Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2011, Article ID 980216, 19 pages
doi:10.1155/2011/980216

Research Article

A Probability Collectives Approach with a Feasibility-Based Rule for Constrained Optimization

Anand J. Kulkarni and K. Tai

School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798

Correspondence should be addressed to K. Tai, mktai@ntu.edu.sg

Received 6 May 2011; Accepted 6 September 2011

Academic Editor: R. Saravanan

This paper demonstrates an attempt to incorporate a simple and generic constraint handling technique to the Probability Collectives (PC) approach for solving constrained optimization problems. The approach of PC optimizes any complex system by decomposing it into smaller subsystems and further treats them in a distributed and decentralized way. These subsystems can be viewed as a Multi-Agent System with rational and self-interested agents optimizing their local goals. However, as there is no inherent constraint handling capability in the PC approach, a real challenge is to take into account constraints and at the same time make the agents work collectively avoiding the tragedy of commons to optimize the global/system objective. At the core of the PC optimization methodology are the concepts of Deterministic Annealing in Statistical Physics, Game Theory and Nash Equilibrium. Moreover, a rule-based procedure is incorporated to handle solutions based on the number of constraints violated and drive the convergence towards feasibility. Two specially developed cases of the Circle Packing Problem with known solutions are solved and the true optimum results are obtained at reasonable computational costs. The proposed algorithm is shown to be sufficiently robust, and strengths and weaknesses of the methodology are also discussed.

1. Introduction

The growing complexity and uncertainty of the problem domain motivated some researchers to resort to a distributed and decentralized optimization approach [1–5]. Such an approach decomposes an entire system into smaller subsystems for individual optimization to reach the system level optimum. These subsystems together can be viewed as a collective, which in other words is a group of learning agents or a multiagent system (MAS). In a distributed MAS, the rational and self-interested behavior of the agents is very important to achieve the best possible local goal/reward/payoff, but it is not trivial to make such agents work collectively to achieve the best possible global or system objective. Moreover, in the context of distributed MAS, the concept of "tragedy of commons" certainly becomes important and requires special attention. It refers to the situation in which the rational and self-interested independent individuals deplete the shared limited resource in a greedy way, even if it is well understood that it may not be beneficial for long-term interest collectively for all; that is, an individual may receive a benefit but on the other hand the loss will be shared among all [6]. In a distributed MAS, if every rational and self-interested agent tries to achieve the individual goal in a greedy way, it may lead to poor system performance [7]. On the other hand, in order to achieve the true global optimum, not every individual agent can receive the best it could have. To achieve the best system objective in a distributed MAS, the tragedy of commons should be avoided. In addition, similar to conventional (centralized) optimization approaches, the problem becomes harder when constraints are involved and thus constraint handling remains a key issue to be addressed [8–10].

An emerging artificial intelligence tool in the framework of collective intelligence (COIN) for modeling and controlling distributed MAS referred to as probability collectives (PC) was first proposed by Dr. David Wolpert in 1999 in a technical report presented to NASA [11]. It is inspired from a sociophysics viewpoint with deep connections to game

theory, statistical physics, and optimization [2, 12]. From another viewpoint, the method of PC theory is an efficient way of sampling the joint probability space, converting the problem into the convex space of probability distribution. PC considers the variables in the system as individual agents/players in a game being played iteratively [3, 13, 14]. Unlike stochastic approaches such as genetic algorithm (GA), swarm optimization or simulated annealing (SA), rather than deciding on the agent's moves/set of actions, PC allocates the probability values for selecting each of the agent's moves. At each iteration, every agent independently updates its own probability distribution over a strategy set which is the set of moves/actions affecting its local goal which in turn also affects the global or system objective [2]. The process continues and reaches equilibrium when no further increase in reward is possible for the individual agent by changing its actions further. This equilibrium concept is referred to as Nash equilibrium [15]. The concept is successfully formalized and implemented through the PC methodology. The approach works on probability distributions, directly incorporating uncertainty, and is based on the prior knowledge of actions/strategies of all the other agents. The approach of PC has been implemented for solving both unconstrained [4, 5, 7, 16–24] as well as constrained [1, 13, 14, 25–28] optimization problems. The associated literature is discussed in the following few paragraphs.

It was demonstrated in [29] optimizing Schaffer's function that the search process in PC is more robust/reproducible as compared to GA. In addition, PC also outperformed GA in the rate of descent, trapping in false minima and long-term optimization when tested and compared for the multimodality, nonlinearity, and nonseparability in solving other benchmark problems such as Schaffer's function, Rosenbrock function Ackley Path function, and Michalewicz epistatic function. Some of the fundamental differences between GA and PC were also discussed in [16]. At the core of the GA optimization algorithm is the population of solutions. In every iteration, each individual solution from the population is tested for its fitness to the problem at hand [16] and the population is updated accordingly. GA plots the best-so-far curve showing the fitness of the best individual in the last preset generations. In PC, on the other hand, the probability distribution of the possible solutions is updated iteratively. After a predefined number of iterations, the probability distribution of the available strategies across the variable space is plotted in PC optimizing an associated homotopy function. It also directly incorporates uncertainty due to both imperfect sampling and the stochastic independence of agents' actions [16]. The above comparison with GA indicated that PC can potentially be applied to wide application areas.

The superiority of the decentralized PC architecture over a centralized one was underlined in [7] solving the 8-queens problem. Both approaches differ from each other because of the distributed sample generation and updating of the probabilities in the former approach. In addition, PC was also compared with the backtracking algorithm referred to as asynchronous distributed optimization (ADOPT) [30]. Although the ADOPT algorithm is a distributed approach, the communication and computational load was not equally distributed among the agents. It was also demonstrated that although ADOPT was guaranteed to find the solution in each run, communication and computations required were more than for the same problem solved using PC.

The approach of PC was successfully applied solving the complex combinatorial optimization problem of airplane fleet assignment having the goal of minimization of the number of flights with 129 variables and 184 constraints. Applying a centralized approach to this problem may increase the communication and computational load. Furthermore, it may add latency in the system and result in the growing possibility of conflict in schedules and continuity. Using PC, the goal was collectively achieved exploiting the advantages of a distributed and decentralized approach by the airplanes selecting their own schedules depending upon the individual payoffs for the possible routes [13]. The approach of PC was also successfully applied solving combinatorial optimization problems such as the joint optimization of the routing and resource allocation in wireless networks [17–23].

Two different PC approaches were proposed in [25] avoiding airplanes collision. In the first approach, every airplane was assumed to be an autonomous agent. These agents selected their individual paths and avoided collision with other airplanes traveling in the neighborhood. In order to implement this approach, a complex negotiation mechanism was required for the airplanes to communicate and cooperate with one another. In the semicentralized approach, every airplane was given a chance to become a host airplane which computed and distributed the solution to all other airplanes. It is important to mention that the host airplane computed the solution based on the independent solution shared by previous host airplane. This process continued in a sequence until all the airplanes selected their individual paths. Both approaches were validated solving an interesting airplane conflict problem in which the airplanes were equidistantly arranged on the periphery of a circle. The targets of the individual airplanes were set as the opposite points on the periphery of the circle setting the center point of the circle as a point of conflict. In both approaches, the collision avoidance constraints were incorporated using a penalty function approach.

A variation of the original PC approach in [3, 12–14] referred to as sequentially updated PC (SPC) was proposed in [24]. The variation was achieved by changing the sampling criterion and the method for estimating the sampling space in every iteration. The SPC was tested by optimizing the unconstrained Hartman's functions and the vehicle target assignment type of game. The SPC performed better with higher dimension Hartman's functions only but failed to converge in the target assignment game.

The sampling method as well as associated sampling space updating scheme of the original PC approach was modified by the authors of this paper. The modified PC approach was validated by successfully optimizing the Rosenbrock function [5]. It was also applied for solving two test cases of the NP-hard combinatorial problem of Multidepot multiple travelling salesmen problem (MDMTSP) [1] as well as the cases of single-depot MTSP (SDMTSP) [26]. In solving the MDMTSP and SDMTSP, in order to handle constraints,

several heuristic techniques were successfully incorporated. In addition, a constrained PC approach using penalty function method successfully solving three test problems was proposed in [27].

The potential of PC in mechanical design was demonstrated for optimizing the cross-sections of individual bars and segments of a 10 bar truss [14] and a segmented beam [4], respectively. The 10 bar truss problem in [14] was solved as a discrete constrained problem while the segmented beam problem in [4] was solved as a continuous unconstrained problem. In [14], the solution was feasible but was worse than those obtained by other methods [31–33]. The approach of PC [13, 14] was also tested on the discrete constrained problem of university course scheduling [28], but the implementation failed to generate any feasible solution.

The above discussion shows that PC is versatile and applicable to variegated areas including constrained optimization problems such as fleet assignment [13], ten bar truss problem [14], MDMTSP [1], SDMTSP [26], university course scheduling [28], and so forth. It is important to note that in [13, 14, 28] the approach of incorporating constraints into PC algorithms was not explicitly mentioned and demonstrated. Furthermore, in [1, 26], a repair approach pushing the solution into the feasible region was implemented. Such an approach may not be usable for handling generic constraints. If complexity of the problem and related constraints increase, the repair work may become more tedious and may add further computational load, limiting the use of the repair approach to smaller-size problems with fewer constraints. In addition, although a constrained PC approach was implemented in [25] as well as by the authors of this paper in [27] using penalty function method, it was noticed that the approach was sensitive to the choice of penalty parameter and its updating scheme, which required several associated preliminary trials.

This paper demonstrates an attempt to develop a generic constraint handling technique for PC in order to make it an even more versatile optimization algorithm. A variation of the feasibility-based rule originally proposed in [34] and further implemented in [35–40] was employed solving two cases of the circle packing problem (CPP). Furthermore, similar to [34–40] where additional techniques were implemented to avoid premature convergence, a perturbation approach was incorporated. In addition, attaining the true optimum solution to CPP using PC clearly demonstrated its ability to avoid the tragedy of commons.

The remainder of this paper is organized as follows. Section 2 discusses various prominent characteristics of the PC method highlighting its competence over other algorithms optimizing collectives. The framework and detailed formulation of the constrained PC method is presented in Section 3. It includes the formulation of homotopy function, constraint handling technique using the feasibility-based rule, and the concept of Nash equilibrium. In Section 4, the validation of the constrained PC approach is shown by solving two test cases of the CPP. It also includes the associated problem specific heuristic technique. The evident features, advantages, and some limitations of the constrained PC approach are discussed in Section 5. Finally, the concluding remarks along with the future directions are presented in Section 6. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) scheme minimizing the homotopy function is discussed in the appendix provided at the end of the paper.

2. Characteristics of PC

The PC approach has the following key characteristics that make it a competitive choice over other algorithms for optimizing collectives.

(1) PC is a distributed solution approach in which each agent independently updates its probability distribution at any time instance and can be applied to continuous, discrete, or mixed variables, and so forth, [2, 11, 13]. Since the probability distribution of the strategy set is always a vector of real numbers regardless of the type of variable under consideration, conventional techniques of optimization for Euclidean vectors, such as gradient descent, can be exploited.

(2) It is robust in the sense that the cost function (global/system objective) can be irregular or noisy; that is, it can accommodate noisy and poorly modeled problems [2, 7].

(3) The failed agent can just be considered as one that does not update its probability distribution, without affecting the other agents. On the other hand, it may severely hamper the performance of other techniques [7].

(4) It provides the sensitivity information about the problem in the sense that a variable with a peaky distribution (having highest probability value) is more important in the solution than a variable with a broad distribution; that is, peaky distribution provides the best choice of action that can optimize the global goal [2].

(5) The formation of the homotopy function for each agent (variable) helps the algorithm to jump out of the possible local minima and further reach the global minima [13, 26].

(6) It can successfully avoid the tragedy of commons, skipping the local minima and further reach the true global minima [7].

(7) The computational and communication load is marginally less and equally distributed among all the agents [7].

(8) It can efficiently handle problems with a large number of variables [13].

With PC solving optimization problems as a MAS, it is worth discussing some of its characteristics to compare the similarities and differences with multiagent reinforcement learning (MARL) methods. Most MARL methods such as fully cooperative, fully competitive, and mixed (neither cooper-ative nor competitive) are based on game theory, optimization and evolutionary computation [41]. According to

[41], most of these types of methods possess less scalability and are sensitive to imperfect observations. Any uncertainty or incomplete information may lead to unexpected behavior of the agents. However, the scalability of the fully cooperative methods such as coordination-free methods can be enhanced by explicitly using the communication and/or uncertainty techniques [41–44]. On the other hand, PC is scalable and can handle uncertainty in terms of probability. Moreover, the random strategies selected by any agent can be coordinated or negotiated with the other agents based on the social conventions, right to communication, and so forth. This social aspect makes PC a cooperative approach. Furthermore, indirect coordination-based methods work on the concept of biasing the selection towards the likelihood of the good strategies. This concept is similar to the one used in the PC algorithm presented here, in which agents choose the strategy sets only in the neighborhood of the best strategy identified in the previous iteration. In the case of mixed MARL algorithms, the agents have no constraints imposed on their rewards. It is similar to the PC algorithm in which the agents respond or select the strategies and exhibit self-interested behavior. However, the mixed MARL algorithms may encounter multiple Nash equilibria while in PC a unique Nash equilibrium can be achieved.

3. Conceptual Framework of Constrained PC

PC treats the variables in an optimization problem as individual self-interested learning agents/players of a game being played iteratively [13]. While working in some definite direction, these agents select actions over a particular interval and receive some local rewards on the basis of the system objective achieved because of those actions. In other words, these agents optimize their local rewards or payoffs, which also optimize the system level performance. The process iterates and reaches equilibrium (referred to as Nash equilibrium) when no further increase in the reward is possible for the individual agent through changing its actions further. Moreover, the method of PC theory is an efficient way of sampling the joint probability space, converting the problem into the convex space of probability distribution. PC allocates probability values to each agent's moves and hence directly incorporates uncertainty. This is based on prior knowledge of the recent action or behavior selected by all other agents. In short, the agents in the PC framework need to have knowledge of the environment along with every other agent's recent action or behavior.

In every iteration, each agent randomly samples from within its own strategy set as well as from within other agents' strategy sets and computes the corresponding system objectives. The other agents' strategy sets are modeled by each agent based on their recent actions or behavior only, that is, based on partial knowledge. By minimizing the collection of system objectives, every agent identifies the possible strategy which contributes the most towards the minimization of the collection of system objectives. Such a collection of functions is computationally expensive to minimize and also may lead to local minima [3]. In order to avoid this difficulty, the collection of system objectives is deformed into another topological space forming the homotopy function parameterized by computational temperature T [45–48]. Due to its analogy to Helmholtz free energy [12, 45–49], the approach of deterministic annealing (DA) converting the discrete variable space into continuous variable space of probability distribution is applied in minimizing the homotopy function. At every successive temperature drop, the minimization of the homotopy function is carried out using a second-order optimization scheme such as the Nearest Newton Descent Scheme [2–5, 7, 11–14, 16–29] or BFGS Scheme, and so forth.

At the end of every iteration, each agent i converges to a probability distribution clearly distinguishing the contribution of its every corresponding strategy value. For every agent, the strategy value with the maximum probability value is referred to as the favorable strategy and is used to compute the system objective and corresponding constraint functions. This system objective and corresponding strategy values are accepted based on a variation of the feasibility-based rule defined in [34] and further successfully implemented in [35–40]. This rule allows the objective function and the constraint information to be considered separately. The rule can be described as follows:

(a) any feasible solution is preferred over any infeasible solution;

(b) between two feasible solutions, the one with better objective is preferred;

(c) between two infeasible solutions, the one with fewer constraint violations is preferred.

In addition to the above, a perturbation approach is also incorporated to avoid premature convergence. It perturbs the individual agent's favorable strategy set based on its reciprocal and associated predefined interval. The solution is accepted if the feasibility is maintained. In this way, the algorithm continues until convergence by selecting the samples from the neighborhood of the recent favorable strategies. The neighborhood space is reduced or expanded according to the improvement in the system objective for a predefined number of iterations.

In some of the applications, the agents are also needed to provide the knowledge of the interagent relationship. It is one of the information/strategy sets which every other entitled agent is supposed to know. There is also global information that every agent is supposed to know. This allows agents to know the right to model other agents' actions or behavior. All of the decisions are taken autonomously by each agent considering the available information in order to optimize the local goals and hence to achieve the optimum global goal or system objective. The following section discusses the constrained PC procedure in detail.

3.1. Constrained PC Algorithm. Consider a general constrained problem (in the minimization sense) as follows:

$$\begin{aligned} &\text{Minimize} \quad G \\ &\text{Subject to} \quad g_j \le 0, \quad j = 1, 2, \ldots, s \\ &\qquad\qquad\ \ h_j = 0, \quad j = 1, 2, \ldots, w. \end{aligned} \tag{1}$$

According to [8–10], the equality constraint $h_j = 0$ can be transformed into a pair of inequality constraints using a tolerance value δ as follows:

$$h_j = 0 \Longrightarrow \begin{cases} g_{s+j} = h_j - \delta \le 0 \quad j = 1, 2, \ldots, w, \\ g_{s+w+j} = -\delta - h_j \le 0. \end{cases} \tag{2}$$

Thus, w equality constraints are replaced by $2w$ inequality constraints with the total number of constraints given by $t = s + 2w$. Then, a generalized representation of the problem in (1) can be stated as follows:

$$\begin{aligned} &\text{Minimize} \quad G \\ &\text{Subject to} \quad g_j \le 0, \quad j = 1, 2, \ldots, t. \end{aligned} \tag{3}$$

In the context of PC, the variables of the problem are considered as computational agents/players of a social game being played iteratively [3, 11]. Each agent i is given a predefined sampling interval referred to as $\Psi_i \in [\Psi_i^{\text{lower}}, \Psi_i^{\text{upper}}]$. As a general case, the interval can also be referred to as the sampling space. The lower limit Ψ_i^{lower} and upper limit Ψ_i^{upper} of the interval Ψ_i may be updated iteratively as the algorithm progresses.

Each agent i randomly samples $X_i^{[r]}$, $r = 1, 2, \ldots, m_i$ strategies from within the corresponding sampling interval Ψ_i forming a strategy set $\mathbf{X}_i$ represented as

$$\mathbf{X}_i = \left\{X_i^{[1]}, X_i^{[2]}, X_i^{[3]}, \ldots, X_i^{[m_i]}\right\}, \quad i = 1, 2, \ldots, N. \tag{4}$$

Every agent is assumed to have an equal number of strategies; that is, $m_1 = m_2 = \cdots = m_i = \cdots = m_{N-1} = m_N$. The procedure of modified PC theory is explained below in detail with the algorithm flowchart in Figure 1.

The procedure begins with the initialization of the sampling interval Ψ_i for each agent i, temperature $T \gg 0$ or $T = T_{\text{initial}}$ or $T \to \infty$ (simply high enough), the temperature step size α_T ($0 < \alpha_T \le 1$), convergence parameter $\varepsilon = 0.0001$, algorithm iteration counter $n = 1$, and number of test iterations n_{test}. The value of α_T and n_{test} are chosen based on preliminary trials of the algorithm. Furthermore, the constraint violation tolerance μ is initialized to the number of constraints $|\mathbf{C}|$; that is, $\mu = |\mathbf{C}|$, where $|\mathbf{C}|$ refers to the cardinality of the constraint vector $\mathbf{C} = [g_1, g_2, \ldots, g_t]$.

Step 1. Agent i selects its first strategy $X_i^{[1]}$ and samples randomly from other agents' strategies as well. This is a random guess by agent i about which strategies have been chosen by the other agents. This forms a "combined strategy set" $\mathbf{Y}_i^{[1]}$ given by

$$\mathbf{Y}_i^{[1]} = \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[1]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\}. \tag{5}$$

The superscript [?] indicates that it is a "random guess" and not known in advance. In addition, agent i forms one combined strategy set for every strategy r of its strategy set $\mathbf{X}_i$, as shown below:

$$\begin{aligned} \mathbf{Y}_i^{[2]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[2]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\}, \\ \mathbf{Y}_i^{[3]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[3]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\}, \\ &\vdots \\ \mathbf{Y}_i^{[r]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[r]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\}, \\ &\vdots \\ \mathbf{Y}_i^{[m_i]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[m_i]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\}. \end{aligned} \tag{6}$$

Similarly, all the remaining agents form their combined strategy sets.

Furthermore, every agent i computes m_i associated objective function values as follows:

$$\left[G\left(\mathbf{Y}_i^{[1]}\right), G\left(\mathbf{Y}_i^{[2]}\right), \ldots, G\left(\mathbf{Y}_i^{[r]}\right), \ldots, G\left(\mathbf{Y}_i^{[m_i]}\right)\right]. \tag{7}$$

The ultimate goal of every agent i is to identify its strategy value which contributes the most towards the minimization of the sum of these system objective values; that is, $\sum_{r=1}^{m_i} G(\mathbf{Y}_i^{[r]})$, hereafter referred to as the collection of system objectives.

Step 2. The minimum of the function $\sum_{r=1}^{m_i} G(\mathbf{Y}_i^{[r]})$ is very hard to achieve as the function may have many possible local minima. Moreover, directly minimizing this function is quite cumbersome as it may need excessive computational effort [3]. One of the ways to deal with this difficulty is to deform the function into another topological space by constructing a related and "easier" function $f(\mathbf{X}_i)$. Such a method is referred to as the homotopy method [45–48]. The function $f(\mathbf{X}_i)$ can be referred to as "easier" because it is easy to compute; the (global) minimum of such a function is known and easy to locate [45–47]. The deformed function can also be referred to as homotopy function J parameterized by computational temperature T represented as follows:

$$J_i(\mathbf{X}_i, T) = \sum_{r=1}^{m_i} G\left(\mathbf{Y}_i^{[r]}\right) + T f(\mathbf{X}_i), \quad T \in [0, \infty). \tag{8}$$

For further simplicity and understanding the above homotopy function, $J_i(\mathbf{X}_i, T)$ can be rewritten as

$$J_i(\mathbf{X}_i, T) = \sum_{r=1}^{m_i} G\left(\mathbf{Y}_i^{[r]}\right) - T f'(\mathbf{X}_i), \quad T \in [0, \infty). \tag{9}$$

The approach of deterministic annealing (DA) is applied to minimize the homotopy function in (9). The motivation behind this is its analogy to the Helmholtz free energy [26, 27]. It suggests the conversion of discrete variables into

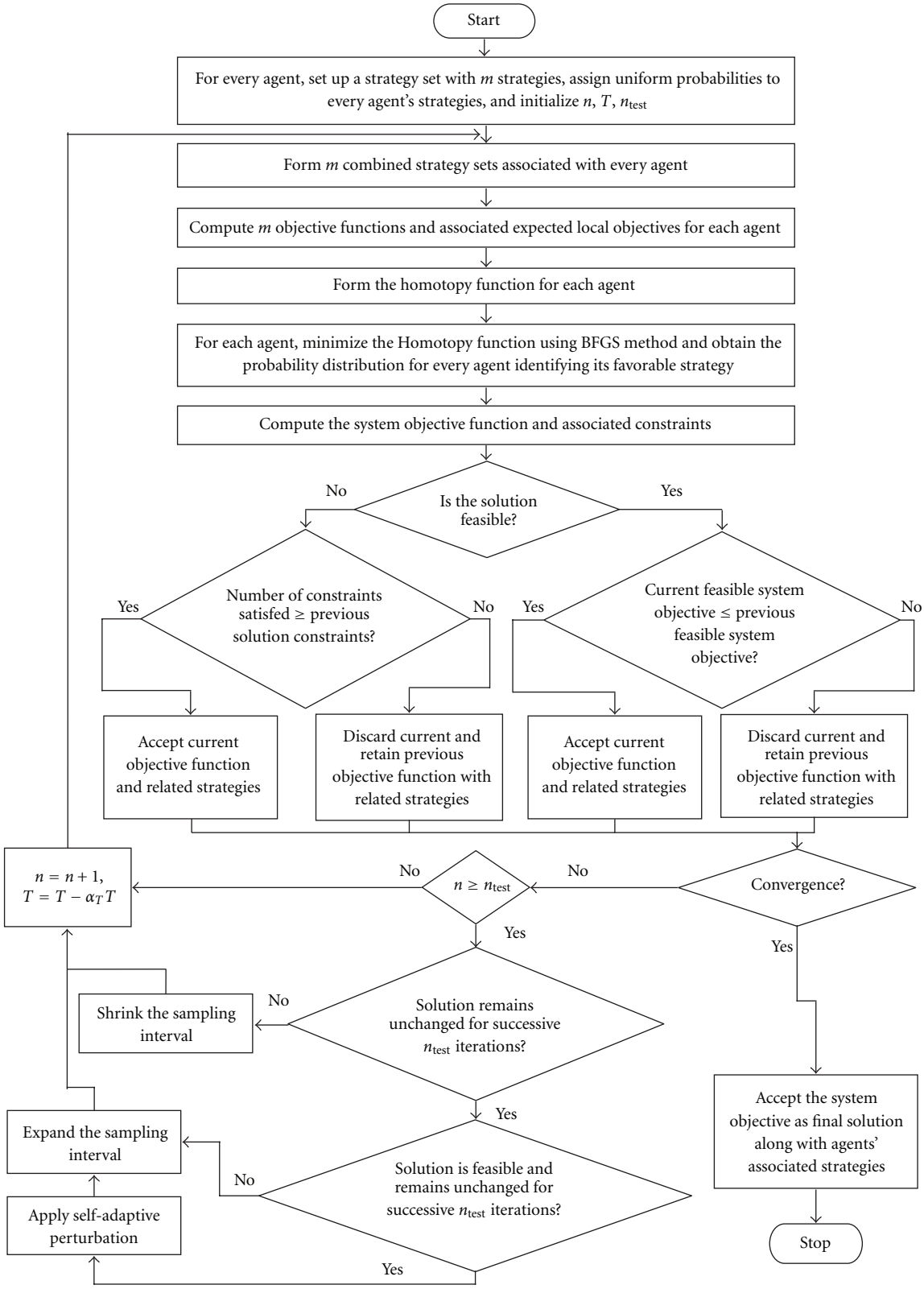


Figure 1: Constrained PC algorithm flowchart.

random real valued variables such as probabilities. This converts the original collection of system objectives $\sum_{r=1}^{m_i} G(\mathbf{Y}_i^{[r]})$ into the "expected collection of system objectives" $\sum_{r=1}^{m_i} E(G(\mathbf{Y}_i^{[r]}))$. Furthermore, a suitable function for $f(\mathbf{X}_i)$ is chosen. The general choice is to use the entropy function $S_i = -\sum_{r=1}^{m_i} q(X_i^{[r]}) \log_2 q(X_i^{[r]})$ [44–46]. The following steps of DA are formulated based on the analogy of the homotopy function to the Helmholtz free energy discussed in [12, 26, 27].

(a) Agent i assigns uniform probabilities to its strategies. This is because, at the beginning, the least information is available (the largest uncertainty and highest entropy) about which strategy is favorable for the minimization of the collection of system objectives $\sum_{r=1}^{m_i} \mathrm{G}(\mathbf{Y}_i^{[r]})$. Therefore, at the beginning of the "game," each agent's every strategy has probability $1/m_i$ of being most favorable. Therefore, probability of strategy r of agent i is

$$q\left(X_i^{[r]}\right) = \frac{1}{m_i}, \quad r = 1, 2, \ldots, m_i. \tag{10}$$

Each agent i, from its every combined strategy set $\mathbf{Y}_i^{[r]}$ and corresponding system objective $G(\mathbf{Y}_i^{[r]})$ computed previously, further computes m_i corresponding expected system objective values $E(G(\mathbf{Y}_i^{[r]}))$ as follows [2, 3, 11–14]:

$$E\left(G\left(\mathbf{Y}_i^{[r]}\right)\right) = G\left(\mathbf{Y}_i^{[r]}\right) q\left(X_i^{[r]}\right) \prod_{(i)} q\left(X_{(i)}^{[?]}\right), \tag{11}$$

where (i) represents every agent other than i. Every agent i then computes the expected collection of system objectives denoted by $\sum_{r=1}^{m_i} E(G(\mathbf{Y}_i^{[r]}))$. This also means that the PC approach can convert any discrete variables into continuous variable values in the form of probabilities corresponding to these discrete variables. As mentioned earlier, the problem now becomes continuous but still not easier to solve.

(b) Thus, the homotopy function to be minimized by each agent i in (9) is modified as follows:

$$\begin{aligned} J_i(q(\mathbf{X}_i), T) &= \sum_{r=1}^{m_i} E\left(G\left(\mathbf{Y}_i^{[r]}\right)\right) - TS_i \\ &= \sum_{r=1}^{m_i} \left(G\left(\mathbf{Y}_i^{[r]}\right) q\left(X_i^{[r]}\right) \prod_{(i)} q\left(X_{(i)}^{[?]}\right) \right) \\ &\quad - T\left(-\sum_{r=1}^{m_i} q\left(X_i^{[r]}\right) \log_2 q\left(X_i^{[r]}\right) \right) \\ &= G\left(\mathbf{Y}_i^{[1]}\right) q\left(X_i^{[1]}\right) \prod_{(i)} q\left(X_{(i)}^{[?]}\right) \\ &\quad + G\left(Y_i^{[2]}\right) q\left(X_i^{[2]}\right) \prod_{(i)} q\left(X_{(i)}^{[?]}\right) \\ &\quad + \cdots + G\left(\mathbf{Y}_i^{[m_i-1]}\right) q\left(X_i^{[m_i-1]}\right) \prod_{(i)} q\left(X_{(i)}^{[?]}\right) \\ &\quad + G\left(Y_i^{[m_i]}\right) q\left(X_i^{[m_i]}\right) \prod_{(i)} q\left(X_{(i)}^{[?]}\right) \\ &\quad - T\left(-\sum_{r=1}^{m_i} q\left(X_i^{[r]}\right) \log_2 q\left(X_i^{[r]}\right) \right), \end{aligned} \tag{12}$$

where $T \in [0, \infty)$. When the temperature T is high enough, the entropy term dominates the expected collection of system objectives and the problem becomes very easy to be solved.

Step 3. In the author's previous work [1, 4, 5, 26], Nearest Newton Descent Scheme [3] was implemented for minimizing the homotopy function $J_i(q(\mathbf{X}_i), T)$. Motivated from this scheme [3], the minimization of the homotopy function $J_i(q(\mathbf{X}_i), T)$ given in (12) is carried out using a suitable second-order optimization technique such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) scheme [50, 51]. It is important to mention that, similar to the Nearest Newton Descent Scheme, the BFGS scheme approximates positive definite Hessian. The BFGS scheme minimizing (12) is discussed in the appendix.

Step 4. For each agent i, the optimization process converges to a probability variable vector $q(\mathbf{X}_i)$ which can be seen as the individual agent's probability distribution distinguishing every strategy's contribution towards the minimization of the expected collection of system objectives $\sum_{r=1}^{m_i} E(G(\mathbf{Y}_i^{[r]}))$. In other words, for every agent i, if strategy r contributes the most towards the minimization of the objective compared to other strategies, its corresponding probability certainly increases by some amount more than those for the other strategies' probability values, and so strategy r is distinguished from the other strategies. Such a strategy is referred to as a favorable strategy $X_i^{[\mathrm{fav}]}$. As an illustration, the converged probability distribution for agent i may look like that shown in Figure 2 for a case where there are 10 strategies, that is, $m_i = 10$.

Compute the corresponding system objective $G(\mathbf{Y}^{[\mathrm{fav}]})$ and constraint vector $\mathbf{C}(\mathbf{Y}^{[\mathrm{fav}]})$ where $\mathbf{Y}^{[\mathrm{fav}]}$ is given by

$$\mathbf{Y}^{[\mathrm{fav}]} = \left\{ X_1^{[\mathrm{fav}]}, X_2^{[\mathrm{fav}]}, \ldots, X_{N-1}^{[\mathrm{fav}]}, X_N^{[\mathrm{fav}]} \right\}. \tag{13}$$

Step 5. Accept the system objective $G(\mathbf{Y}^{[\mathrm{fav}]})$ and corresponding $\mathbf{Y}^{[\mathrm{fav}]}$ as current solution if the number of constraints violated $C_{\mathrm{violated}} \leq \mu$. Update the constraint violation tolerance $\mu = C_{\mathrm{violated}}$ and continue to Step 6.

If $C_{\mathrm{violated}} > \mu$, then discard current system objective $G(\mathbf{Y}^{[\mathrm{fav}]})$ and corresponding $\mathbf{Y}^{[\mathrm{fav}]}$, and retain the previous iteration solution and continue to Step 6.

If the current system objective $G(\mathbf{Y}^{[\mathrm{fav}]})$ is feasible, that is, $\mu = C_{\mathrm{violated}} = 0$ and is not worse than the previous feasible solution, accept the current system objective $G(\mathbf{Y}^{[\mathrm{fav}]})$ and corresponding $\mathbf{Y}^{[\mathrm{fav}]}$ as current solution and continue to *Step 6*; else discard current feasible system objective $G(\mathbf{Y}^{[\mathrm{fav}]})$ and corresponding $\mathbf{Y}^{[\mathrm{fav}]}$, and retain the previous iteration feasible solution and continue to Step 6.

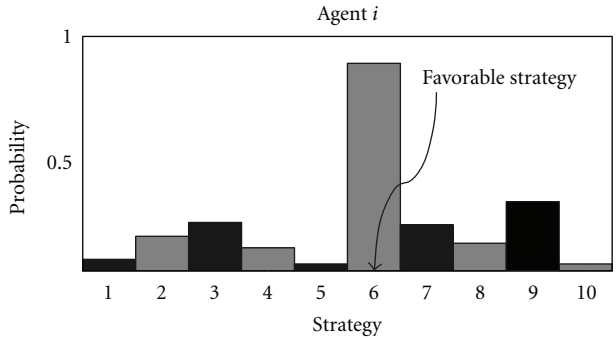


Figure 2: Probability Distribution of agent i.

Step 6. On the completion of prespecified n_{test} iterations, the following conditions are checked for every further iteration.

(a) If $G(\mathbf{Y}^{[\text{fav}],n}) \leq G(\mathbf{Y}^{[\text{fav}],n-n_{\text{test}}})$, then every agent shrinks its sampling interval as follows:

$$\Psi_i \in \left[\left(X_i^{[\text{fav}]} - \left\|\Psi_i^{\text{upper}} - \Psi_i^{\text{lower}}\right\| \cdot \lambda_{\text{down}}\right), \left(X_i^{[\text{fav}]} + \left\|\Psi_i^{\text{upper}} - \Psi_i^{\text{lower}}\right\| \cdot \lambda_{\text{down}}\right)\right], \quad 0 < \lambda_{\text{down}} \leq 1, \tag{14}$$

where λ_{down} is referred to as the interval factor corresponding to the shrinking of sample space.

(b) If $G(\mathbf{Y}^{[\text{fav}],n})$ and $G(\mathbf{Y}^{[\text{fav}],n-n_{\text{test}}})$ are feasible and $\|G(\mathbf{Y}^{[\text{fav}],n}) - G(\mathbf{Y}^{[\text{fav}],n-n_{\text{test}}})\| \leq \varepsilon$, then the system objective $G(\mathbf{Y}^{[\text{fav}],n})$ can be referred to as a stable solution $G(\mathbf{Y}^{[\text{fav}],s})$ or possible local minimum. In order to jump out of this possible local minimum, a perturbation approach is incorporated. It is described below.

Every agent i perturbs its current favorable strategy $X_i^{[\text{fav}]}$ by a perturbation factor fact_i corresponding to the reciprocal of its favorable strategy $X_i^{[\text{fav}]}$ as follows:

$$X_i^{[\text{fav}]} = X_i^{[\text{fav}]} \pm \left(X_i^{[\text{fav}]} \times \text{fact}_i\right), \tag{15}$$

where

$$\text{fact}_i = \begin{cases} \text{random value} \in \left(\sigma_1^{\text{lower}}, \sigma_1^{\text{upper}}\right) & \text{if } \dfrac{1}{X_i^{[\text{fav}]}} \leq \gamma, \\ \text{random value} \in \left(\sigma_2^{\text{lower}}, \sigma_2^{\text{upper}}\right) & \text{if } \dfrac{1}{X_i^{[\text{fav}]}} > \gamma \end{cases} \tag{16}$$

and $\sigma_1^{\text{lower}}, \sigma_1^{\text{upper}}, \sigma_2^{\text{lower}}, \sigma_2^{\text{upper}}$ are randomly generated values between 0 and 1; that is, $0 < \sigma_1^{\text{lower}}, \sigma_1^{\text{upper}}, \sigma_2^{\text{lower}}, \sigma_2^{\text{upper}} < 1$, and $\sigma_1^{\text{lower}} < \sigma_1^{\text{upper}} \leq \sigma_2^{\text{lower}} < \sigma_2^{\text{upper}}$. The value of γ as well as "+" or "−" sign in (15) is chosen based on the preliminary trials of the algorithm.

It gives a chance to every agent i to jump out of the local minima and further may help to search for a better solution. The perturbed solution is accepted if and only if the feasibility is maintained. Furthermore, every agent expands its sampling interval as follows:

$$\Psi_i \in \left[\left(\Psi_i^{\text{lower}} - \left\|\Psi_i^{\text{upper}} - \Psi_i^{\text{lower}}\right\| \cdot \lambda_{\text{up}}\right), \left(\Psi_i^{\text{upper}} + \left\|\Psi_i^{\text{upper}} - \Psi_i^{\text{lower}}\right\| \cdot \lambda_{\text{up}}\right)\right], \quad 0 < \lambda_{\text{up}} \leq 1, \tag{17}$$

where λ_{up} is referred to as the interval factor corresponding to the expansion of sample space.

Step 7. If either of the two criteria listed below is valid, accept the current stable system objective $G(\mathbf{Y}^{[\text{fav}],s})$ and corresponding $\mathbf{Y}^{[\text{fav}],s}$ as the final solution referred to as $G(\mathbf{Y}^{[\text{fav}],\text{final}})$ and $\mathbf{Y}^{[\text{fav}],\text{final}} = \{X_1^{[\text{fav}],\text{final}}, X_2^{[\text{fav}],\text{final}}, \ldots, X_{N-1}^{[\text{fav}],\text{final}}, X_N^{[\text{fav}],\text{final}}\}$, respectively and stop; else continue to Step 8.

(a) If temperature $T = T_{\text{final}}$ or $T \to 0$.

(b) If there is no significant change in the successive stable system objectives (i.e., $\|G(\mathbf{Y}^{[\text{fav}],s}) - G(\mathbf{Y}^{[\text{fav}],s-1})\| \leq \varepsilon$) for two successive implementations of the perturbation approach.

Step 8. Each agent i then samples m_i strategies from within the updated sampling interval Ψ_i and forms the corresponding updated strategy set $\mathbf{X}_i$ represented as follows:

$$\mathbf{X}_i = \left\{X_i^{[1]}, X_i^{[2]}, X_i^{[3]}, \ldots, X_i^{[m_i]}\right\}, \quad i = 1, 2, \ldots, N. \tag{18}$$

Reduce the temperature $T = T - \alpha_T T$, update the iteration counter $n = n + 1$, and return to Step 1.

3.2. Nash Equilibrium. To achieve a Nash equilibrium, every agent in a MAS should have the properties of rationality and convergence [41–44]. Rationality refers to the property by which every agent selects (or converges to) the best possible strategy given the strategies of the other agents. The convergence property refers to the stability condition, that is, a policy using which every agent selects (or converges to) the best possible strategy when all the other agents use their policies from a predefined class (preferably same class). The Nash equilibrium is naturally achieved when all the agents in a MAS are convergent and rational. Moreover, a Nash equilibrium is guaranteed when all the agents use stationary policies, that is, those policies that do not change over time. It is worth to mention here that all the agents in the MAS proposed using PC algorithm exhibit the above-mentioned properties. It is elaborated in the detailed PC algorithm discussed in the previous few paragraphs.

In any game, there may be a large but finite number of Nash equilibria present, depending on the number of strategies per agent as well as the number of agents. It is essential to choose the best possible combination of the individual strategies selected by each agent. It is quite hard to go through every possible combination of the individual agent strategies and choose the best out of it that can produce a best possible Nash equilibrium and hence the system objective.

As discussed in the detailed PC algorithm, in each iteration n, every agent i selects the best possible strategy referred to as the favorable strategy $X_i^{[\text{fav}],n}$ by guessing the possible strategies of the other agents. This information about its favorable strategy $X_i^{[\text{fav}],n}$ is made known to all the other agents as well. In addition, the corresponding global knowledge such as system objective value $G(\mathbf{Y}^{[\text{fav}],n}) = G(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n})$ is also available to each agent which clearly helps all the agents take the best possible informed decision in every further iteration. This makes the entire system ignore a considerably large number of Nash equilibria but select the best possible one in each iteration and accept the corresponding system objective $G(\mathbf{Y}^{[\text{fav}],n})$. Mathematically, the Nash equilibrium solution in any iteration can be represented as follows:

$$\begin{aligned}
&G\left(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n}\right) \\
&\quad \le G\left(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n}\right), \\
&G\left(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n}\right) \\
&\quad \le G\left(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n}\right), \quad (19) \\
&\qquad \vdots \\
&G\left(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n}\right) \\
&\quad \le G\left(X_1^{[\text{fav}],n}, X_2^{[\text{fav}],n}, X_3^{[\text{fav}],n}, \ldots, X_{N-1}^{[\text{fav}],n}, X_N^{[\text{fav}],n}\right),
\end{aligned}$$

where $X_i^{[\text{fav}],n}$ represents any strategy other than the favorable strategy $X_i^{[\text{fav}],n}$ from the same sample space Ψ_i^n.

Furthermore, from this current Nash equilibrium point with system objective $G(\mathbf{Y}^{[\text{fav}],n})$, the algorithm progresses to the next Nash equilibrium point with better system objective $G(\mathbf{Y}^{[\text{fav}],n+1})$, that is, $G(\mathbf{Y}^{[\text{fav}],n}) \ge G(\mathbf{Y}^{[\text{fav}],n+1})$. As the algorithm progresses, those ignored Nash equilibria as well as the best Nash equilibria selected at previous iterations would be noticed as inferior solutions.

This process continues until there is no change in the current solution $G(\mathbf{Y}^{[\text{fav}],n})$, that is, no new Nash equilibrium has been identified that proves the current Nash equilibrium to be inferior. Hence, the system exhibits stage-wise convergence to a unique Nash equilibrium and the corresponding system objective is accepted as the final solution $\mathrm{G}(\mathbf{Y}^{[\text{fav}],\text{final}})$. As a general case, this progress can be represented as $G(\mathbf{Y}^{[\text{fav}],1}) \ge G(\mathbf{Y}^{[\text{fav}],2}) \ge \cdots \ge G(\mathbf{Y}^{[\text{fav}],n}) \ge G(\mathbf{Y}^{[\text{fav}],n+1}) \ge \cdots \ge G(\mathbf{Y}^{[\text{fav}],\text{final}})$.

4. The Circle Packing Problem (CPP)

A generalized packing problem consists of determining how best to pack z objects into a predefined bounded space that yields best utilization of space with no overlap of object boundaries [52, 53]. The bounded space can also be referred to as a container. The packing objects and container can be circular, rectangular, or irregular. Although the problem appears rather simple and in spite of its practical applications in production and packing for the textile, apparel, naval, automobile, aerospace, food industries, and so forth [54], the CPP received considerable attention in the "pure" mathematics literature but only limited attention in the operations research literature [55]. As it is proven to be a NP-hard problem [53, 56–58] and cannot be effectively solved by purely analytical approaches [59–69], a number of heuristic techniques were proposed solving the CPP [52, 53, 70–82]. Most of these approaches address the CPP in limited ways, such as close packing of fixed and uniform sized circles inside a square or circle container [53, 59–70], close packing of fixed and different-sized circles inside a square or circle container [52, 54, 75–82], simultaneous increase in the size of the circles covering the maximum possible area inside a square [71–74], and so forth.

As per knowledge of the authors of this paper, the CPP was never solved in a distributed way. In this paper, as a distributed MAS, every individual circle changes its size and position autonomously. This allows for addressing the important issue of the avoidance of the tragedy of commons which was also never addressed before in the context of the CPP. The next few sections describe the mathematical formulation and the solution to two cases of the CPP.

4.1. Formulation of the CPP. The objective of the CPP solved here was to cover the maximum possible area within a square by z number of circles without overlapping one another or exceeding the boundaries of the square. In order to achieve this objective, all the circles were allowed to increase their sizes as well as change their locations. The problem is formulated as follows:

$$\text{Minimize} \quad f = L^2 - \sum_{i=1}^{z} \pi r_i^2 \tag{20}$$

$$\text{Subject to} \quad \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \ge r_i + r_j, \tag{21}$$

$$x_l \le x_i - r_i, \tag{22}$$

$$x_u \ge x_i + r_i, \tag{23}$$

$$y_l \le y_i - r_i, \tag{24}$$

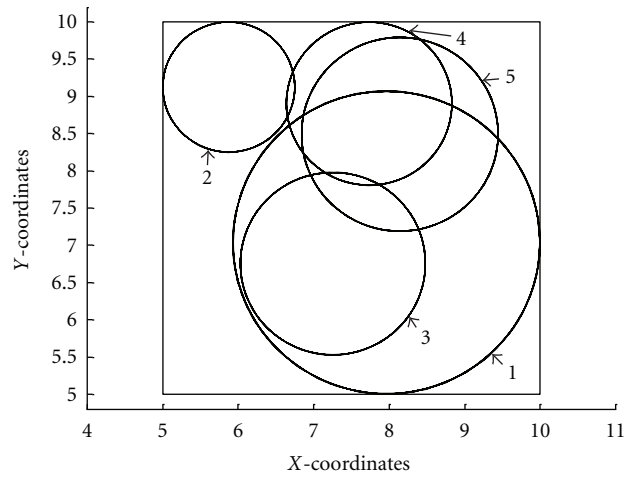
$$y_u \ge y_i + r_i, \tag{25}$$

$$0.001 \le r_i \le \frac{L}{2}, \tag{26}$$

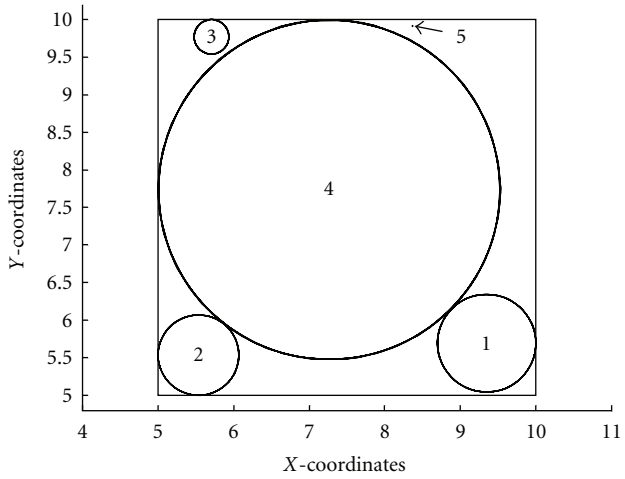
$$i,\ j = 1, 2, \ldots, z, \quad i \ne j, \tag{27}$$

where L is length of the side of the square, r_i is radius of circle i, x_i, y_i are x and y coordinates of the center of circle i, x_l, y_l are x and y coordinates of the lower left corner of the square, x_u, y_u are x and y coordinates of the upper right corner of the square.

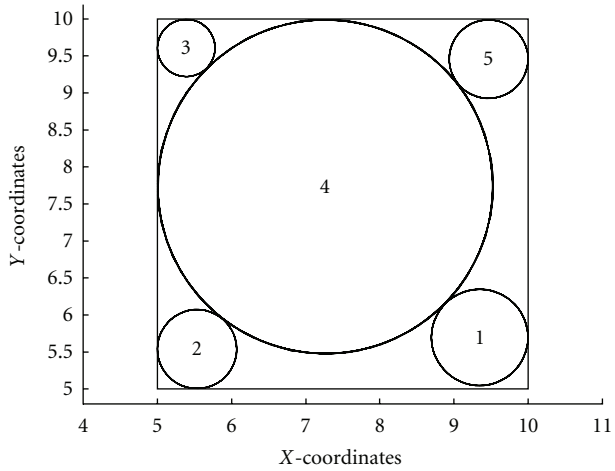
In solving the proposed CPP using constrained PC approach, the circles were considered as autonomous agents. These circles were assigned the strategy sets of X-coordinates and Y-coordinates of the center and the radius. Two cases of the CPP were solved. These cases differ from each other based



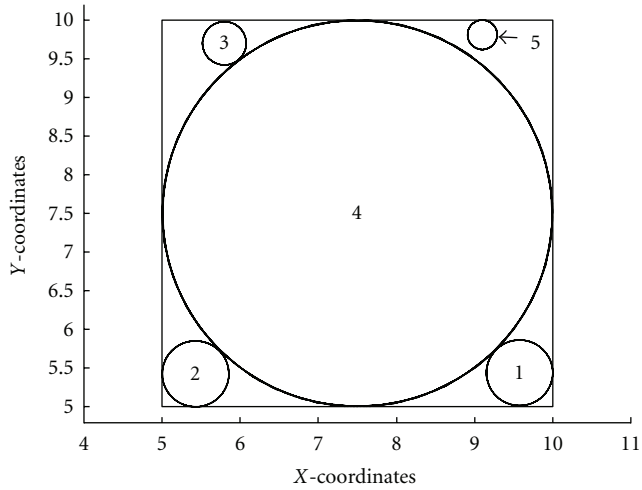
(a) Randomly generated initial solution



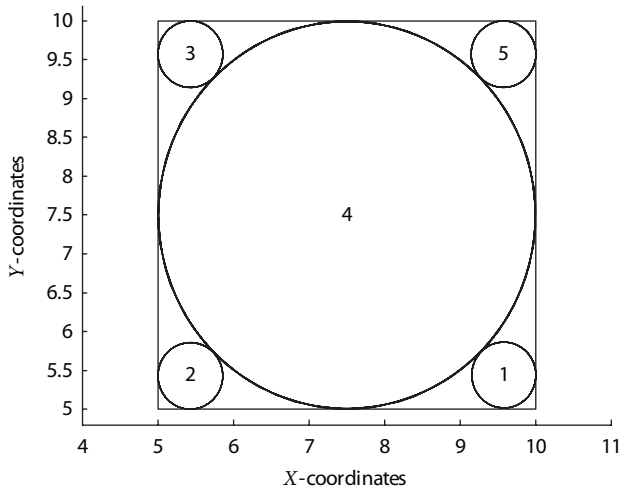
(b) Solution at iteration 401



(c) Solution at iteration 901



(d) Solution at iteration 1001



(e) Stable solution at iteration 1055

Figure 3: Stepwise solution for Case 1.

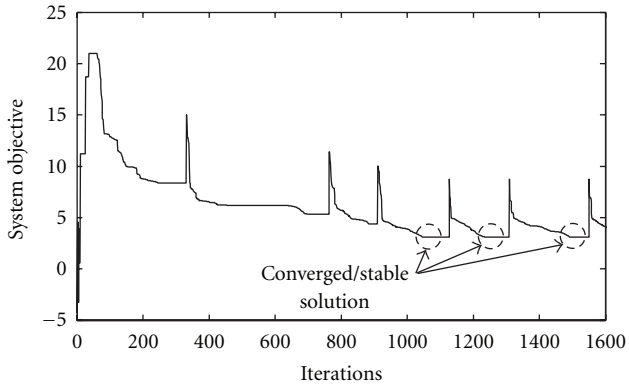


Figure 4: Convergence of the objective function for Case 1.

on the initial configuration (location) of the circles as well as the constraint handling method(s) applied solving each case. In Case 1, the circles were randomly initialized inside the square and were not allowed to cross the square boundaries. The constraints in (21) were satisfied using the feasibility-based approach described in Section 3. And the constraints in (22) to (25) were satisfied in every iteration of the algorithm using a repair approach. The repair approach refers to pushing the circles inside the square if they crossed the boundaries of it. It is similar to the one implemented by the authors of this paper in their previous work [1, 26]. In Case 2, the circles were randomly located in and around the square and all the constraints from (21) to (25) were satisfied using the feasibility-based approach described in Section 3. The initial configuration of Case 1 and Case 2 is shown in Figures 3(a) and 6(a), respectively.

The constrained PC algorithm solving both cases was coded in MATLAB 7.8.0 (R2009A), and the simulations were run on a Windows platform using an Intel Core 2 Duo, 3 GHz processor speed and 3.25GB memory capacity. Furthermore, for both cases, the set of parameters chosen was as follows: (a) individual agent sample size $m_i = 5$, (b) number of test iterations $n_{\text{test}} = 20$, (c) the shrinking interval factor $\lambda_{\text{down}} = 0.05$, (d) the expansion interval factor $\lambda_{\text{up}} = 0.1$, (e) perturbation parameters $\sigma_1^{\text{lower}} = 0.001$, $\sigma_1^{\text{upper}} = 0.01$, $\sigma_2^{\text{lower}} = 0.5$, $\sigma_2^{\text{upper}} = 0.7$, $\gamma = 0.99$, and the sign in (15) was chosen to be "$-$". In addition to it, a voting heuristic was also incorporated in the constrained PC algorithm. It is described in the Section 4.4.

4.2. Case 1: CPP with Circles Randomly Initialized Inside the Square. In this case of the CPP, five circles ($z = 5$) were initialized randomly inside the square without exceeding the boundary edges of the square. The length of the side of the square was five units (i.e., $L = 5$). More than 30 runs of the constrained PC algorithm described in Section 3 were conducted solving Case 1 of the CPP with different initial configurations of the circles inside the square. The true optimum solution was achieved in every run with the average CPU time of 14.05 minutes, and average number of function evaluations is 17515.

The randomly generated initial solution, the intermediate iteration solutions, and the converged true optimum solution from one of the instances are presented in Figure 3. The corresponding convergence plot of the system objective is presented in Figure 4. The convergence of the associated variables such as radius of the circles, X-coordinates, and Y-coordinates of the center of the circles is presented in Figures 5(a), 5(b) and 5(c), respectively. The solution was converged at iteration 1035 with 26910 function evaluations. The true optimum value of the objective function (f) achieved was 3.0807 units.

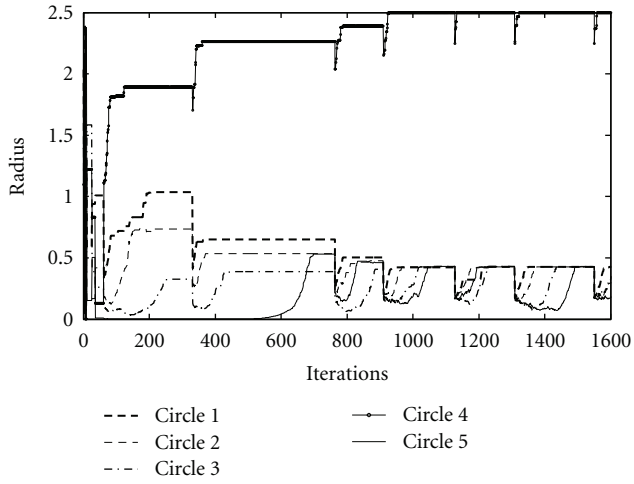
As mentioned before, the algorithm was assumed to have converged when successive implementations of the perturbation approach stabilize to equal objective function value. It is evident from Figures 3, 4, and 5 that the solution was converged to true optimum at iteration 1055 as the successive implementations of the perturbation approach produced stable and equal objective function values. Furthermore, it is also evident from Figures 4 and 5 that the solution was perturbed at iteration 778, 901, 1136, and 1300. It is clear that the implementation of the perturbation approach at iteration 901 helped the solution to jump out of the local minima and further achieve the true optimum solution at iteration 1106.

4.3. Case 2: CPP with Circles Randomly Initialized. In this case of the CPP, five circles ($z = 5$) were initialized randomly in the space with no restriction as in Case 1 where circles were randomly placed inside the square. The length of the side of the square was five units (i.e., $L = 5$). Similar to Case 1, more than 30 runs of the constrained PC algorithm described in Section 3 with different initial configuration of the circles were conducted solving Case 2. The true optimum solution was achieved in every run with the average CPU time of 14.05 minutes, and average number of function evaluations is 68406.

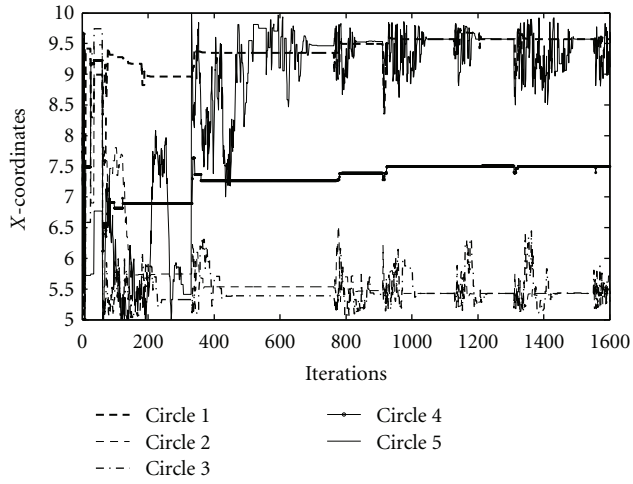
The randomly generated initial solution, the intermediate iteration solutions, and the converged true optimum solution from one of the instances of Case 2 are presented in Figure 6. The corresponding convergence plot of the system objective is presented in Figure 7. The convergence of the associated variables is presented in Figures 8(a), 8(b), and 8(c), respectively. The solution was converged at iteration 955 with 24830 function evaluations. The true optimum value of the objective function (f) achieved was 3.0807 units.

In the instance of Case 2 represented here, the solution was perturbed at iteration 788, 988, 1170, and 1355. It is clear that the implementation of the perturbation approach at iteration 788 helped the solution to jump out of the local minima and further achieve the true optimum solution at iteration 955. It is important to mention that the instance illustrated here did not require the voting heuristic to be applied.

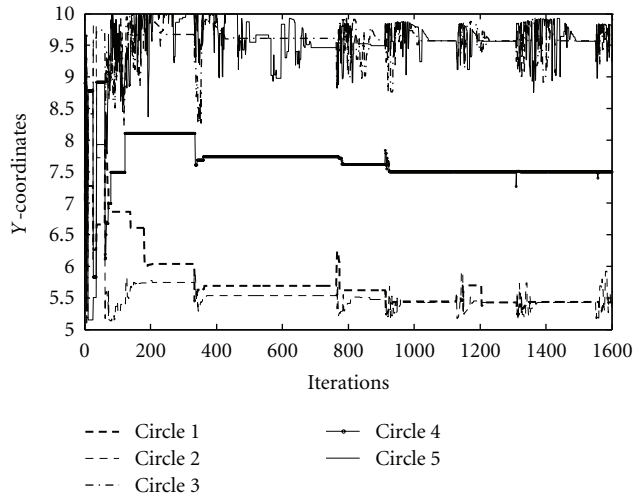
4.4. Voting Heuristic. In a few instances of the CPP cases solved here, in order to jump out of the local minimum, a voting heuristic was required. It was implemented in conjunction with the perturbation approach. Once the solution was perturbed, every circle voted 1 to each quadrant which it does not belong to at all and voted 0 otherwise. The circle



(a) Convergence of the radius



(b) Convergence of the X-coordinates of the center



(c) Convergence of the Y-coordinates of the center

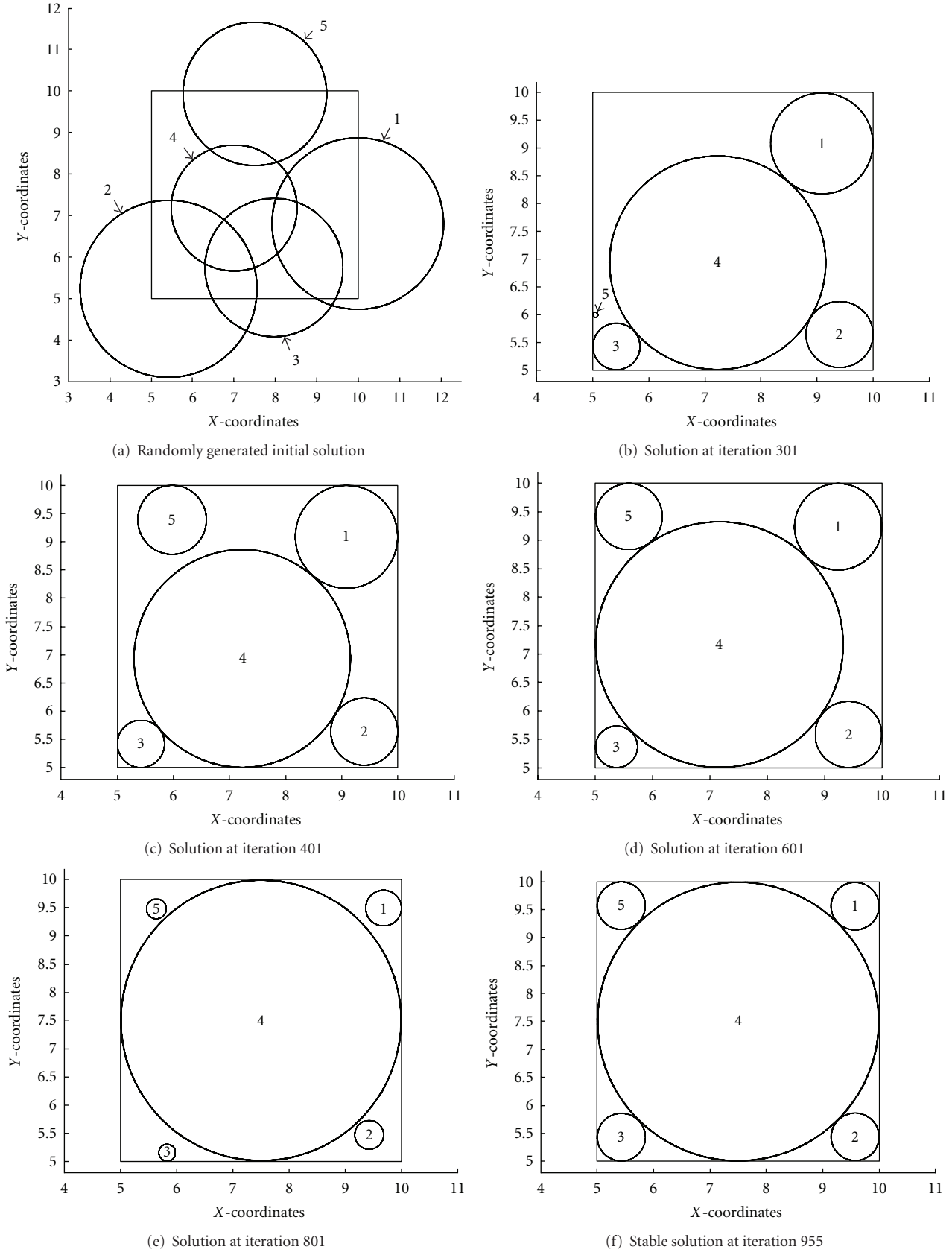
Figure 5: Convergence of the strategies of circle 1.

with the smallest size shifted itself to the extreme corner of the quadrant with the highest number of votes, that is, the winner quadrant. The new position of the smallest size circle was confirmed only when the solution remained feasible and the algorithm continues. If all the quadrants acquire equal number of votes, then no circle moves its position and the algorithm continues. The voting heuristic is demonstrated in Figure 9.

A voting grid corresponding to every quadrant of the square in Figure 9(a) is represented in Figure 9(b). The solid circles represent the solution before perturbation while corresponding perturbed ones are represented in dotted lines. The votes given by the perturbed circles (dotted circles) to the quadrants are presented in the grid. As the maximum number of votes are given to quadrant 1 (i.e., Q 1), the circle with smallest size (circle 3) shifts to the extreme corner of the quadrant Q 1 and confirms the new position as the solution remains feasible. Based on the trials conducted so far, it was noticed that the voting heuristic was not necessary to be implemented in every run of the constrained PC algorithm solving the CPP. Moreover, in those of the few cases in which the voting heuristic was required, it was required to be implemented only once in the entire execution of the algorithm. A variant of the voting heuristic was also implemented in conjunction with energy landscape paving algorithm [54, 76, 77]. The smallest circle was picked and placed randomly at the vacant place producing new configuration. It was claimed that this heuristic helped the algorithm jump out of the local minima. Furthermore, this heuristic was required to be implemented in every iteration of the algorithm.

5. Discussion

The above solutions using constrained PC indicated that it could successfully be used to solve constrained optimization problems such as the CPP. It is evident from the results that the approach was sufficiently robust and produced true optimum results in every run of both cases. It implies that the rational behavior of the agents could be successfully formulated and demonstrated. It is important to highlight that



(a) Randomly generated initial solution

(b) Solution at iteration 301

(c) Solution at iteration 401

(d) Solution at iteration 601

(e) Solution at iteration 801

(f) Stable solution at iteration 955

Figure 6: Stepwise solution for Case 2.

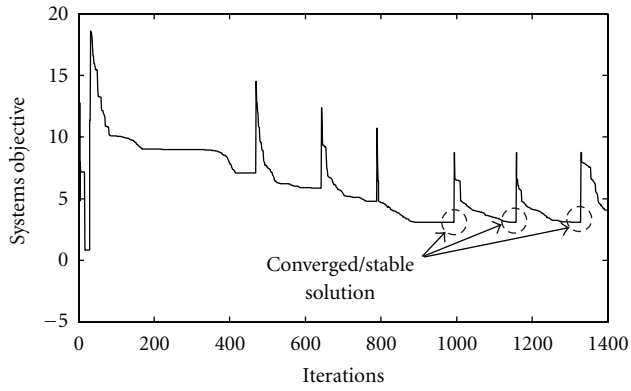


Figure 7: Solution convergence plot for Case 2.

the distributed nature of the PC approach allowed the total number of function evaluations to be equally divided among the agents of the system. This can be made practically evident by implementing the PC approach on a real distributed platform assigning separate workstations carrying out the computations independently. These advantages along with the directly incorporated uncertainty using the real-valued probabilities treated as variables suggest that PC can potentially be applied to real world complex problems.

It is worth to mention some of the key differences of the PC methodology presented here and the original PC approach [2, 3, 13, 14]. In the present approach, fewer numbers of samples were drawn from the uniform distribution of the individual agent's sampling interval. On the contrary, the original PC approach used a Monte Carlo sampling method which was computationally expensive and slow as the number of samples needed was in the thousands or even millions. Most significantly, the sampling in further stages of the PC algorithm presented here was narrowed down in every iteration by selecting the sampling interval in the neighborhood of the most favorable value in the particular iteration. This ensures faster convergence and an improvement in efficiency over the original PC approach in which regression was necessary to sample the strategy values in the close neighborhood of the favorable value. Moreover, the coordination among the agents representing the variables in the system was achieved based on the partial small bit of information. In other words, in order to optimize the global/system objective, every agent selects its best possible strategy by guessing the model of every other agent based merely on their recent favorable strategies communicated. This gives the advantage to the agents and the entire system to quickly search the better solution and reach the Nash equilibrium and avoid the tragedy of commons.

In addition, the feasibility-based rule in [34–40] suffered from maintaining the diversity and further required additional techniques such as niching [34], SA [35], modified mutation approach [38, 39], and several associated trials in [36–39], and so forth. It may require further computations and memory usage. On the other hand, a simple perturbation approach assisting the feasibility-based rule implemented here was computationally cheaper and requires no additional memory usage.

In agreement with the no-free-lunch theorem [83], some limitations were also identified. The rate of convergence and the quality of the solution were dependent on the parameters such as the number of strategies m_i in every agent's strategy set X_i, the interval factor λ, number of test iterations n_{test}, the shrinking interval factor λ_{down}, the expansion interval factor λ_{up}, and also the perturbation parameters such as σ_1^{lower}, σ_1^{upper}, σ_2^{lower}, σ_2^{upper}, and γ. It necessitated some preliminary trials for fine-tuning these parameters. Additionally, in order to confirm the convergence, the algorithm was required to be run beyond the convergence for a considerable number of iterations.

6. Concluding Remarks and Future Work

This paper proposes a generalized constrained PC approach using a variation of the feasibility-based rule originally proposed in [34]. Similar to [27], the constraint violation tolerance was iteratively tightened in order to obtain the fitter solution. In addition, in order to jump out of the possible local minima, the perturbation approach was successfully incorporated into the constrained PC algorithm. The concept of Nash equilibrium was also successfully formalized and demonstrated. Furthermore, the authors believe that the PC algorithm is made simpler and faster by improving the sampling method, the convergence criterion, and most importantly the neighboring approach narrowing the sampling options.

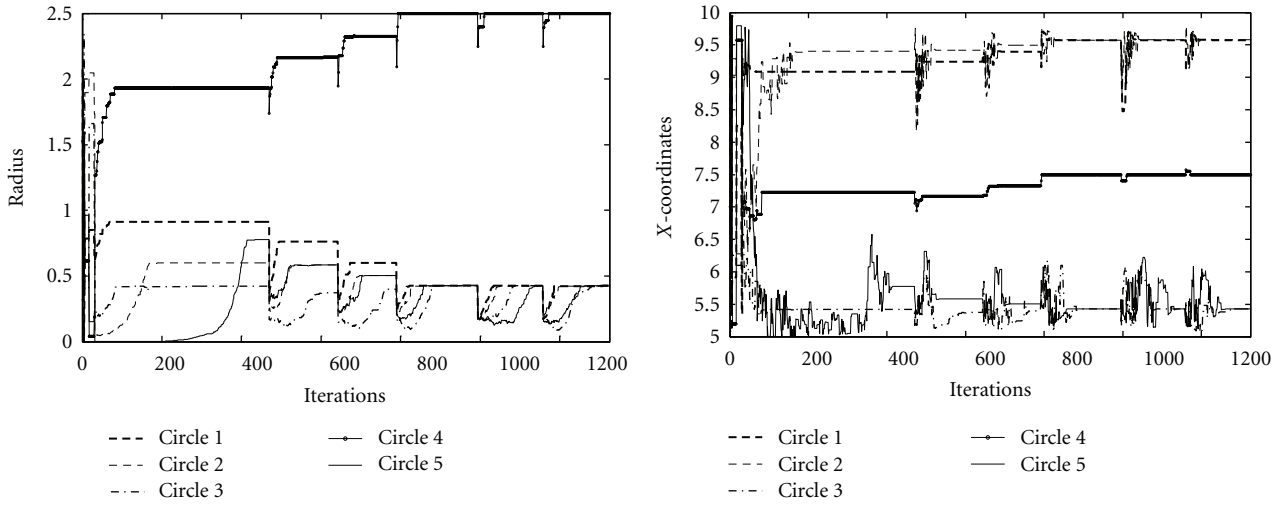
The constrained PC approach was successfully demonstrated solving two cases of the CPP. In both cases, the approach could find the true optimal solution in reasonable computational efforts. It is important to mention that the concept of the avoidance of tragedy of commons was also successfully demonstrated solving two cases of the CPP. Although only inequality constraints were handled in both cases of the CPP solved here, the approach of transformation of the equality constraints into the inequality constraints [8–10] can be implemented.

In the future, to make the approach more generalized and to improve the diversification of sampling, the rate of convergence, the quality of results, and so forth, a self-adaptive scheme can be developed for the parameters such as the number of strategies m_i and interval factor λ. Furthermore, the constraint handling technique may be further improved/developed using a multicriteria optimization approach [8–10, 84]. The constrained PC approach can be used for solving more realistic problems such as machine shop scheduling and urban traffic, and so forth. The authors also see some potential in the field of healthcare systems management [85].

Appendix

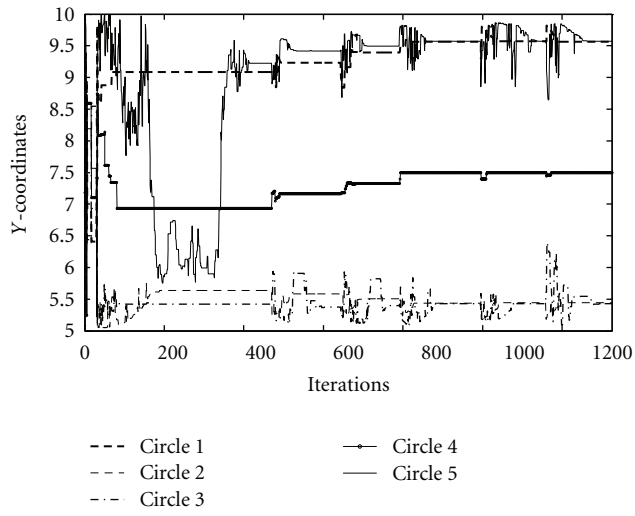
Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method Minimizing the Homotopy Function

The minimization of the Homotopy function given in (12) was carried out using a suitable second-order optimization technique such as Broyden-Fletcher-Goldfarb-Shanno



(a) Convergence of the radius

(b) Convergence of the X-coordinates of the center



(c) Convergence of the Y-coordinates of the center

FIGURE 8: Convergence of the strategies of circle 2.

(BFGS) method [47, 48]. The approximated Hessian in this method is positive definite. Moreover, the updating of the Hessian also preserves the positive definiteness. The BFGS method minimizing the Homotopy function in (12) is discussed below.

(1) Set BFGS iteration counter $k = 1$, BFGS maximum number of iterations ν, and step size α_{step} ($0 < \alpha_{\text{step}} \leq 1$). The value of α_{step} is held constant throughout the optimization and chosen based on the preliminary trials of the algorithm.

(a) Initialize the convergence criterion $q(\mathbf{X}_i)^k - q(\mathbf{X}_i)^{k-1} \leq \varepsilon_2$. The convergence parameter $\varepsilon_2 = 0.0001$ is equal for all the N agents.

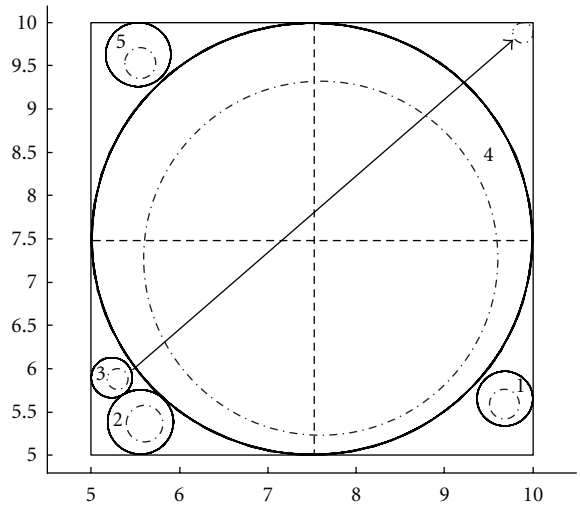
(b) Initialize the Hessian $\mathbf{H}_i^k$ to a positive definite matrix, preferably identity matrix $\mathbf{I}$ of size $m_i \times m_i$.

(c) Initialize the probability variables as follows:

$$q(X_i)^k = \left\{ \left(q\left(\mathbf{X}_i^{[1]}\right)^k = \frac{1}{m_i} \right), \left(q\left(X_i^{[2]}\right)^k = \frac{1}{m_i} \right), \ldots, \left(q\left(X_i^{[m_i]}\right)^k = \frac{1}{m_i} \right) \right\}. \tag{A.1}$$

That is, assign uniform probabilities to the strategies of agent i. This is because, at the beginning, least information is available (largest uncertainty and highest entropy) about which strategy is favorable for the minimization of the collection of system objectives $\sum_{r=1}^{m_i} \mathrm{G}(\mathbf{Y}_i^{[r]})$.

(d) Compute the gradient of the Homotopy function in (12) as follows



(a) A case for voting heuristic

Q_2				
1	2	3	4	5
1	1	1	0	0

Total votes = 3

Q_1				
1	2	3	4	5
1	1	1	0	1

Total votes = 4

Q_3				
1	2	3	4	5
1	0	0	0	1

Total votes = 2

Q_4				
1	2	3	4	5
0	1	1	0	1

Total votes = 3

(b) Voting grids

Figure 9: Voting heuristic.

$$\mathbf{C}^k = \left[\frac{\partial J_i(q(\mathbf{X}_i), T)^k}{\partial q\left(X_i^{[1]}\right)^k} \quad \frac{\partial J_i(q(\mathbf{X}_i), T)^k}{\partial q\left(X_i^{[2]}\right)^k} \quad \cdots \quad \frac{\partial J_i(q(\mathbf{X}_i), T)^k}{\partial q\left(X_i^{[m_i]}\right)^k} \right] = \left[G\left(\mathbf{Y}_i^{[1]}\right) \cdot \prod q\left(X_{(i)}^{[?]}\right) + \frac{T}{\ln(2)} \left[1 + \ln\left(q\left(X_i^{[1]}\right)^k \right) \right] \right.$$

$$\left. G\left(\mathbf{Y}_i^{[1]}\right) \cdot \prod q\left(X_{(i)}^{[?]}\right) + \frac{T}{\ln(2)} \left[1 + \ln\left(q\left(X_i^{[2]}\right)^k \right) \right] \cdots G\left(\mathbf{Y}_i^{[m_i]}\right) \cdot \prod q\left(X_{(i)}^{[?]}\right) + \frac{T}{\ln(2)} \left[1 + \ln\left(q\left(X_i^{[m_i]}\right)^k \right) \right] \right]. \tag{A.2}$$

(2) Compute the search direction as $\mathbf{d}_i^k = -\mathbf{C}_i^k \cdot (\mathbf{H}_i^k)^{-1}$.

(3) Compute $J_i((q(\mathbf{X}_i) + \alpha_{\text{step}} \cdot \mathbf{d}_i^k), T)$.

(4) Update the probability vector $q(\mathbf{X}_i)^{k+1} = q(\mathbf{X}_i)^k + \alpha_{\text{step}} \cdot \mathbf{d}_i^k$

(5) Update the Hessian $\mathbf{H}_i^{k+1} = \mathbf{H}_i^k + \mathbf{D}_i^k + \mathbf{E}_i^k$, where

$$\mathbf{D}_i^k = \frac{\mathbf{y}_i^k \cdot \left(\mathbf{y}_i^k\right)^T}{\mathbf{y}_i^k \cdot \mathbf{s}_i^k}, \qquad \mathbf{E}_i^k = \frac{\mathbf{C}_i^k \cdot \left(\mathbf{C}_i^k\right)^T}{\mathbf{C}_i^k \cdot \mathbf{d}_i^k}, \tag{A.3}$$

$$\mathbf{s}_i^k = \alpha_{\text{step}} \cdot \mathbf{d}_i^k, \qquad \mathbf{y}_i^k = \mathbf{C}_i^{k+1} - \mathbf{C}_i^k,$$

$$\mathbf{C}^{k+1} = \left[\frac{\partial J_i(q(\mathbf{X}_i), T)^{k+1}}{\partial q\left(X_i^{[1]}\right)^{k+1}} \quad \frac{\partial J_i(q(\mathbf{X}_i), T)^{k+1}}{\partial q\left(X_i^{[2]}\right)^{k+1}} \quad \cdots \quad \frac{\partial J_i(q(\mathbf{X}_i), T)^{k+1}}{\partial q\left(X_i^{[m_i]}\right)^{k+1}} \right] = \left[G\left(\mathbf{Y}_i^{[1]}\right) \cdot \prod q\left(X_{(i)}^{[?]}\right) + \frac{T}{\ln(2)} \left[1 + \ln\left(q\left(X_i^{[1]}\right)^{k+1} \right) \right] \right.$$

$$\left. G\left(\mathbf{Y}_i^{[1]}\right) \cdot \prod q\left(X_{(i)}^{[?]}\right) + \frac{T}{\ln(2)} \left[1 + \ln\left(q\left(X_i^{[2]}\right)^{k+1} \right) \right] \cdots G\left(\mathbf{Y}_i^{[m_i]}\right) \cdot \prod q\left(X_{(i)}^{[?]}\right) + \frac{T}{\ln(2)} \left[1 + \ln\left(q\left(X_i^{[m_i]}\right)^{k+1} \right) \right] \right]. \tag{A.4}$$

(6) Accept the current probability distribution $q(\mathbf{X}_i)^k$ if $k \geq \nu$ or the condition $q(\mathbf{X}_i)^k - q(\mathbf{X}_i)^{k-1} \leq \varepsilon_2$ is true for successive considerable number of iterations, then stop; else update $k = k + 1$ and go to (2).

References

[1] A. J. Kulkarni and K. Tai, "Probability Collectives: a multi-agent approach for solving combinatorial optimization problems," *Applied Soft Computing Journal*, vol. 10, no. 3, pp. 759–771, 2010.

[2] D. H. Wolpert, "Information theory—the bridge connecting bounded rational game theory and statistical physics," in *Complex Engineered Systems*, D. Braha, A. A. Minai, and Y. Bar-Yam, Eds., pp. 262–290, Springer, 2006.

[3] S. R. Bieniawski, *Distributed optimization and flight control using collectives*, PhD dissertation, Stanford University, Stanford, Calif, USA, 2005.

[4] A. J. Kulkarni and K. Tai, "Probability collectives for decentralized, distributed optimization: a collective intelligence approach," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '08)*, pp. 1271–1275, October 2008.

[5] A. J. Kulkarni and K. Tai, "Probability collectives: a decentralized, distributed optimization for multi-agent systems," in *Applications of Soft Computing, Mehnen*, J. Mehnen, M. Koeppen, A. Saad, and A. Tiwari, Eds., pp. 441–450, Springer, 2009.

[6] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[7] M. Vasirani and S. Ossowski, "Collective-based multiagent coordination: a case study," in *Engineering Societies in the Agents World VIII*, vol. 4995 of *Lecture Notes in Computer Science*, pp. 240–253, 2008.

[8] T. Ray, K. Tai, and K. C. Seow, "An evolutionary algorithm for constrained optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 771–777, 2000.

[9] T. Ray, K. Tai, and K. C. Seow, "Multiobjective design coordination by an evolutionary algorithm," *Engineering Optimization*, vol. 33, no. 4, pp. 399–424, 2001.

[10] K. Tai and J. Prasad, "Target-matching test problem for multiobjective topology optimization using genetic algorithms," *Structural and Multidisciplinary Optimization*, vol. 34, no. 4, pp. 333–345, 2007.

[11] D. H. Wolpert and K. Tumer, "An introduction to collective intelligence," Tech. Rep. NASA ARC-IC-99-63, NASA Ames Research Center, 1999.

[12] D. H. Wolpert, C. E. M. Strauss, and D. Rajnarayan, "Advances in distributed optimization using probability collectives," *Advances in Complex Systems*, vol. 9, no. 4, pp. 383–436, 2006.

[13] D. H. Wolpert, N. E. Antoine, S. R. Bieniawski, and I. R. Kroo, "Fleet assignment using collective intelligence," in *Proceedings of the 42nd AIAA Aerospace Science Meeting Exhibit*, 2004.

[14] S. R. Bieniawski, I. M. Kroo, and D. H. Wolpert, "Discrete, continuous, and constrained optimization using collectives," in *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, vol. 5, pp. 3079–3087, September 2004.

[15] T. Basar and G. J. Olsder, *Dynamic Non-Cooperative Game Theory*, Academic Press, New York, NY, USA, 1995.

[16] C. F. Huang, D. H. Wolpert, S. Bieniawski, and C. E. M. Strauss, "A comparative study of probability collectives based multi-agent systems and genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 751–752, June 2005.

[17] S. Bhadra, S. Shakkottai, and P. Gupta, "Min-cost selfish multicast with network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5077–5087, 2006.

[18] Y. Xi and E. M. Yeh, "Distributed algorithms for minimum cost multicast with network coding in wireless networks," in *Proceedings of 4th International Symposium on Modeling and Optimization in Mobile*, Ad Hoc and Wireless Networks, 2006.

[19] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *Proceedings of the 1st International Conference on Wireless Internet (WICON '05)*, pp. 47–54, July 2005.

[20] M. Chatterjee, S. K. Das, and D. Turgut, "On-demand weighted clustering algorithm (WCA) for ad hoc networks," in *Proceedings of 43rd IEEE Global Telecommunication Conference (GLOBECOM '00)*, pp. 1697–1701, 2000.

[21] M. H. Amerimehr, B. K. Khalaj, and P. M. Crespo, "A distributed cross-layer optimization method for multicast in interference-limited multihop wireless networks," *Journal on Wireless Communications and Networking*, vol. 2008, Article ID 702036, 13 pages, 2008.

[22] M. H. A. Mehr and B. H. Khalaj, "A distributed probability collectives optimization method for multicast in CDMA wireless data networks," in *Proceedings of the 4th IEEE International Symposium on Wireless Communication Systems (ISWCS '07)*, pp. 617–621, October 2007.

[23] G. S. Ryder and K. G. Ross, "A probability collectives approach to weighted clustering algorithms for ad hoc networks," in *Proceedings of the 3rd IASTED International Conference on Communications and Computer Networks (CCN '05)*, pp. 94–99, October 2005.

[24] M. Smyrnakis and D. S. Leslie, "Sequentially updated probability collectives," in *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 5774–5779, Shanghai, China, December 2009.

[25] D. Sislak, P. Volf, M. Pechoucek, and N. Suri, "Automated Conflict Resolution Utilizing Probability Collectives Optimizer," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 41, no. 3, pp. 365–375, 2011.

[26] A. J. Kulkarni and K. Tai, "Probability collectives: a distributed optimization approach for constrained problems," in *Proceedings of IEEE World Congress on Computational Intelligence*, pp. 3844–3851, 2010.

[27] A. J. Kulkarni and K. Tai, "Solving constrained optimization problems using probability collectives and a penalty function approach," *International Journal of Computational Intelligence and Applications*. In press.

[28] B. Autry, *University course timetabling with probability collectives*, M.S. thesis, Naval Postgraduate School, Montery, Calif, USA, 2008.

[29] C. F. Huang and B. R. Chang, "Probability collectives multiagent systems: a study of robustness in search," in *Proceedings of the 2nd International Conference on Computational Collective Intelligence—Technology and Applications (ICCCI '10)*, vol. 6422 of *Lecture Notes in Artificial Intelligence*, pp. 334–343, 2010.

[30] P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo, "Adopt: asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, vol. 161, no. 1-2, pp. 149–180, 2005.

[31] D. E. Goldberg and M. P. Samtani, "Engineering optimization via genetic algorithm," in *Proceedings of the 9th Conference on Electronic Computation*, pp. 471–484, 1986.

[32] M. R. Ghasemi, E. Hinton, and R. D. Wood, "Optimization of trusses using genetic algorithms for discrete and continuous variables," *Engineering Computations*, vol. 16, no. 3, pp. 272–301, 1999.

[33] J. S. Moh and D. Y. Chiang, "Improved simulated annealing search for structural optimization," *AIAA journal*, vol. 38, no. 10, pp. 1965–1973, 2000.

[34] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[35] Q. He and L. Wang, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization,"

Applied Mathematics and Computation, vol. 186, no. 2, pp. 1407–1422, 2007.

[36] V. P. Sakthivel, R. Bhuvaneswari, and S. Subramanian, "Design optimization of three-phase energy efficient induction motor using adaptive bacterial foraging algorithm," *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 29, no. 3, pp. 699–726, 2010.

[37] A. Kaveh and S. Talatahari, "An amproved ant colony optimization for constrained engineering design problems," *Computer Aided Engineering and Software*, vol. 27, no. 1, pp. 155–182, 2010.

[38] S. Bansal, A. Mani, and C. Patvardhan, "Is stochastic ranking really better than feasibility rules for constraint handling in evolutionary algorithms?" in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 1564–1567, December 2009.

[39] J. Gao, H. Li, and Y. C. Jiao, "Modified differential evolution for the integer programming problems," in *Proceedings of International Conference on Artificial Intelligence (AICI '09)*, pp. 213–219, November 2009.

[40] P. Wang and X. Tian, "A hybrid DE-SQP algorithm with switching procedure for dynamic optimization," in *Proceedings of Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pp. 2254–2259, December 2009.

[41] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: a critical survey," Tech. Rep., Department of Computer Science, Stanford University, Stanford, Calif, USA, 2003.

[42] L. Buşoniu, R. Babuška, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 38, no. 2, pp. 156–172, 2008.

[43] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[44] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," in *Proceedings of 17th International Conference on Artificial Intelligence*, pp. 1021–1026, 2001.

[45] V. Sindhwani, S. S. Keerthi, and O. Chapelle, "Deterministic annealing for semi-supervised kernel machines," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 841–848, June 2006.

[46] A. V. Rao, D. J. Miller, K. Rose, and A. Gersho, "A deterministic annealing approach for parsimonious design of piecewise regression models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 2, pp. 159–173, 1999.

[47] A. V. Rao, D. Miller, K. Rose, and A. Gersho, "Mixture of experts regression modeling by deterministic annealing," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2811–2820, 1997.

[48] R. Czabanski, "Deterministic annealing integrated with intensive learning in neuro-fuzzy systems," in *Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing*, vol. 4029 of *Lecture Notes in Artificial Intelligence*, pp. 220–229, 2006.

[49] http://hyperphysics.phy-astr.gsu.edu/hbase/thermo/helmholtz.html, 2010.

[50] J. S. Arora, *Introduction to Optimum Design*, Elsevier Academic Press, 2004.

[51] G. N. Vanderplaat, *Numerical Optimization Techniques for Engineering Design*, Mcgraw-Hill, 1984.

[52] D. F. Zhang and A. S. Deng, "An effective hybrid algorithm for the problem of packing circles into a larger containing circle," *Computers and Operations Research*, vol. 32, no. 8, pp. 1941–1951, 2005.

[53] V. E. Theodoracatos and J. L. Grimsley, "The optimal packing of arbitrarily-shaped polygons using simulated annealing and polynomial-time cooling schedules," *Computer Methods in Applied Mechanics and Engineering*, vol. 125, no. 1–4, pp. 53–70, 1995.

[54] J. Liu, S. Xue, Z. Liu, and D. Xu, "An improved energy landscape paving algorithm for the problem of packing circles into a larger containing circle," *Computers and Industrial Engineering*, vol. 57, no. 3, pp. 1144–1149, 2009.

[55] I. Castillo, F. J. Kampas, and J. D. Pintér, "Solving circle packing problems by global optimization: numerical results and industrial applications," *European Journal of Operational Research*, vol. 191, no. 3, pp. 786–802, 2008.

[56] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[57] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *Journal of the Association for Computing Machinery*, vol. 1, no. 32, pp. 130–136, 1985.

[58] H. Wang, W. Huang, Q. Zhang, and D. Xu, "An improved algorithm for the packing of unequal circles within a larger containing circle," *European Journal of Operational Research*, vol. 141, no. 2, pp. 440–453, 2002.

[59] P. G. Szabo, M. C. Markot, and T. Csendes, "Global optimization in geometry—circle packing into the square," in *Essays and Surveys in Global Optimization*, P. Audet, P. Hansen, and G. Savard, Eds., pp. 233–265, Kluwer, 2005.

[60] K. J. Nurmela and P. R. J. Östergård, "Packing up to 50 equal circles in a square," *Discrete and Computational Geometry*, vol. 18, no. 1, pp. 111–120, 1997.

[61] K. J. Nurmela and P. R. J. Östergård, "More optimal packings of equal circles in a square," *Discrete and Computational Geometry*, vol. 22, no. 3, pp. 439–457, 1999.

[62] R. L. Graham and B. D. Lubachevsky, "Repeated patterns of dense packings of equal disks in a square," *Electronic Journal of Combinatorics*, vol. 3, no. 1, article R16, pp. 1–17, 1996.

[63] P. G. Szabo, T. Csendes, L. G. Casado, and I. Garcia, "Equal circles packing in a square I—problem setting and bounds for optimal solutions," in *Optimization Theory: Recent Developments from Matrahaza*, F. Giannessi, P. Pardalos, and T. Rapcsak, Eds., pp. 191–206, Kluwer, 2001.

[64] C. de Groot, R. Peikert, and D. Wurtz, "The optimal packing of ten equal circles in a square," PS Research Report 90-12, ETH, Zurich, Switzerland, 1990.

[65] M. Goldberg, "The packing of equal circles in a square," *Mathematics Magazine*, vol. 43, pp. 24–30, 1970.

[66] M. Mollard and C. Payan, "Some progress in the packing of equal circles in a square," *Discrete Mathematics*, vol. 84, no. 3, pp. 303–307, 1990.

[67] J. Schaer, "On the packing of ten equal circles in a square," *Mathematics Magazine*, vol. 44, pp. 139–140, 1971.

[68] K. Schluter, "Kreispackung in quadraten," *Elemente der Mathematics*, vol. 34, pp. 12–14, 1979.

[69] G. Valette, "A better packing of ten equal circles in a square," *Discrete Mathematics*, vol. 76, no. 1, pp. 57–59, 1989.

[70] D. W. Boll, J. Donovan, R. L. Graham, and B. D. Lubachevsky, "Improving dense packings of equal disks in a square," *Electronic Journal of Combinatorics*, vol. 7, no. 1, article R46, pp. 1–9, 2000.

[71] B. D. Lubachevsky and R. L. Graham, "Curved hexagonal packings of equal disks in a circle," *Discrete and Computational Geometry*, vol. 18, no. 2, pp. 179–194, 1997.

[72] P. G. Szabo and E. Specht, "Packing up to 200 equal circles in a square," in *Models and Algorithms for Global Optimization*, A. Torn and J. Zilinskas, Eds., pp. 141–156, 2007.

[73] N. Mladenovic, F. Plastria, and D. Urosevi, "Formulation space search for circle packing problems," in *Proceedings of the Society of Legal Scholars Annual Conference*, Lecture Notes in Computer Science, pp. 212–216, September 2007.

[74] N. Mladenović, F. Plastria, and D. Urošević, "Reformulation descent applied to circle packing problems," *Computers and Operations Research*, vol. 32, no. 9, pp. 2419–2434, 2005.

[75] W. Huang and M. Chen, "Note on: an improved algorithm for the packing of unequal circles within a larger containing circle," *Computers and Industrial Engineering*, vol. 50, no. 3, pp. 338–344, 2006.

[76] J. Liu, D. Xu, Y. Yao, and Y. Zheng, "Energy landscape paving algorithm for solving circles packing problem," in *Proceedings of the International Conference on Computational Intelligence and Natural Computing (CINC '09)*, pp. 107–110, June 2009.

[77] J. Liu, Y. Yao, Y. Zheng, H. Geng, and G. Zhou, "An effective hybrid algorithm for the circles and spheres packing problems," in *Proceedings of the 3rd Annual International Conference on Combinatorial Optimization and Applications (COCOA '09*, vol. 5573 of *Lecture Notes in Computer Science*, pp. 135–144, June 2009.

[78] Y. G. Stoyan and G. N. Yaskov, "Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints," *International Transactions in Operational Research*, vol. 5, pp. 45–57, 1998.

[79] Y. G. Stoyan and G. Yas'kov, "A mathematical model and a solution method for the problem of placing various-sized circles into a strip," *European Journal of Operational Research*, vol. 156, no. 3, pp. 590–600, 2004.

[80] J. A. George, "Multiple container packing: a case study of pipe packing," *Journal of the Operational Research Society*, vol. 47, no. 9, pp. 1098–1109, 1996.

[81] J. A. George, J. M. George, and B. W. Lamar, "Packing different-sized circles into a rectangular container," *European Journal of Operational Research*, vol. 84, no. 3, pp. 693–712, 1995.

[82] M. Hifi and R. M'Hallah, "Approximate algorithms for constrained circular cutting problems," *Computers and Operations Research*, vol. 31, no. 5, pp. 675–694, 2004.

[83] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[84] N. F. Wang and K. Tai, "Target matching problems and an adaptive constraint strategy for multiobjective design optimization using genetic algorithms," *Computers and Structures*, vol. 88, no. 19-20, pp. 1064–1076, 2010.

[85] E. Y. K. Ng, K. Tai, W. K. Ng, and R. U. Acharya, "Optimization of the pharmacokinetic simulation models for the nanoparticles-in-nanoshapes hybrid drug delivery system using heat diffusion analogy," in *Distributed Diagnosis and Home Healthcare*, U. Rajendra Acharya, T. Tamura, E. Y. K. Ng, C. M. Lim, and J. S. Suri, Eds., pp. 211–230, American Scientific Publishers, 2010.