as linear, nonlinear, quadratic, cubic, and polynomial. These benchmark problems vary in number of decision variables, n, i.e., between 2 and 24, and number of constraints, i.e., between 1 and 38. In this table, ρ is the estimated ratio of the feasible region over the search space, which varies by as low as 0.0000% and as high as 99.9971%. The numbers of different types of constraints are also shown for each test function: the number of linear inequality (LI), the number of nonlinear inequality

TABLE I
SUMMARY OF 24 BENCHMARK TEST FUNCTIONS

Pr	n	Type	ρ	LI	NI	L	N	a	Optimum
g01	13	Quadratic	0.0111%	9	0	0	0	6	-15.0000000000
g02	20	Nonlinear	99.9971%	0	2	0	0	1	-0.8036191042
g03	10	Polynomial	0.0000%	0	0	1	1	1	-1.0005001000
g04	5	Quadratic	52.1230%	0	6	0	0	2	-30665.5386717834
g05	4	Cubic	0.0000%	2	0	3	3	3	5126.4967140071
g06	2	Cubic	0.0066%	0	2	0	0	2	-6961.8138755802
g07	10	Quadratic	0.0003%	3	5	0	0	6	24.3062090681
g08	2	Nonlinear	0.8560%	0	2	0	0	0	-0.0958250415
g09	7	Polynomial	0.5121%	0	4	0	0	2	680.6300573745
g10	8	Linear	0.0010%	3	3	0	0	6	7049.2480205286
g11	2	Quadratic	0.0000%	0	0	0	1	1	0.7499000000
g12	3	Quadratic	4.7713%	0	1	0	0	0	-1.0000000000
g13	5	Nonlinear	0.0000%	0	0	0	3	3	0.0539415140
g14	10	Nonlinear	0.0000%	0	0	3	0	3	-47.7648884595
g15	3	Quadratic	0.0000%	0	0	1	1	2	961.7150222899
g16	5	Nonlinear	0.0204%	4	34	0	0	4	-1.9051552586
g17	6	Nonlinear	0.0000%	0	0	0	4	4	8853.5396748064
g18	9	Quadratic	0.0000%	0	13	0	0	6	-0.8660254038
g19	15	Nonlinear	33.4761%	0	5	0	0	0	32.6555929502
g20	24	Linear	0.0000%	0	6	2	12	16	0.2049794002
g21	7	Linear	0.0000%	0	1	0	5	6	193.7245100700
g22	22	Linear	0.0000%	0	1	8	11	19	236.4309755040
g23	9	Linear	0.0000%	0	2	3	1	6	-400.0551000000
g24	2	Linear	79.6556%	0	2	0	0	2	-5.5080132716

(NI), the number of linear equality, and the number of nonlinear equality. In this table, a is the number of active constraints at the known optimal solution, $\widetilde{\mathbf{x}}$, and $f(\widetilde{\mathbf{x}})$ is the objective function of the known optimal solution [49].

C. Simulation Results

The experiments reported in this paper are performed on a computer with 1.66-GHz dual-core processor and 1-GB RAM operating on Windows XP Professional. The programs are written in Matlab. We performed extensive experiments on all 24 benchmark test functions based upon the comparison methods suggested in [49]. For three different function evaluations (FEs) of 5000, 50 000, and 500 0000, the objective function error values, $f(\mathbf{x}) - f(\widetilde{\mathbf{x}})$, are found, while $f(\widetilde{\mathbf{x}})$ is the best known solution [49] presented in the rightmost column in Table I. Notice that when $f(\mathbf{x}) - f(\widetilde{\mathbf{x}}) < 1e-10$, the final error is considered as zero. For each benchmark test problem, a total of 25 independent runs are performed.

The statistical measures, including the best, median, worst, mean, and standard deviations, are then computed. These results are tabulated in Table II. For the best, median, and worst solutions, the number of constraints that cannot satisfy the feasibility condition is found and shown as an integer inside the parenthesis after the best, median, and worst solutions, respectively, in this table. The parameter c shows three different integers demonstrating the number of constraints, including the equality and inequality ones that are violated by more than 1, 0.01 and 0.0001, respectively, for the median solution. The parameter $\overline{v}$ indicates the average value of the violations of all constraints at the median solution, as defined in [49].

For each independent run, the number of FEs to locate a solution satisfying $f(\mathbf{x}) - f(\widetilde{\mathbf{x}}) < 0.0001$ is recorded. For each benchmark function, the statistical measures of these 25 runs, including the best, median, worst, mean, and standard deviations, are then computed. These results are shown in Table III. In the same table, feasible rate, success rate, and success performance are also calculated for each test function. Feasible rate is a ratio of feasible runs over total runs, where feasible run is defined as a run with maximum FEs of 500 000 during which at least one feasible solution is found. Success rate is a ratio of successful runs over total runs, where successful run is defined as a run during which the algorithm finds a feasible solution, $\mathbf{x}$, satisfying $f(\mathbf{x}) - f(\widetilde{\mathbf{x}}) < 0.0001$. Success performance is defined as [49]

$$\text{success performance} = \frac{\text{mean}\{\text{FEs for successful runs}\} \times (\text{no. of total runs})}{\text{no. of successful runs}}. \quad (19)$$

TABLE II
ERROR VALUES FOR DIFFERENT FES ON TEST PROBLEMS g01–g24

FEs	Prob.	g01	g02	g03	g04	g05	g06
5×10^3	Best	1.4372e1(0)	3.5723e-1(0)	5.8738e-1(0)	1.2328e2(0)	6.4758e2(4)	6.9146e1(0)
	Median	9.8536 (4)	4.5976e-1(0)	8.9452e-1(0)	6.4738e2(0)	8.6193e2(4)	8.4628e2(0)
	Worst	1.9525(7)	5.4657e-1(0)	1.12648(0)	9.4384e2(0)	1.5495e3(4)	2.3859e3(0)
	c	(2, 4, 4)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(4, 4, 4)	(0, 0, 0)
	$\overline{v}$	3.4517e-1	0	0	0	2.53456e1	0
	Mean	8.3780	4.6576e-1	9.9473e-1	6.3810e2	8.5907e2	7.1844e2
	Std.	3.3715	3.7841e-2	1.4528e-1	1.4925e2	4.8496e2	5.6820e2
5×10^4	Best	2.4729e-10(0)	1.4365e-2(0)	0(0)	6.3404e-8(0)	8.4357e-7(0)	4.9348e-6 (0)
	Median	3.5467e-10(0)	3.1324e-2(0)	0(0)	2.3748e-7(0)	7.5597e-7(0)	6.9834e-6(0)
	Worst	4.0234e-10(0)	5.9435e-2(0)	0(0)	7.8263e-6(0)	4.9528e-6(0)	8.5197e-6(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	0	0	0
	Mean	3.6294e-10	3.1048e-2	0	2.9230e-7	7.7823e-7	7.0125e-6
	Std.	4.5637e-12	1.6403e-2	0	4.3839e-7	1.8347e-7	5.9238e-7
5×10^5	Best	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Median	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Worst	0(0)	1.9543e-2(0)	0(0)	0(0)	0(0)	0(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	0	0	0
	Mean	0	1.9659e-3	0	0	0	0
	Std.	0	4.7549e-3	0	0	0	0
FEs	Prob.	g07	g08	g09	g10	g11	g12
5×10^3	Best	4.3452e1(0)	7.6478e-8(0)	9.5829(0)	5.3675e3(0)	2.5643e-4(0)	4.5645e-8(0)
	Median	2.6788e2(0)	3.2784e-4(0)	5.3950e1(0)	6.8574e3(2)	5.8274e-3(0)	3.5965e-5(0)
	Worst	3.9643e3(1)	8,5367e-1(0)	4.7204e2(0)	7.4534e2(4)	3.9837e-2(0)	1.6754e-2(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 2, 3)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	2.4545e-2	0	0
	Mean	2.8642e2	4.8947e-4	5.0025e1	8.3554e3	6.9445e-3	8.5645e-4
	Std.	4.8034e2	7.3674e-3	2.6584e1	5.8689e3	4.5685e-3	6.1904e-3
5×10^4	Best	0(0)	0(0)	0(0)	4.2219e-7(0)	5.9854e-9(0)	0(0)
	Median	0(0)	0(0)	0(0)	3.9540e-6(0)	4.0546e-7(0)	0(0)
	Worst	0(0)	0(0)	0(0)	6.4859e-6(0)	6.9434e-5(0)	0(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	0	0	0
	Mean	0	0	0	4.0143e-6	7.8687e-6	0
	Std.	0	0	0	1.9344e-7	8.9676e-6	0
5×10^5	Best	0(0)	0(0)	0(0)	1.3494e-9(0)	0(0)	0(0)
	Median	0(0)	0(0)	0(0)	4.6015e-8(0)	0(0)	0(0)
	Worst	0(0)	0(0)	0(0)	9.5246e-8(0)	0(0)	0(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	0	0	0
	Mean	0	0	0	4.5064e-8	0	0
	Std.	0	0	0	7.0345e-9	0	0
FEs	Prob.	g13	g14	g15	g16	g17	g18
5×10^3	Best	6.8764(3)	-4.3950e1(3)	3.4289(2)	2.4959e-1(0)	3.5859e2(4)	3.8494(12)
	Median	8.2840(3)	-2.0960e2(3)	4.3395(2)	4.5851e-1(0)	6.2048e2(4)	4.5005(12)
	Worst	1.3940e1(3)	-2.3849e2(3)	5.3859(2)	7.4930e-1(2)	9.8363e2(4)	6.0375(12)
	c	(0, 3, 3)	(3, 3, 3)	(0, 2, 2)	(0, 0, 0)	(4, 4, 4)	(10, 11, 11)
	$\overline{v}$	1.3947	7.0902	1.4759e-1	0	8.3839e1	9.3849
	Mean	7.3904	-2.0035e2	4.2174	4.3735e-1	5.9303e2	4.6720
	Std.	1.8473	6.2387e1	2.6102	1.8276e-1	9.9278e1	1.8494
5×10^4	Best	2.3894e-9(0)	0(0)	0(0)	4.4748e-8(0)	2.1273e1(0)	0(0)
	Median	4.9694e-6(0)	0(0)	0(0)	1.9323e-4(0)	6.2893e1(0)	0(0)
	Worst	6.3938e-1(0)	0(0)	3.5796e-5(0)	2.4385e-2(0)	8.4849e1(0)	1.4634e-7(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	0	0	0
	Mean	5.9404e-2	0	3.7594e-7	2.5782e-4	3.8373e1	8.7561e-9
	Std.	3.8949e-1	0	4.2893e-4	6.4839e-3	3.2394e1	6.9661e-2

TABLE II
(Continued.) ERROR VALUES FOR DIFFERENT FEs ON TEST PROBLEMS g01–g24

5×10^5	Best	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Median	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Worst	6.8495e-8(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	c	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	0	0	0	0	0
	Mean	4.8055e-9	0	0	0	0	0
	Std.	2.5855e-6	0	0	0	0	0
FEs \ Prob.		g19	g20	g21	g22	g23	g24
5×10^3	Best	3.9605e2(0)	5.6996 (12)	7.5479e1(5)	8.4563e3(19)	4.2033e2(4)	8.4834e-4(0)
	Median	5.0387e2(0)	1.3656e1(19)	1.8977e2(5)	9.7685e3(19)	6.2017e2(5)	9.5092e-3(0)
	Worst	6.4760e2(0)	1.9574e1(17)	5.7689e2(5)	9.9964e3(19)	9.3945e2(6)	6.9804e-2(0)
	c	(0, 0, 0)	(5, 16, 16)	(1, 4, 6)	(19, 19, 19)	(2, 5, 6)	(0, 0, 0)
	$\overline{v}$	0	2.8796	4.8632	8.6785e7	1.8495	0
	Mean	4.8792e2	1.4098e1	2.6778e2	1.1205e4	5.6996e2	1.0034e-2
	Std.	9.7634e1	1.7860e1	3.6781e2	4.8754e3	3.4856e2	1.8075e-2
5×10^4	Best	8.9457e-8(0)	3.6759e-1(16)	8.9865e-5(0)	6.657(4)	4.7893e-4(0)	0(0)
	Median	3.6790e-6(0)	3.6758(16)	4.6453e-3(0)	2.4567e3(16)	2.6778e-3(0)	0(0)
	Worst	1.9426e-5(0)	7.9865(20)	6.0965(0)	5.7685e4(19)	8.5623e-2(0)	0(0)
	c	(0, 0, 0)	(2, 5, 8)	(0, 0, 0)	(3, 8, 16)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	8.9863e-1	0	2.5673e1	0	0
	Mean	4.9453e-6	3.7396	7.8757e-1	7.5678e3	7.5610e-3	0
	Std.	5.8438e-6	1.1930	8.9868	6.9868e3	3.7609e-2	0
5×10^5	Best	0(0)	-3.0694e-2(18)	6.9854e-8(0)	1.4568(0)	0(0)	0(0)
	Median	0(0)	-2.4096e-2(16)	6.7685e-6(0)	7.9653e1(0)	0(0)	0(0)
	Worst	0(0)	-2.0129e-2(19)	9.0956e-6(0)	1.3576e2(0)	0(0)	0(0)
	c	(0, 0, 0)	(1, 4, 6)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	$\overline{v}$	0	1.3459e-2	0	0	0	0
	Mean	0	-2.5001e-2	2.5609e-6	9.7685e1	0	0
	Std.	0	4.6950e-3	5.8796e-6	3.5475e1	0	0

TABLE III
NUMBER OF FEs TO ACHIEVE THE FIXED ACCURACY LEVEL ($f(\mathbf{x}) - f(\widetilde{\mathbf{x}}) < 0.0001$), SUCCESS RATE, FEASIBILITY RATE, AND SUCCESS PERFORMANCE

Prob.	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success Performance
g01	24786	27348	49601	35834	11639	100%	100%	35834
g02	56392	93674	500000	184530	173487	100%	76%	242803
g03	26498	28564	29129	28602	673.86	100%	100%	28602
g04	25983	26934	27045	26903	403.91	100%	100%	26903
g05	29629	31897	32983	30961	693.52	100%	100%	30961
g06	27688	29549	30189	29429	503.59	100%	100%	29429
g07	26024	28388	30877	28109	458.15	100%	100%	28109
g08	2302	5280	8938	5418.4	1935.4	100%	100%	5418.4
g09	30178	31866	32353	31327	331.57	100%	100%	31327
g10	26356	27990	29234	28028	459.09	100%	96%	29196
g11	4589	10678	31878	12897	10558	100%	100%	12897
g12	3289	7580	10454	6738.1	1378.5	100%	100%	6738.1
g13	31897	36878	256891	47895	43788	100%	100%	47895
g14	24678	28512	48724	26980	3589.2	100%	100%	26980
g15	30219	31029	32064	30984	335.76	100%	100%	30984
g16	28373	31795	69374	42750	2647.3	100%	100%	42750
g17	158367	193045	273890	210454	42084	100%	92%	228754
g18	28504	30496	62567	37575	6467	100%	100%	37575
g19	21345	23768	27910	24502	1032	100%	100%	24502
g20	-	-	-	-	-	0%	0%	-
g21	37385	122705	197614	141639	39574	100%	96%	147541
g22	-	-	-	-	-	100%	0%	-
g23	62091	182065	500000	259393	112038	100%	100%	259393
g24	17364	19391	29047	18972	4283	100%	100%	18972

TABLE IV
SUMMARY OF STATISTICAL RESULTS FOUND BY CULTURAL CPSO (IS DENOTED FOR INFEASIBLE SOLUTION)

Prob.	Optimal	Best	Median	Mean	Worst	Std. Dev.
g01	-15.0000000000	-15.0000000000	-15.0000000000	-15.0000000000	-15.0000000000	0.0000e0
g02	-0.8036191042	-0.8036191042	-0.8036191042	-0.8016532042	-0.7840761042	4.6784e-3
g03	-1.0005001000	-1.0005001000	-1.0005001000	-1.0005001000	-1.0005001000	3.6759e-13
g04	-30665.5386717834h	-30665.5386717834	-30665.5386717834	-30665.5386717834	-30665.5386717834	1.7890e-16
g05	5126.4967140071	5126.4967140071	5126.4967140071	5126.4967140071	5126.4967140071	6.0912e-12
g06	-6961.8138755802	-6961.8138755802	-6961.8138755802	-6961.8138755802	-6961.8138755802	3.8095e-11
g07	24.3062090681	24.3062090681	24.3062090681	24.3062090681	24.3062090681	1.3724e-12
g08	-0.0958250415	-0.0958250415	-0.0958250415	-0.0958250415	-0.0958250415	7.8088e-11
g09	680.6300573745	680.6300573745	680.6300573745	680.6300573745	680.6300573745	5.8797e-17
g10	7049.2480205286	7049.2480205299494	7049.248020574615	7049.248020573664	7049.248020623846	6.9806e-7
g11	0.7499000000	0.7499000000	0.7499000000	0.7499000000	0.7499000000	4.6756e-17
g12	-1.0000000000	-1.0000000000	-1.0000000000	-1.0000000000	-1.0000000000	1.7648e-14
g13	0.0539415140	0.0539415140	0.0539415140	0.0539415188	0.0539415825	1.5409e-7
g14	-47.7648884595	-47.7648884595	-47.7648884595	-47.7648884595	-47.7648884595	6.7830e-11
g15	961.7150222899	961.7150222899	961.7150222899	961.7150222899	961.7150222899	2.6598e-16
g16	-1.9051552586	-1.9051552586	-1.9051552586	-1.9051552586	-1.9051552586	3.9578e-13
g17	8853.5396748064	8853.5396748064	8853.5396748064	8853.5396748064	8853.5396748064	1.5329e-11
g18	-0.8660254038	-0.8660254038	-0.8660254038	-0.8660254038	-0.8660254038	8.0934e-14
g19	32.6555929502	32.6555929502	32.6555929502	32.6555929502	32.6555929502	5.9083e-12
g20	0.2049794002	0.1742854002 (**IS**)	0.1435914002 (**IS**)	0.01128974002 (**IS**)	0.1848504002 (**IS**)	7.3832e-2
g21	193.7245100700	193.7245101398	193.7245168385	193.7245126309	193.7245191656	4.6482e-5
g22	236.4309755040	237.887775504	316.083975504	334.115975504	372.190975504	1.5438e2
g23	-400.0551000000	-400.0551000000	-400.0551000000	-400.0551000000	-400.0551000000	6.2319e-11
g24	-5.5080132716	-5.5080132716	-5.5080132716	-5.5080132716	-5.5080132716	5.8794e-15

These tables show that feasible solutions can be reliably found within the maximum FEs for all benchmark problems except for function $g20$. The final solutions of all benchmark problems can be identified with an error of less than 0.0001 from the optimal solution within the maximum FEs except for functions $g20$ and $g22$. Most benchmark functions find the optimal solution with an error of less than 0.0001 before 50 000 FEs except for functions $g02$, $g17$, $g20$, $g22$, and $g23$. It can also be observed that the cultural CPSO has 100% feasible rate for all benchmark problems except for function $g20$, and 100% success rate for all benchmark problems except for functions $g02$, $g10$, $g17$, $g21$, and $g22$. However, we should note that for functions $g10$, $g17$, and $g21$, the success rates are fairly high at 96%, 92%, and 96%, respectively. A summary of the statistical results for the best, median, mean, worst, and standard deviations obtained by the cultural CPSO over 25 independent runs is shown in Table IV. As can be seen in this table, except for function $g20$, we have found feasible solutions for all other benchmark problems.

D. Convergence Graphs

For the median run of each test function with FEs of 500 000, two semilog graphs are plotted for each test function. The first graph is $\log_{10}[f(\mathbf{x}) - f(\widetilde{\mathbf{x}})]$ versus FEs, while $f(\widetilde{\mathbf{x}})$ is given in the rightmost column of Table I, and $f(\mathbf{x})$ is the objective value for the best solution at specific FEs. The second graph is $\log_{10}(\overline{v})$ versus FEs, where $\overline{v}$ is the average value of the constraint violations at specific FEs. For these two graphs, points which satisfy $f(\mathbf{x}) - f(\widetilde{\mathbf{x}}) \leq 0$ are not plotted, since the logarithm for zero or negative numbers cannot be computed. Figs. 5–8 show these two graphs for all 24 benchmark problems.

E. Complexity Analysis

In Table V, the algorithm's complexity corresponding to all 24 benchmark problems is shown. The computed times in seconds for complexity are $T1$, $T2$, and $(T2 - T1)/T1$, where $T1$ represents the average computing time of 10 000 evaluations for each test problem and $T2$ is the average of 10 000 evaluations for all benchmark problems [49].

Furthermore, in this section, the average time for 1000 FEs had been found for some test functions to examine the memory access and processing time and its relation with the problems. We have selected eight problems based on their FEs that they reach to the optimum with an error of less than 0.0001. From Table III, we can see that g08, g12, and g11 will reach to the optimum point fairly early at FEs of 2302, 3289, and 4589, respectively, showing that g08, g12, and g11 are fairly easier than the others in the set of 24 benchmark problems. On the other hand, g17, g23, and g02 will reach to the optimum point fairly late at FEs of 158 367, 62 091, and 56 392, respectively, showing that g17, g23, and g02 are harder than the others in the set of 24 benchmark problems. By choosing these six benchmark functions, we compare the average time for 1000 FEs. These values are summarized in Table VI. As it is shown, the processing time for 1000 FEs for the easier problems is, in general, less than that for the more difficult problems.

F. Performance Comparison

Furthermore, we have compared the performance of the cultural CPSO with nine state-of-the-art constrained optimization heuristics in terms of two performance indicators, namely, feasible rate and success rate. The selected high-performance algorithms are PSO [50], DMS-PSO [51], ε_DE [52], GDE

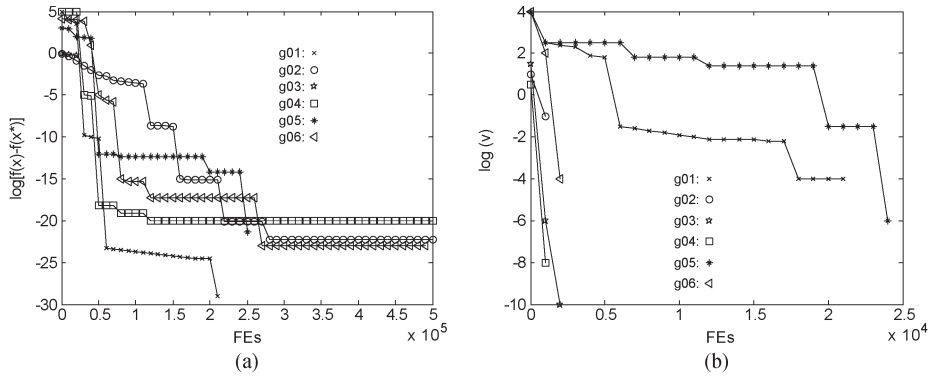

Fig. 5. Convergence graphs for problems g01–g06. (a) Function error values. (b) Mean constraint violations.

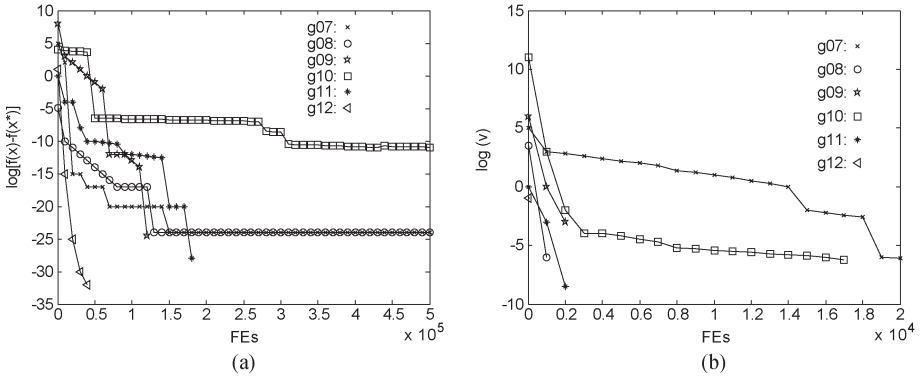

Fig. 6. Convergence graphs for problems g07–g12. (a) Function error values. (b) Mean constraint violations.

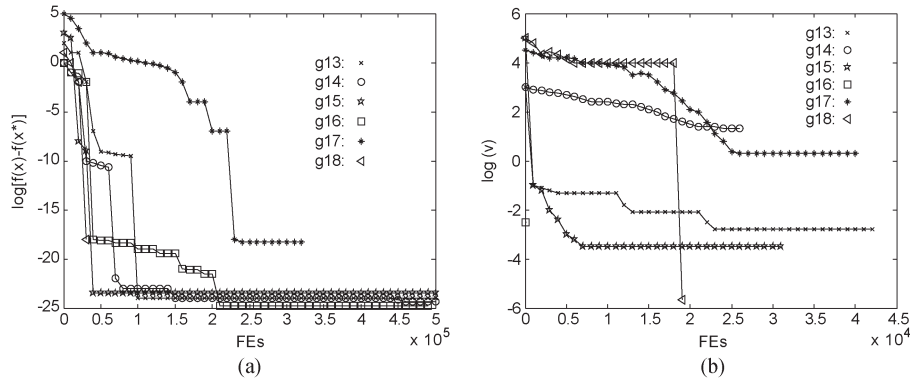

Fig. 7. Convergence graphs for problems g13–g18. (a) Function error values. (b) Mean constraint violations.

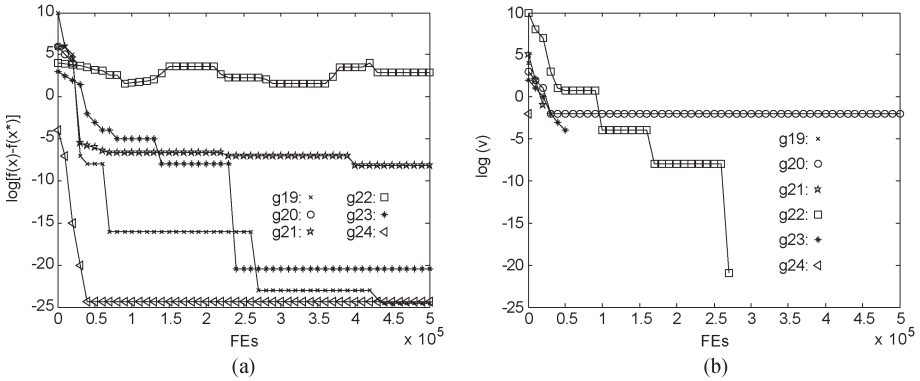

Fig. 8. Convergence graphs for problems g19–g24. (a) Function error values. (b) Mean constraint violations.

TABLE V
COMPUTATIONAL COMPLEXITY

$T1$	$T2$	$(T2-T1)/T1$
6.2351	11.3280	0.8168

TABLE VI
AVERAGE PROCESSING TIME (IN SECONDS) FOR DIFFERENT BENCHMARK TEST FUNCTIONS FOR 1000 EVALUATIONS

Pr.	g02	g08	g11	g12	g17	g23
FE optimum	56'392	2'302	4'589	3'289	158'367	62'091
Aver. Time	0.0092	0.0022	0.0039	0.0017	0.0162	0.0084

[53], jDE-2 [54], MDE [55], MPDE [56], PCX [57], PESO+ [58], and SaDE [59]. The comparative results are then demonstrated in Table VII. The average performance for each algorithm is also computed. This table demonstrates that the cultural CPSO has an average feasible rate of 95.83% on 24 benchmark problems, which places it as the top-performing algorithm along with DMS-PSO [51], ε_DE [52], and SaDE [59]. The results in the same table indicate that the proposed cultural CPSO has an average success rate of 90.00% on 24 benchmark problems, making it the third best performing algorithm after ε_DE [52] and PCX [57] with 91.67% and 90.17% success rates, respectively. ε_DE [52] is a differential algorithm adopted to solve constraint problems. The differential algorithm is a stable population-based search algorithm that is robust to multimodal problems. The ε_DE improves the behavior of a regular differential algorithm by adopting a gradient-based mutation that finds feasible points using the gradient of constraints at an infeasible point [52]. As a result, the ε_DE is suitable for multimodal problems. PCX [57] is a population-based algorithm generator that includes systematic plans such as selection, generation, replacement, and update plans. Each essential feature of an optimization task is independently designed via these four plans which are suitable for optimization problems. The generation plan in PCX uses a planar version of parent-centric recombination operator to produce an offspring [57].

G. Sensitivity Analysis

In this subsection, the sensitivity of algorithm performance with respect to some parameters is briefly assessed. The parameters to be tuned in the proposed algorithm are personal acceleration (c_1), swarm acceleration (c_2), global acceleration (c_3), and the rate for information exchange among swarms (r). Notice that the allowance number of particles to migrate ($N_{\text{migration}}$) is a fraction of the population size and does not need to be tuned. The tolerance for equality constraints is considered a fixed number of 0.0001 to be able to fairly compare our results with those of other algorithms. The flight momentum is also randomly selected from a uniform distribution and does not have a tuning issue, and the maximum velocity of particles in a specific dimension depends on the particle's positional range, which will consequently not be adjusted either.

We have applied sensitivity analysis to a selected set of benchmark problems by varying one parameter at a time while the other parameters are set as values discussed in the earlier sections. Table VIII shows the results of the sensitivity analysis. For every set of parameters, 25 independent runs are performed. We have recorded the mean statistical results for feasible solutions, feasible rate, and success rate, as defined earlier, for every set of parameters. This table shows that the effect of varying the acceleration on the algorithm's performance is by some extent problem dependent. This makes it difficult to identify the optimum parameters in order to achieve the best performance. We suggest the further analysis of this issue and the implementation of an adaptive dynamic law based upon the need for exploration or exploitation in the $\mathbf{f}-\mathrm{V}$ space discussed in the spatial knowledge of the belief space. This approach is similar to the one introduced in [38]. This table also shows that by increasing the rate for information exchange, the success rate will be greatly improved for all selected benchmark problems. On the other hand, by decreasing this rate, the success rate gets deteriorated.

H. Real-World Application Problem

In this subsection, the proposed algorithm is applied on a real-world application problem. Spring design is a mechanical design problem [60] to minimize the weight of a tension/compression spring, as shown in Fig. 9. There are NI constraints on minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. The design variables are mean coil diameter (x_1), wire diameter (x_2), and the number of active coils (x_3), along with four inequality constraints. The mathematical formulation of the problem is as follows:

$$\text{Minimize}: \quad f(x) = (x_3+2)x_1x_2^2 \tag{20}$$

$$\begin{aligned}\text{Subject to}: \quad & 1-\frac{x_1^3x_3}{71785x_2^4}\le 0\\ & \frac{4x_1^2-x_1x_2}{12566\,(x_1x_2^3-x_1^3x_3)}-\frac{1}{5108x_2^2}-1\le 0\\ & 1-\frac{140.45x_2}{x_1^3x_3}\le 0\\ & \frac{x_1+x_2}{1.5}-1\le 0\end{aligned} \tag{21}$$

with the following limits on variables: $0.25 \le x_1 \le 1.3, 0.05 \le x_2 \le 2.0$, and $2 \le x_3 \le 15$.

Table IX demonstrates the simulation results for the spring design problem using the proposed cultural PSO. The setting of the algorithm is considered as it has been discussed in Section IV-A. These results are computed after 30 independent runs have been performed. The decision variable values and optimized solution are summarized in the table. As can be observed from the table, the proposed algorithm finds the better optimum solution compared to the other algorithms.

V. CONCLUSION

In this paper, we have proposed the cultural CPSO, a novel heuristic to solve constrained optimization problems, which incorporates the information of objective function and constraint violation, to construct a cultural framework consisting

TABLE VII
COMPARISON OF CULTURAL CPSO WITH STATE-OF-THE-ART METHODS IN TERMS OF FEASIBLE RATE AND SUCCESS RATE

Prob.	PSO [51]	DMS-PSO[52]	ε_DE [53]	GDE [54]	jDE-2 [55]	MDE [56]	MPDE [57]	PCX [58]	PESO+ [59]	SaDE [60]	Cultural CPSO
g01	100,72	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g02	100,0	100,84	100,100	100,72	100,92	100,16	100,92	100,64	100,56	100,84	100,76
g03	100,0	100,100	100,100	96,4	100,0	100,100	100,84	100,100	100,100	100,96	100,100
g04	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g05	100,24	100,100	100,100	96,92	100,68	100,100	100,100	100,100	100,100	100,100	100,100
g06	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g07	100,72	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,96	100,100	100,100
g08	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g09	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g10	100,8	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,16	100,100	100,96
g11	100,100	100,100	100,100	100,100	100,96	100,100	100,96	100,100	100,100	100,100	100,100
g12	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g13	100,0	100,100	100,100	88,40	100,0	100,100	88,48	100,100	100,100	100,100	100,100
g14	100,0	100,100	100,100	100,96	100,100	100,100	100,100	100,100	100,0	100,80	100,100
g15	100,84	100,100	100,100	100,96	100,96	100,100	100,100	100,100	100,100	100,100	100,100
g16	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
g17	100,0	100,0	100,100	76,16	100,4	100,100	96,28	100,100	100,0	100,4	100,92
g18	100,100	100,100	100,100	84,76	100,100	100,100	100,100	100,100	100,92	100,92	100,100
g19	100,12	100,100	100,100	100,88	100,100	100,0	100,100	100,100	100,0	100,100	100,100
g20	0,0	0,0	0,0	0,0	4,0	0,0	0,0	0,0	0,0	0,0	0,0
g21	8,0	100,100	100,100	88,60	100,92	100,100	100,68	100,100	100,0	100,60	100,96
g22	0,0	100,0	100,0	0,0	0,0	0,0	0,0	0,0	0,0	100,0	100,0
g23	100,0	100,100	100,100	88,40	100,92	100,100	100,100	100,100	96,0	100,88	100,100
g24	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100	100,100
Average	87.83, 48.83	95.83, 86.83	95.83, 91.67	88.17, 74.17	91.83, 76.67	91.67, 84.00	91.00, 84.00	91.67, 90.17	91.50, 65.00	95.83, 83.50	95.83, 90.00

TABLE VIII
SENSITIVITY ANALYSIS WITH RESPECT TO PERSONAL ACCELERATION (c_1), SWARM ACCELERATION (c_2), GLOBAL ACCELERATION (c_3), AND RATE OF INFORMATION EXCHANGE (R). THE MEAN RESULTS OF FEASIBLE SOLUTIONS, FEASIBLE RATE, AND SUCCESS RATE ARE COMPUTED OVER 25 INDEPENDENT RUNS

c_1 / Prob.	Mean results of feasible solutions, Feasible Rate%, Success Rate%			
	1.0	1.5	2.0	2.5
g03	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%
g10	7049.248020570674,100%,96%	7049.248020573664,100%,96%	7049.248020573941,100%,100%	7049.248020570062,100%,96%
g14	-47.7648884595,100%,100%	-47.7648884595,100%,100%	-47.7648884595,100%,100%	-47.7648884595,100%,92%
g18	-0.8660254038,100%,100%	-0.8660254038,100%,100%	-0.8660254038,100%,96%	-0.8660254038,100%,100%
g21	193.7245128803,100%,96%	193.7245126309,100%,96%	193.7245121603,100%,100%	193.7245139367,100%,92%
c_2	1.0	1.5	2.0	2.5
g03	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%
g10	7049.248020574453,100%,100%	7049.248020573664,100%,96%	7049.248020579940,100%,96%	7049.248020573296,100%,96%
g14	-47.7648884595,100%,100%	-47.7648884595,100%,100%	-47.7648884595,100%,100%	-47.7648884595,100%,100%
g18	-0.8660254038,100%,96%	-0.8660254038,100%,100%	-0.8660254038,100%,100%	-0.8660254038,100%,96%
g21	193.7245126006,100%,100%	193.7245126309,100%,96%	193.7245124569,100%,100%	193.7245124389,100%,96%
c_3	1.0	1.5	2.0	2.5
g03	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%
g10	7049.248020573377,100%,96%	7049.248020573664,100%,96%	7049.248020584087,100%,100%	7049.248020593467,100%,96%
g14	-47.7648884595,100%,100%	-47.7648884595,100%,100%	-47.7648884595,100%,92%	-47.7648884595,100%,100%
g18	-0.8660254038,100%,96%	-0.8660254038,100%,100%	-0.8660254038,100%,96%	-0.8660254038,100%,100%
g21	193.7245146753,100%,96%	193.7245126309,100%,96%	193.7245128903,100%,92%	193.7245136098,100%,100%
r	10%	20%	30%	40%
g03	-1.0005001000,100%,92%	-1.0005001000,100%,100%	-1.0005001000,100%,100%	-1.0005001000,100%,100%
g10	7049.248020692614,100%,92%	7049.248020579157,100%,96%	7049.248020573664,100%,96%	7049.248020550004,100%,100%
g14	-47.7648884586,100%,96%	-47.7648884595,100%,100%	-47.7648884595,100%,100%	-47.7648884595,100%,100%
g18	-0.8660254017,100%,96%	-0.8660254038,100%,100%	-0.8660254038,100%,100%	-0.8660254038,100%,100%
g21	193.7245268306,100%,92%	193.7245138506,100%,96%	193.7245126309,100%,96%	193.7245110215,100%,100%

of two sections: a multiple-swarm PSO with the ability of inter-swarm communication as population space and a belief space including four elements, namely, normative knowledge, spatial knowledge, situational knowledge, and temporal knowledge. Each swarm assembles two lists of particles to share with other swarms based upon the cultural information retrieved from

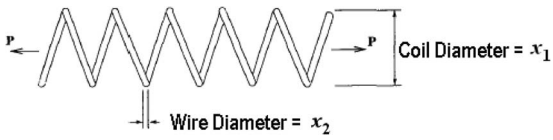

Fig. 9. Schema for the spring design problem.

TABLE IX
BEST SOLUTION FOR THE SPRING DESIGN APPLICATION PROBLEM

Algorithms	x_1	x_2	x_3	$f(x)$
Proposed Alg.	0.351551	0.05048	11.6321	0.012212
Arora [62]	0.399180	0.053396	9.815400	0.013447
Ray and Saini [63]	0.321532	0.050417	13.979915	0.013060

different sections of the belief space. This cultural-information-based communication facilitates the algorithm's performance on better handling the constraints along with optimizing the objective function simultaneously. In the proposed algorithm, the feedbacklike communication channels are to connect the population and belief spaces to increase the efficiency of the PSO mechanism in the search process. In system science, a negative feedback loop is used to stabilize the system and to tune the output to the desired values, and results in a better behavior of the closed-loop system compared to that of the open-loop one. If we look at the problem of optimization as the output of a system that needs to be optimized and if we consider the population of particles and its associated knowledge as an overall system, the behavior of this closed-loop system should have been improved by applying the feedback-type communication channels compared to that of the open-loop one, i.e., only population space.

The cultural CPSO shows competitive results when performing extensive experiments on 24 benchmark test functions. The comparison study with the chosen state-of-the-art constrained optimization techniques indicates that the cultural CPSO is able to perform competitively in terms of commonly used performance metrics, namely, feasible rate and success rate. Furthermore, sensitivity analysis was performed on the parameters of the paradigm, which shows that by increasing the rate of information exchange, the success rate is greatly improved. We have also applied our proposed algorithm on a real-world optimization problem, which demonstrates better performance compared to the other algorithms applied on the problem. As future work, the proposed framework for single-objective optimization will be extended into a cultural-based multiobjective PSO (Cultural-MOPSO) [63]–[65] to exploit its robust performance under a dynamic environment when fitness landscape and constraints will change periodically or sporadically. Additionally, applications of Cultural-MOPSO to medical diagnosis [66] and structural control [67] will be exploited.

REFERENCES

[1] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 9, no. 4, pp. 424–435, Aug. 2005.

[2] Y. Wang, Y. C. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 221–232, May 2005.

[3] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 1, pp. 122–128, Jan. 1986.

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. Int. Joint Conf. Neural Netw.*, Perth, Australia, 1995, pp. 1942–1948.

[5] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[6] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[7] N. R. Samal, A. Konar, S. Das, and A. Abraham, "A closed loop stability analysis and parameter selection of the particle swarm optimization dynamics for faster convergence," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 1769–1776.

[8] S. Bhattacharya, A. Konar, and A. Nagar, "A Lyapunov-based extension to PSO dynamics for continuous function optimization," in *Proc. Eur. Symp. Comput. Model. Simul.*, Liverpool, U.K., 2008, pp. 28–33.

[9] N. R. Samal, A. Konar, and A. Nagar, "Stability analysis and parameter selection of a particle swarm optimizer in a dynamic environment," in *Proc. Eur. Symp. Comput. Model. Simul.*, Liverpool, U.K., 2008, pp. 21–27.

[10] A. Ratnaweera, S. K. Halganuge, and H. C. Watson, "Self organizing hierarchical particle swarm optimizer with time-varying acceleration co-efficient," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[11] X. Hu and R. C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proc. World Multiconf. Syst., Cybern. Informat.*, Orlando, FL, 2002, pp. 122–127.

[12] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Intelligent Technologies-Theory and Application: New Trends in Intelligent Technologies*, vol. 76, P. Sincak, J. Vascak, V. Kvasnicka, and J. Pospichal, Eds. Amsterdam, The Netherlands: IOS Press, 2002, pp. 214–220.

[13] G. Coath and S. K. Halgamuge, "A comparison of constraint handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Canberra, Australia, 2003, pp. 2419–2425.

[14] U. Paquet and A. P. Engelbrecht, "A new particle swarm optimizer for linearly constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Canberra, Australia, 2003, pp. 227–233.

[15] T. Takahama and S. Sakai, "Solving constrained optimization problems by the constrained particle swarm optimizer with adaptive velocity limit control," in *Proc. IEEE Int. Conf. Cybern. Intell. Syst.*, Bangkok, Thailand, 2006, pp. 1–7.

[16] R. A. Krohling and L. S. Coelho, "Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1407–1416, Dec. 2006.

[17] B. Yang, Y. Chen, Z. Zhao, and Q. Han, "A master–slave particle swarm optimization algorithm for solving constrained optimization problems," in *Proc. World Congr. Intell. Control Autom.*, Dalian, China, 2006, pp. 3208–3212.

[18] J. Zheng, Q. Wu, and W. Song, "An improved particle swarm algorithm for solving nonlinear constrained optimization problems," in *Proc. IEEE Int. Conf. Natural Comput.*, Haikou, China, 2007, pp. 112–117.

[19] A. Y. Saber, S. Ahmmed, A. Alshareef, A. Abdulwhab, and K. Adbullah-Al-Mamun, "Constrained non-linear optimization by modified particle swarm optimization," in *Proc. Int. Conf. Comput. Inf. Technol.*, Gyeongju, South Korea, 2008, pp. 1–7.

[20] J. Li, Z. Liu, and C. Peng, "Solving constrained optimization via dual particle swarm optimization with stochastic ranking," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, Wuhan, China, 2008, pp. 1215–1218.

[21] J. I. Flores-Mendoza and E. Mezura-Montes, "Dynamic adaptation and multiobjective concepts in a particle swarm optimizer for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 3427–3434.

[22] T. O. Ting, K. P. Wong, and C. Y. Chung, "Hybrid constrained genetic algorithm/particle swarm optimization load flow algorithm," *IET Gener., Transm., Distrib.*, vol. 2, no. 6, pp. 800–812, Nov. 2008.

[23] Z. Liu, C. Wang, and J. Li, "Solving constrained optimization via a modified genetic particle swarm optimization," in *Proc. Int. Workshop Forensic Appl. Tech. Telecommun., Inf., Multimedia*, Adelaide, Australia, 2008, pp. 217–220.

[24] G. G. Yen and W. Leong, "Constraint handling in particle swarm optimization," *Int. J. Swarm Intell. Res.*, vol. 1, no. 1, pp. 42–63, 2010.

[25] R. G. Reynolds, "An introduction to cultural algorithms," in *Proc. Annu. Conf. Evol. Program.*, River Edge, NJ, 1994, pp. 131–139.
[26] R. G. Reynolds, "Cultural algorithms: Theory and applications," in *Advanced Topics in Computer Science Series: New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. New York: McGraw- Hill, 1999, pp. 367–377.
[27] R. G. Reynolds and W. Sverdlik, "Problem solving using cultural algorithms," in *Proc. IEEE Congr. Evol. Comput.*, Orlando, FL, 1994, pp. 645–650.
[28] C. J. Chung and R. G. Reynolds, "A testbed for solving optimization problems using cultural algorithms," in *Proc. Annu. Conf. Evol. Programm.*, San Diego, CA, 1996, pp. 225–236.
[29] C. J. Chung and R. G. Reynolds, "Function optimization using evolutionary programming with self-adaptive cultural algorithms," in *Proc. Asia-Pacific Conf. Simul. Evol. Learn.*, Taejon, South Korea, 1996, pp. 17–26.
[30] Z. Kobti, R. G. Reynolds, and T. Kohler, "A multi-agent simulation using cultural algorithms: The effect of culture on the resilience of social systems," in *Proc. IEEE Congr. Evol. Comput.*, Canberra, Australia, 2003, pp. 1988–1995.
[31] R. G. Reynolds, B. Peng, and J. Brewster, "Cultural swarms II: Virtual algorithm emergence," in *Proc. IEEE Congr. Evol. Comput.*, Canberra, Australia, 2003, pp. 1972–1979.
[32] R. Iacoban, R. G. Reynolds, and J. Brewster, "Cultural swarms: Modeling the impact of culture on social interaction and problem solving," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, 2003, pp. 205–211.
[33] B. Peng, R. G. Reynolds, and J. Brewster, "Cultural swarms," in *Proc. IEEE Congr. Evol. Comput.*, Canberra, Australia, 2003, pp. 1965–1971.
[34] B. Peng and R. G. Reynolds, "Cultural algorithms: Knowledge learning in dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, Portland, OR, 2004, pp. 1751–1758.
[35] R. L. Becerra and C. A. C. Coello, "Culturizing differential evolution for constrained optimization," in *Proc. Mexican Int. Conf. Comput. Sci.*, Colima, Mexico, 2004, pp. 304–311.
[36] R. L. Becerra and C. A. C. Coello, "Optimization with constraints using a cultured differential evolution approach," in *Proc. Genetic Evol. Comput. Conf.*, Washington, DC, 2005, pp. 27–34.
[37] R. G. Reynolds and B. Peng, "Cultural algorithms: Computational modeling of how cultures learn to solve problems: An engineering example," *Cybern. Syst.*, vol. 36, no. 8, pp. 753–771, 2005.
[38] M. Daneshyari and G. G. Yen, "Cultural-based multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 553–567, Apr. 2011.
[39] X. Jin and R. G. Reynolds, "Using knowledge-based system with hierarchical architecture to guide the search of evolutionary computation," in *Proc. IEEE Int. Conf. Tools Artif. Intell.*, Chicago, IL, 1999, pp. 29–36.
[40] X. Yuan, A. Su, Y. Yuan, X. Zhang, B. Cao, and B. Yang, "A chaotic hybrid cultural algorithm for constrained optimization," in *Proc. IEEE Int. Conf. Genetic Evol. Comput.*, Jingzhou, China, 2008, pp. 307–310.
[41] W. Tang and Y. Li, "Constrained optimization using triple spaces cultured genetic algorithm," in *Proc. IEEE Int. Conf. Nat. Comput.*, Jinan, China, 2008, pp. 589–593.
[42] Z. Zhao and H. Gao, "FIR digital filters based on cultural particle swarm optimization," in *Proc. Int. Workshop Inf. Secur. Appl.*, Qingdao, China, 2009, pp. 252–255.
[43] L. A. Sjaastad, "The costs and returns of human migration," *J. Political Econ.*, vol. 70, no. 5, pp. 80–93, Oct. 1962.
[44] G. Bottomley, *From Another Place: Migration and the Politics of Culture*. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[45] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, Jun. 2009.
[46] G. G. Yen and M. Daneshyari, "Diversity-based information exchange among multiple swarms in particle swarm optimization," *Int. J. Comput. Intell. Appl.*, vol. 7, no. 1, pp. 57–75, 2008.
[47] G. G. Yen and W. F. Leong, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 4, pp. 890–911, Jul. 2009.
[48] B. Tessema and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 3, pp. 565–578, May 2009.
[49] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC2006," Nanyang Technological Univ., Singapore, Tech. Rep., 2006, Special Session on Constrained Real-Parameter Optimization.
[50] K. Zielinski and R. Laur, "Constrained single-objective optimization using particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 443–450.
[51] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 9–16.
[52] T. Takahama and S. Sakai, "Constrained optimization by the constrained differential evolution with gradient based mutation and feasible elites," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 1–8.
[53] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 207–214.
[54] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 215–222.
[55] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. C. Coello, "Modified differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 25–32.
[56] M. F. Tasgetiren and P. N. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 33–40.
[57] A. Sinha, A. Srinivasan, and K. Deb, "A population-based parent centric procedure for constrained real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 239–245.
[58] A. E. Munoz-Zavala, A. Hernandez-Aguirre, E. R. Villa-Diharce, and S. Botello-Rionda, "PESO+ for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 231–238.
[59] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 17–24.
[60] M. Daneshyari and G. G. Yen, "Talent-based social algorithm for optimization," in *Proc. IEEE Congr. Evol. Comput.*, Portland, OR, 2005, pp. 786–791.
[61] J. S. Arora, *Introduction to Optimum Design*. New York: McGraw-Hill, 1989.
[62] T. Ray and P. Saini, "Engineering design optimization using a swarm with an intelligent information sharing among individuals," *Eng. Optim.*, vol. 33, no. 6, pp. 735–748, 2001.
[63] H. Lu and G. G. Yen, "Rank-density-based multiobjective genetic algorithm and benchmark test function study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 325–343, Aug. 2003.
[64] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multi-objective evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 514–525, Jun. 2009.
[65] W. F. Leong and G. G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1270–1293, Oct. 2008.
[66] P. Meesad and G. G. Yen, "Combined numerical and linguistic knowledge representation and its application to medical diagnosis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 2, pp. 206–222, Mar. 2003.
[67] L. Davis, D. Hyland, and G. G. Yen, "Adaptive neural control for space vibration suppression," *Smart Mater. Struct.*, vol. 8, no. 6, pp. 753–766, Dec. 1999.

Moayed Daneshyari (S'96–M'08) received the B.S. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1995 and the M.S. degree in physics and the Ph.D. degree in electrical and computer engineering from Oklahoma State University, Stillwater, in 2007 and 2010, respectively.

He has been an Assistant Professor with the Department of Technology, Elizabeth City State University, Elizabeth City, NC.

Gary G. Yen (S'87–M'88–SM'97–F'09) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, in 1992.

He is currently a Professor with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater.