Chapter 2
Probability Collectives: A Distributed Optimization Approach

An emerging Artificial Intelligence tool in the framework of Collective Intelligence (COIN) for modeling and controlling distributed Multi-agent System (MAS) referred to as Probability Collectives (PC) was first proposed by Dr. David Wolpert in 1999 in a technical report presented to NASA [1]. It is inspired from a sociophysics viewpoint with deep connections to Game Theory, Statistical Physics, and Optimization [2–4]. 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 [2, 5, 6]. 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 [3]. 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 [5]. The concept is successfully formalized and implemented through the PC methodology. The approach works on probability distributions, directly incorporating uncertainty and is based on prior knowledge of the actions/strategies of all the other agents. The available literature on PC is discussed in the following section.

2.1 Background of PC

The approach of PC has been tested proving its variegated application domains and comparing its performance with other algorithms. The literature is discussed in the following few paragraphs.

Huang and Chang [7] demonstrated the search process of PC methodology is more robust as compared with the GA optimizing the multimodal function such as the Schaffer's function. In addition, it was also demonstrated that PC also

A.J. Kulkarni et al., *Probability Collectives*, Intelligent Systems Reference Library 86, DOI 10.1007/978-3-319-16000-9_2

outperformed GA in the rate of descent, trapping in false minima and long term optimization when tested and compared for the multimodality, nonlinearity and non-separability in solving several other benchmark problems such as Rosenbrock function, Ackley Path function and Michalewicz Epistatic function. Furthermore, Huang et al. [8] also discussed fundamental differences between GA and PC. 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 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 when optimizing an associated Homotopy function. It also directly incorporates uncertainty due to both imperfect sampling and the stochastic independence of the agents' actions. Furthermore, PC outperformed the Heuristic Adaptive GA (HAGA) [9], Heuristic ACO (HACO) [10] and Heuristic PSO (HPSO) [11] in stability and robustness solving the air combat decision-making for coordinated multiple target assignment problem [12]. The above comparison indicated that PC can potentially be applied to wide application areas.

Vasirani and Ossowski [13] solved the 8-queens problem and underscored the superiority of the decentralized PC architecture over a centralized one. They also demonstrated that both the 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) [14]. 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.

Wolpert et al. [5] successfully applied the PC methodology for 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 by 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. Mohammad and Babak [15] as well as Ryder and Ross [16] successfully applied the approach of PC solving combinatorial optimization problems such as the joint optimization of the routing and resource allocation in wireless networks.

Furthermore, Wolpert et al. [6] also tested the potential of PC in mechanical design domain for optimizing the cross-sections of individual bars and segments of a 10 bar truss. The problem was formulated as a discrete problem and the solution was feasible but was worse than those obtained by other methods [17–19]. In

addition, Autry [20] also tested the PC methodology on the discrete problem of university course scheduling; however, the implementation failed to generate any feasible solution. It is important to note that although the problems in [5, 6, 20] are constrained problems, the constraints were not explicitly treated or incorporated in the PC algorithms.

Recently, Sislak et al. [21] proposed decentralized and semi-centralized approaches for solving an autonomous conflict resolution problem for cooperating airplanes to avoid the mid-air 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 second approach which is a semi-centralized 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. In both the approaches 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 and collision constraints were molded into special penalty functions which were further integrated into the objective function using suitable balancing factors. Similar to the weighted sum multi-objective approach [22, 23], the balancing factors were assigned based on the importance of the corresponding constraints. Smyrnakis and Leslie [24] proposed a Sequentially Updated PC (SPC) which 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.

2.2 Conceptual Framework of PC

PC treats the variables in an optimization problem as individual self interested learning agents/players of a game being played iteratively. 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 also referred to as Nash equilibrium (See Sect. 2.2.2 for proof and details) 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, i.e. 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. 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. Due to its analogy to Helmholtz free energy, the approach of Deterministic Annealing (DA) is applied in minimizing the Homotopy function (See Appendix A for details). 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 (See Appendix B for details) or Broyden–Fletcher–Goldfarb–Shanno (BFGS) Scheme (See Appendix C for details).

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. The solution is accepted only if the associated system objective and corresponding strategy values are not worse than the previous iterations solution. In this way, the algorithm continues until convergence by selecting the samples from the neighborhood of the recent favorable strategies.

In some of the applications, the agents are also needed to provide the knowledge of the inter-agent-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 describes the detailed formulation of the PC procedure as an unconstrained approach [25–28].

2.2.1 Formulation of Unconstrained PC

Consider a general unconstrained problem (in minimization sense) comprising N variables and objective function G. In the context of PC, the variables of the problem are considered as computational agents/players of a social game being played iteratively.

Each variable, i.e. each agent i is given a predefined sampling interval referred to as $\Psi_i \in \left[\Psi_i^{lower}, \Psi_i^{upper}\right]$. 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

$$\mathrm{X}_i = \left\{X_i^{[1]}, X_i^{[2]}, X_i^{[3]}, \ldots, X_i^{[m_i]}\right\}, \quad i = 1, 2, \ldots, N \tag{2.1}$$

Every agent is assumed to have an equal number of strategies, i.e. $m_1 = m_2 = \ldots = m_i = \ldots = m_{N-1} = m_N$. The procedure of modified PC theory is explained below in detail with the algorithm flowchart in Fig. 2.1.

The procedure begins with the initialization of the parameters such as sampling interval Ψ_i for each agent i, temperature $T \gg 0$ or $T = T_{initial}$ or $T \to \infty$ (simply high enough), the temperature step size $\alpha_T (0 < \alpha_T \leq 1)$, convergence parameter $\varepsilon = 0.0001$, interval factor λ_{down} and algorithm iteration counter $n = 1$. The value of α_T, m_i and λ_{down} can be chosen based on preliminary trials of the algorithm.

Step 1 Agent i selects its first strategy $X_i^{[1]}$. Agent i then samples randomly from other agents' strategies as well. These are random guesses by agent i about which strategies have been chosen by the other agents. All these form a 'combined strategy set' $\mathbf{Y}_i^{[1]}$ given by

$$\mathrm{Y}_i^{[1]} = \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[1]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \tag{2.2}$$

The superscript [?] indicates that it is a 'random guess' and not known in advance. In addition, agent i forms one combined strategy set for each of the remaining strategies of its strategy set $\mathbf{X}_i$ (i.e. $r = 2, 3, \ldots, m_i$), as shown below.

$$\begin{aligned}
\mathbf{Y}_i^{[2]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[2]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \\
\mathbf{Y}_i^{[3]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[3]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \\
&\vdots \\
\mathbf{Y}_i^{[r]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[r]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \\
&\vdots \\
\mathbf{Y}_i^{[m_i]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[m_i]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\}
\end{aligned} \tag{2.3}$$

Similarly, all the remaining agents form their combined strategy sets. Furthermore, every agent i computes m_i associated objective function values as follows:

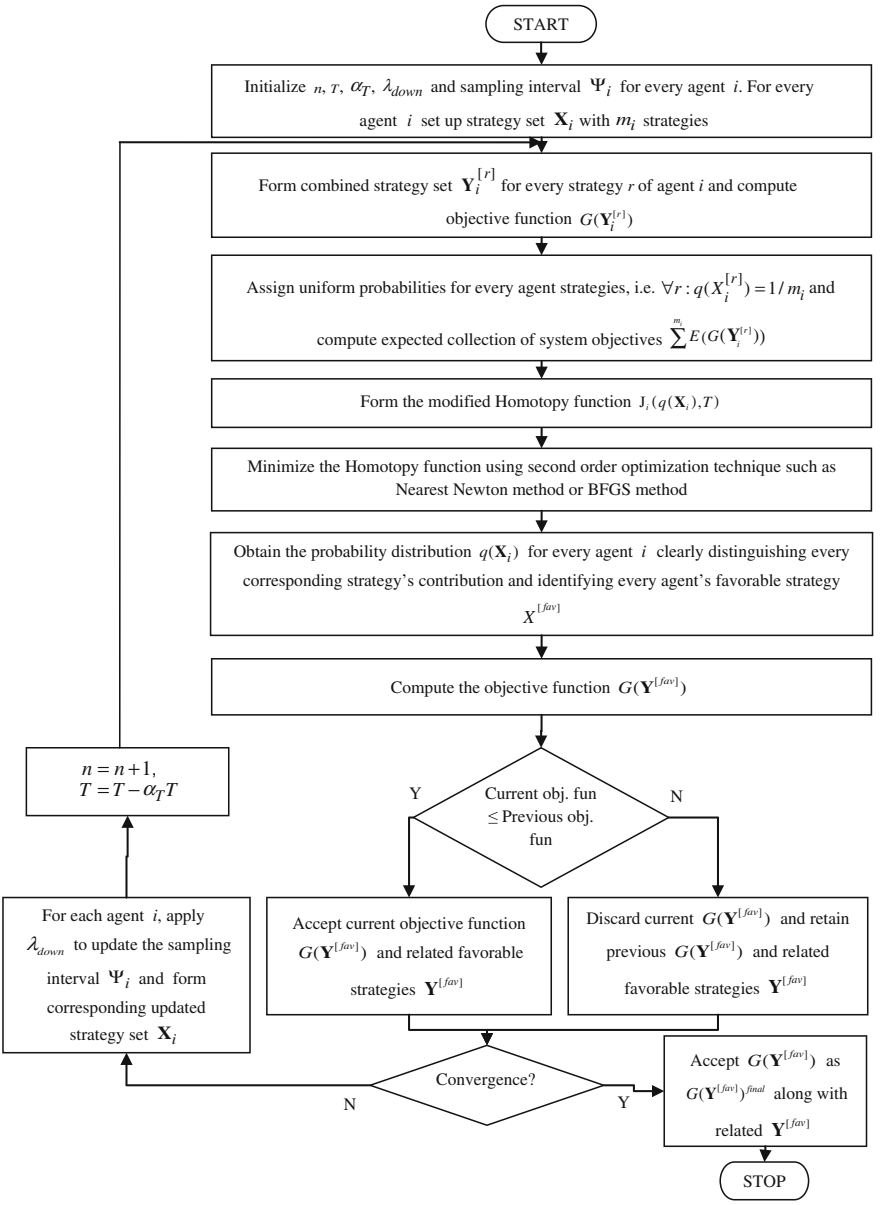

Fig. 2.1 Unconstrained PC algorithm flowchart

$$\left[G\left(\mathbf{Y}_i^{[1]}\right), G\left(\mathbf{Y}_i^{[2]}\right), \ldots, G\left(\mathbf{Y}_i^{[r]}\right), \ldots, G\left(\mathbf{Y}_i^{[m_i]}\right)\right] \tag{2.4}$$

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, i.e. $\sum_{r=1}^{m_i} G(\mathbf{Y}_i^{[r]})$, hereafter referred to as the collection of system objectives. The combined strategy sets, associated objective functions and the collection of system objectives for all the N agents are as follows:

$$\left.\begin{aligned}
\mathbf{Y}_1^{[1]} &= \left\{X_1^{[1]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Rightarrow G\left(\mathbf{Y}_1^{[1]}\right) \\
\mathbf{Y}_1^{[2]} &= \left\{X_1^{[2]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Rightarrow G\left(\mathbf{Y}_1^{[2]}\right) \\
&\vdots \\
\mathbf{Y}_1^{[r]} &= \left\{X_1^{[r]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Rightarrow G\left(\mathbf{Y}_1^{[r]}\right) \\
&\vdots \\
\mathbf{Y}_1^{[m_i]} &= \left\{X_1^{[m_i]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[?]}\right\} \Rightarrow G\left(\mathbf{Y}_1^{[m_i]}\right)
\end{aligned}\right\} \Rightarrow \sum_{r=1}^{m_i} G(\mathbf{Y}_1^{[r]})$$

$$\vdots$$

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

$$\vdots$$

$$\left.\begin{aligned}
\mathbf{Y}_N^{[1]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[1]}\right\} \Rightarrow G\left(\mathbf{Y}_N^{[1]}\right) \\
\mathbf{Y}_N^{[2]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[2]}\right\} \Rightarrow G\left(\mathbf{Y}_N^{[2]}\right) \\
&\vdots \\
\mathbf{Y}_N^{[r]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[r]}\right\} \Rightarrow G\left(\mathbf{Y}_N^{[r]}\right) \\
&\vdots \\
\mathbf{Y}_N^{[m_i]} &= \left\{X_1^{[?]}, X_2^{[?]}, \ldots, X_i^{[?]}, \ldots, X_{N-1}^{[?]}, X_N^{[m_i]}\right\} \Rightarrow G\left(\mathbf{Y}_N^{[m_i]}\right)
\end{aligned}\right\} \Rightarrow \sum_{r=1}^{m_i} G(\mathbf{Y}_N^{[r]}) \tag{2.5}$$

Step 2 For every agent i, 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. 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 E. Such a method is referred to as the Homotopy method. The function E 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. The deformed function can also be referred to as Homotopy function J parameterized by computational temperature T represented as follows:

$$J_i(q(\mathbf{X}_i), T) = \sum_{r=1}^{m_i} G(\mathbf{Y}_i^{[r]}) - TE \ , \quad T \in [0, \infty) \tag{2.6}$$

(a) Agent i assigns uniform probabilities to its strategies. As an illustration, the uniform probability distribution for agent i may look like that shown in Fig. 2.2 for a case where there are 10 strategies, i.e. $m_i = 10$. 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 $\sum_{r=1}^{m_i} G(\mathbf{Y}_i^{[r]})$. 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

$$q(X_i^{[r]}) = 1/m_i \ , \quad r = 1, 2, \ldots, m_i \tag{2.7}$$

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

$$E\left(G(\mathrm{Y}_i^{[r]})\right) = G(\mathrm{Y}_i^{[r]})\, q(X_i^{[r]}) \prod_{(i)} q(X_{(i)}^{[?]}) \tag{2.8}$$

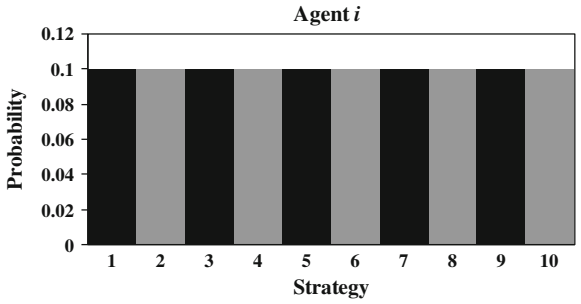

Fig. 2.2 Uniform probability distribution of agent i

where (i) represents every agent other than i. Every agent i then computes the expected collection of system objectives denoted by $\sum_{r=1}^{m_i} E\Big(G(\mathbf{Y}_i^{[r]})\Big)$. The expected system objectives and the associated expected collection of system objective for all the N agents are as follows:

$$\left.\begin{array}{l} G\Big(\mathrm{Y}_1^{[1]}\Big)\, q\Big(X_1^{[1]}\Big) \prod\limits_{(1)} q\Big(X_{(1)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_1^{[1]}\Big)\Big) \\ \vdots \\ G\Big(\mathrm{Y}_1^{[r]}\Big)\, q\Big(X_1^{[r]}\Big) \prod\limits_{(1)} q\Big(X_{(1)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_1^{[r]}\Big)\Big) \\ \vdots \\ G\Big(\mathrm{Y}_1^{[m_1]}\Big)\, q\Big(X_1^{[m_1]}\Big) \prod\limits_{(1)} q\Big(X_{(1)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_1^{[m_1]}\Big)\Big) \end{array}\right\} \Rightarrow \sum_{r=1}^{m_1} E\Big(G(\mathbf{Y}_1^{[r]})\Big)$$

$$\vdots$$

$$\left.\begin{array}{l} G\Big(\mathrm{Y}_i^{[1]}\Big)\, q\Big(X_i^{[1]}\Big) \prod\limits_{(i)} q\Big(X_{(i)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_i^{[1]}\Big)\Big) \\ \vdots \\ G\Big(\mathrm{Y}_i^{[r]}\Big)\, q\Big(X_i^{[r]}\Big) \prod\limits_{(i)} q\Big(X_{(i)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_i^{[r]}\Big)\Big) \\ \vdots \\ G\Big(\mathrm{Y}_i^{[m_i]}\Big)\, q\Big(X_i^{[m_i]}\Big) \prod\limits_{(i)} q\Big(X_{(i)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_i^{[m_i]}\Big)\Big) \end{array}\right\} \Rightarrow \sum_{r=i}^{m_i} E\Big(G(\mathbf{Y}_i^{[r]})\Big)$$

$$\vdots$$

$$\left.\begin{array}{l} G\Big(\mathrm{Y}_N^{[1]}\Big)\, q\Big(X_N^{[1]}\Big) \prod\limits_{(N)} q\Big(X_{(N)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_N^{[1]}\Big)\Big) \\ \vdots \\ G\Big(\mathrm{Y}_N^{[r]}\Big)\, q\Big(X_N^{[r]}\Big) \prod\limits_{(N)} q\Big(X_{(N)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_N^{[r]}\Big)\Big) \\ \vdots \\ G\Big(\mathrm{Y}_N^{[m_N]}\Big)\, q\Big(X_N^{[m_N]}\Big) \prod\limits_{(N)} q\Big(X_{(N)}^{[?]}\Big) = E\Big(G\Big(\mathbf{Y}_N^{[m_N]}\Big)\Big) \end{array}\right\} \Rightarrow \sum_{r=N}^{m_N} E\Big(G(\mathbf{Y}_N^{[r]})\Big) \tag{2.9}$$

It 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) Furthermore, a suitable function for E is chosen. The general choice is to use the entropy function $S_i = -\sum_{r=1}^{m_i} q\left(X_i^{[r]}\right) \log_2 q\left(X_i^{[r]}\right)$. Thus the Homotopy function to be minimized by each agent i in Eq. (2.6) is developed as follows:

$$\begin{aligned} J_i(q(\mathbf{X}_i), T) &= \sum_{r=1}^{m_i} E\left(G(\mathbf{Y}_i^{[r]})\right) - T\, S_i \\ &= \sum_{r=1}^{m_i} \left(G(\mathbf{Y}_i^{[r]})\, q(X_i^{[r]}) \prod_{(i)} q(X_{(i)}^{[?]}) \right) - T\left(-\sum_{r=1}^{m_i} q(X_i^{[r]}) \log_2 q(X_i^{[r]}) \right) \\ &= G(\mathbf{Y}_i^{[1]})\, q(X_i^{[1]}) \prod_{(i)} q(X_{(i)}^{[?]}) + G(\mathbf{Y}_i^{[2]})\, q(X_i^{[2]}) \prod_{(i)} q(X_{(i)}^{[?]}) + \cdots \\ &\quad \cdots + G(\mathbf{Y}_i^{[m_i-1]})\, q(X_i^{[m_i-1]}) \prod_{(i)} q(X_{(i)}^{[?]}) + G(\mathbf{Y}_i^{[m_i]})\, q(X_i^{[m_i]}) \prod_{(i)} q(X_{(i)}^{[?]}) \\ &\quad - T\left(-\sum_{r=1}^{m_i} q(X_i^{[r]})\, \log_2 q(X_i^{[r]}) \right) \end{aligned} \tag{2.10}$$

where $T \in [0, \infty)$.

Step 3 The approach of Deterministic Annealing (DA) discussed in Appendix A is applied to minimize the Homotopy function in Eq. (2.10). The motivation behind this is its analogy to the Helmholtz free energy. Starting from $T \gg 0$ or $T = T_{initial}$ or $T \to \infty$ (simply high enough), with every temperature drop, the minimization of the Homotopy function in Eq. (2.10) can be carried out using suitable optimization technique such as Nearest Newton Descent Scheme as well as a suitable second order optimization approach such as BFGS scheme. The Nearest Newton Descent Scheme and the BFGS scheme minimizing Eq. (2.10) are discussed in Appendix B and Appendix C, respectively

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 $\sum_{r=1}^{m_i} E\left(G(\mathbf{Y}_i^{[r]})\right)$. In other words, for each agent i, if a particular 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,

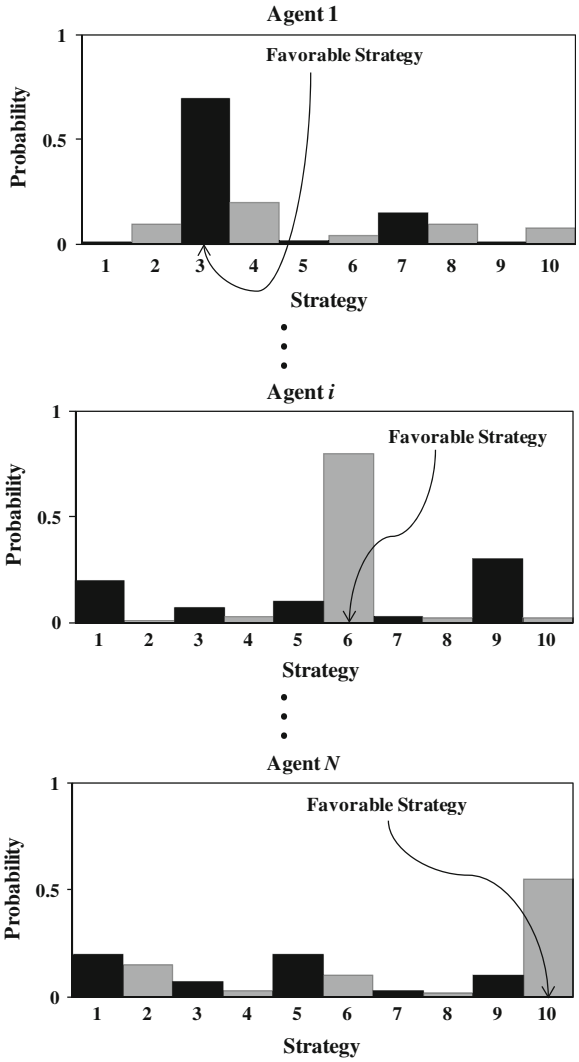

Fig. 2.3 Probability distribution of agent $i = 1, 2, \ldots, N$

and so strategy r is distinguished from the other strategies. Such a strategy is referred to as a favorable strategy $X_i^{[fav]}$. As an illustration, the converged probability distribution for every agent i may look like that shown in Fig. 2.3 for a case where there are 10 strategies, i.e. $m_i = 10$.

The above optimization process when converges, the Nash equilibrium is achieved at the particular temperature T. The concept of Nash equilibrium is discussed in Sect. 2.2.2.

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

$$\mathbf{Y}^{[fav]} = \left\{X_1^{[fav]}, X_2^{[fav]}, \ldots, X_{N-1}^{[fav]}, X_N^{[fav]}\right\} \tag{2.11}$$

Step 5 If the current system objective $G(\mathbf{Y}^{[fav]})$ is not worse than that from the previous iteration solution, accept the current system objective $G(\mathbf{Y}^{[fav]})$ and corresponding $\mathbf{Y}^{[fav]}$ as current solution and continue to *step 6*, else discard current system objective $G(\mathbf{Y}^{[fav]})$ and corresponding $\mathbf{Y}^{[fav]}$, and retain the previous iteration solution and continue to *step 6.*

Step 6 If either of the two criteria listed below is valid, accept the current system objective $G(\mathbf{Y}^{[fav]})$ and corresponding $\mathbf{Y}^{[fav]}$ as the final solution referred to as $G(\mathbf{Y}^{[fav],final})$ and $\mathbf{Y}^{[fav],final} = \left\{X_1^{[fav],final}, X_2^{[fav],final}, \ldots, X_{N-1}^{[fav],final}, X_N^{[fav],final}\right\}$, respectively and stop, else continue to *step 7.*

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

(b) If there is no significant change in the system objectives for successive considerable number of iterations (i.e. $\left\|G(\mathbf{Y}^{[fav],n}) - G(\mathbf{Y}^{[fav],n-1})\right\| \leq \varepsilon$).

Step 7 Every agent shrinks its sampling interval as follows:

$$\Psi_i \in \left[\left(X_i^{[fav]} - \lambda_{down}\left\|\Psi_i^{upper} - \Psi_i^{lower}\right\|\right), \left(X_i^{[fav]} + \lambda_{down}\left\|\Psi_i^{upper} - \Psi_i^{lower}\right\|\right)\right],\ 0 < \lambda_{down} \leq 1 \tag{2.12}$$

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

Each agent i then samples m_i strategies from within the updated sampling interval Ψ_i and forms the corresponding updated strategy set X_i represented as follows.

$$X_i = \left\{X_i^{[1]}, X_i^{[2]}, X_i^{[3]}, \ldots, X_i^{[m_i]}\right\}\ ,\quad i = 1, 2, \ldots, N \tag{2.13}$$

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

On convergence, the above process reaches the global Nash equilibrium. The concept and detailed formulation of the Nash equilibrium achieved is described in the following section.

2.2.2 Nash Equilibrium in PC

To achieve, a Nash equilibrium, every agent in a MAS should have the properties of Rationality and Convergence [29–32]. 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 i.e. 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, i.e. 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. For example, as detailed in the PC algorithm discussed in Sect. 2.2.1, there is a fixed framework by which the agents select their own strategies and guess the other agents' strategies.

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 computationally prohibitive 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 $X_i^{[fav],n}$ by guessing the possible strategies of the other agents. This information about its favorable strategy $X_i^{[fav],n}$ is made known to all the other agents as well. In addition, the corresponding global knowledge such as system objective value $G(\mathbf{Y}^{[fav],n}) = G\left(X_1^{[fav],n}, X_2^{[fav],n}, X_3^{[fav],n}, \ldots, X_{N-1}^{[fav],n}, X_N^{[fav],n}\right)$ 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 $G(\mathbf{Y}^{[fav],n})$. Mathematically the Nash equilibrium solution at any iteration of the PC algorithm can be represented as follows:

$$\begin{aligned}
G\left(X_1^{[fav],n}, X_2^{[fav],n}, \ldots, X_{N-1}^{[fav],n}, X_N^{[fav],n}\right) &\leq G\left(X_1^{(fav),n}, X_2^{[fav],n}, \ldots, X_{N-1}^{[fav],n}, X_N^{[fav],n}\right) \\
G\left(X_1^{[fav],n}, X_2^{[fav],n}, \ldots, X_{N-1}^{[fav],n}, X_N^{[fav],n}\right) &\leq G\left(X_1^{[fav],n}, X_2^{(fav),n}, \ldots, X_{N-1}^{[fav],n}, X_N^{[fav],n}\right) \\
&\vdots \\
G\left(X_1^{[fav],n}, X_2^{[fav],n}, \ldots, X_{N-1}^{[fav],n}, X_N^{[fav],n}\right) &\leq G\left(X_1^{[fav],n}, X_2^{[fav],n}, \ldots, X_{N-1}^{[fav],n}, X_N^{(fav),n}\right)
\end{aligned} \tag{2.14}$$

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

Furthermore, from this current Nash equilibrium point with system objective $G(\mathbf{Y}^{[fav],n})$, the algorithm progresses to the next Nash equilibrium point with better system objective $G(\mathbf{Y}^{[fav],n+1})$, i.e. $G(\mathbf{Y}^{[fav],n}) \geq G(\mathbf{Y}^{[fav],n+1})$. 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 $G(\mathbf{Y}^{[fav],n})$, i.e. 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 $G(\mathbf{Y}^{[fav],final})$. As a general case, this progress can be represented as follows:

$$G(\mathbf{Y}^{[fav],1}) \geq G(\mathbf{Y}^{[fav],2}) \geq \cdots \geq G(\mathbf{Y}^{[fav],n}) \geq G(\mathbf{Y}^{[fav],n+1}) \geq \cdots \geq G(\mathbf{Y}^{[fav],final}).$$

2.3 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, etc. 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. This is explicitly demonstrated in Chap. 6.
2. It is robust in the sense that the cost function (global/system objective) can be irregular or noisy, i.e. it can accommodate noisy and poorly modeled problems.
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. This is explicitly demonstrated in Chap. 5.
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, i.e. peaky distribution provides the best choice of action that can optimize the global goal. This is explicitly demonstrated in Sect. 2.2.1.
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. This is explicitly discussed in Appendix A and also evident when solved a variety of functions using Penalty Function approach in Chap. 4.
6. It can successfully avoid the tragedy of commons, skipping the local minima and further reach the true global minima. This is explicitly demonstrated in Chap. 5.
7. The computational and communication load is significantly less and equally distributed among all the agents.
8. It can efficiently handle problems with moderately large number of variables. This is demonstrated in Chaps. 6 and 7.

With PC solving optimization problems as a MAS, it is worth discussing some of its characteristics to compare the similarities and differences with Multi-Agent Reinforcement Learning (MARL) methods.

9. Most MARL methods such as fully cooperative, fully competitive and mixed (neither cooperative nor competitive) are based on Game Theory, Optimization and Evolutionary Computation. 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. On the other hand, being a distributed optimization technique 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, etc. This social aspect makes PC a cooperative approach.
10. 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.
11. 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.

2.4 Modified PC Approach Versus Original PC Approach

It is worth to mention some of the key differences of the PC methodology presented here and the original PC approach proposed by Dr. David Wolpert [1–4]

1. In the PC approach presented here, 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.

2. 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.
3. Moreover, the coordination among the agents representing the variables in the system is achieved based on the partial small bit of information. In other words, 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.

2.5 Validation of the Unconstrained PC

There are a number of benchmark test functions for contemporary optimization algorithms like GAs and evolutionary computation. The Rosenbrock function (see Fig. 2.4) is an example of a nonlinear function having strongly coupled variables and is a real challenge for any optimization algorithm because of its slow convergence for most optimization methods [33, 34]. The Rosenbrock function with N number of variables is given by

$$G = \sum_{i=1}^{N-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2 \right] \tag{2.15}$$

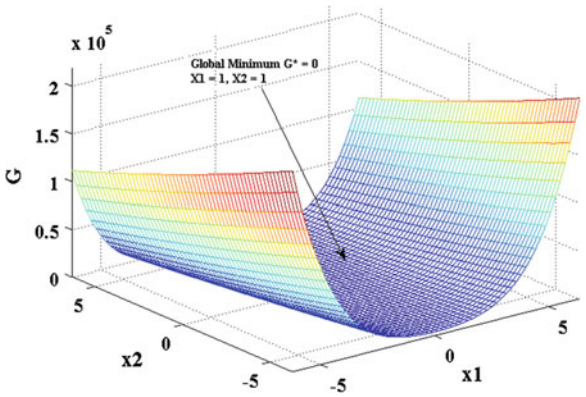

Fig. 2.4 Rosenbrock function

$$\Psi_i \in \left[\Psi_i^{lower}, \Psi_i^{upper}\right], \quad i = 1, 2, \ldots, N$$

and the global minimum is at, $x_i = 1, \forall i \ i = 1, 2, \ldots, N$, where $G^* = 0$. A difficulty arises from the fact that the optimum is located in a deep and narrow parabolic valley with a flat bottom. Also, gradient based methods may have to spend a large number of iterations to reach the global minimum.

In the context of PC, every function variable x_i , $i = 1, 2, \ldots, N$ was represented as an autonomous agent. These agents competed with one another to optimize their individual values and ultimately the entire function value (i.e. global objective G). The Rosenbrock function with 5 variables/agents ($N = 5$) was solved here in which each agent randomly selected the strategies from within a predefined sampling interval Ψ_i. Although the optimal value of each and every agent x_i is 1, the allowable sampling interval Ψ_i for each agent was intentionally assigned to be different (as shown in Table 2.1). The procedure explained in Sect. 2.2.1 was followed to reach convergence. For all the runs, the temperature step size α_T, the interval factor λ_{down} and sample size m_i for each agent chosen were 0.9, 0.1 and 42, respectively. It is worth to mention that the temperature T was reduced on completion of every 10 iterations.

The PC algorithm discussed in Sect. 2.2.1 was coded in MATLAB 7.4.0 (R2007A) on Windows platform using a Pentium 4, 3 GHz processor speed and 512 MB RAM. The results of 5 runs are shown in Table 2.1. The convergence plot for run 5 is shown in Fig. 2.5. For run 5, the value of the function G and associated variable values at iteration 113 was accepted as the final solution as there was no change in the solution for a considerable number of iterations. The variations in the function value and the number of function evaluations among the runs are due to the randomness in selection of the strategies.

A number of researchers have solved the Rosenbrock function using various algorithms. Table 2.2 summarizes the results obtained by these various algorithms:

Table 2.1 Performance using PC approach

Agents or (variables)	Strategy values selected with maximum probability					
	Run 1	Run 2	Run 3	Run 4	Run 5	Range of values (Ψ_i)
Agent 1	1	0.9999	1.0002	1.0001	0.9997	−1.0 to 1.0
Agent 2	1	0.9998	1.0001	1.0001	0.9994	−5.0 to 5.0
Agent 3	1.0001	0.9998	1	0.9999	0.9986	−3.0 to 3.0
Agent 4	0.9998	0.9998	0.9998	0.9995	0.9967	−3.0 to 8.0
Agent 5	0.9998	0.9999	0.9998	0.9992	0.9937	1.0 to 10.0
Fun. value G	2×10^{-5}	1×10^{-5}	2×10^{-5}	2×10^{-5}	5×10^{-5}	
Fun. evaluations	288,100	223,600	359,050	204,750	249,500	

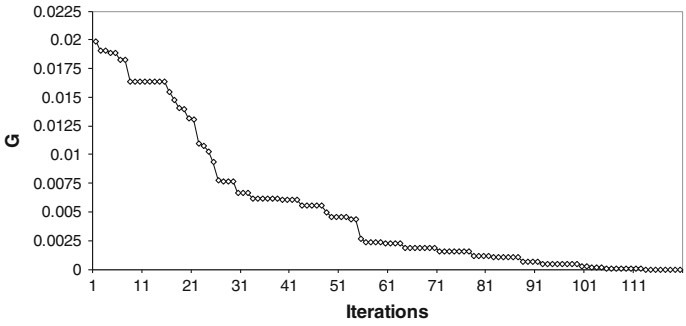

Fig. 2.5 Convergence plot for run 5

Chaos Genetic Algorithm (CGA) [33], Punctuated Anytime Learning (PAL) system [34], Modified Differential Evolution (Modified DE) proposed in [35] and Loosely Coupled GA (LCGA) implemented in [36] are some of the algorithms demonstrated on the Rosenbrock function.

Within each of the results shown in Table 2.2, every variable was assigned an identical interval of allowable values, different from Table 2.1 where each variable was assigned a different allowable interval. Furthermore, even with the larger number of variables, the optimal function values in Table 2.1 are better than those in Table 2.2 except for the modified DE results. This makes it clear that the approach presented here produced fairly good comparable results to those produced by previous researchers. In addition, several unconstrained test problems were solved and the solutions (refer to Table 2.3) achieved were very close to the true optimum solution with acceptable computation cost.

The following few chapters discuss various constraint handling techniques incorporated into the PC which made it a more generic and powerful optimization technique.

Table 2.2 Performance comparison of various algorithms solving Rosenbrock function

Method	Number of agents/ variables (N)	Function value G	Function evaluations	Range of values (Ψ_i)
CGA [33]	2	0.000145	250	−2.048 to 2.048
PAL [34]	2	≈0.01	5,250	−2.048 to 2.048
	5	≈2.5	100,000	−2.048 to 2.048
Modified DE [35]	2	1×10^{-6}	1,089	−5 to 10
	5	1×10^{-6}	11,413	−5 to 10
LCGA [36]	2	≈0.00003	–	−2.12 to 2.12

Table 2.3 Other unconstrained test problems

Function	True optimum solution	PC solution	Average function evaluations	Average CPU time (s)
Ackley	0	0.0000945	23,665	212.40
Beale	0	0	55	0.48
Bohachevsky	0	0	179,747	61.14
Booth	0	0	188,657	100.28
Branin	0.397887	0.397	128,079	109.69
Colville	0	0.00007	612,208	296.65
Rosenbrock	0	0	248,223	179.63
Zakharov	0	1.19E-10	197,550	125.45
Sum squares	0	5.79E-11	176,840	70.75
Shubert	−186.7309	−186.7096	234,734	121.85
Rastrigin		0.00000132	28,709	1,152
Power sum	0	0	221	0.05
Powell	0	0.000000196	774,556	310.65
Perm	0	9.4E-09	162,333	167.65
Michalewics	−1.8013	−1.8013	4,482	2.65
Matyas	0	0	191,898	127.15
Hump	0	0	96,119	51.07
Dixon and price	0	0.00000692	539,267	186.46
Goldstein and price	3	3	169,364	384.82
Griewank	0	1.79E-12	227,517	123.09
Hartmann (3 variables)	−3.86278	−3.86	133,038	70.03
Hartmann (6 variables)	−3.32237	−3.3223	31,535	21.45
Levy	0	0.0000041	161,394	85.78
Shifted sphere	−450	−450	42,653	13.76
Shifted rastrigin	−330	−330	36,032	14.28
Shifted ackley	−140	−140	213,296	276.75
Shifted greiwank	−180	−180	1,041	1.31
Schwefel	0	0	143,732	77.60

References

1. Wolpert, D.H., Tumer, K.: An introduction to collective intelligence. Technical Report, NASA ARC-IC-99–63, NASA Ames Research Center (1999)
2. Bieniawski, S.R.: Distributed optimization and flight control using collectives. Ph.D dissertation, Stanford University, CA, USA, (2005)
3. Wolpert, D.H.: Information theory—the bridge connecting bounded rational game theory and statistical physics. In: Braha, D., Minai, A.A., Bar-Yam, Y. (eds.) Complex Engineered Systems, pp. 262–290. Springer (2006)
4. Wolpert, D.H., Strauss, C.M.E., Rajnarayan, D.: Advances in distributed optimization using probability collectives. Adv. Complex Syst. **9**(4), 383–436 (2006)
5. Wolpert, D.H., Antoine, N.E., Bieniawski, S.R., Kroo, I.R.: Fleet assignment using collective intelligence. In: Proceedings of the 42nd AIAA Aerospace Science Meeting Exhibit (2004)
6. Bieniawski, S.R., Kroo, I.M., Wolpert, D.H.: Discrete, continuous, and constrained optimization using collectives. In: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, vol. 5, pp. 3079–3087 (2004)
7. Huang, C.F., Chang, B.R.: Probability collectives multi-agent systems: a study of robustness in search. LNAI 6422, Part II, pp. 334–343 (2010)
8. Huang, C.F., Bieniawski, S., Wolpert, D., Strauss, C.E.M.: A comparative study of probability collectives based multiagent systems and genetic algorithms. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp. 751–752 (2005)
9. Luo, D.L., Shen, C.L., Wang, B., Wu, W.H.: Air combat decision-making for cooperative multiple target attack using heuristic adaptive genetic algorithm. In: Proceedings of IEEE International Conference on Machine Learning and Cybernetics IEEE Press, pp. 473–478 (2005)
10. Luo, D.L., Duan, H.B., Wu, S.X., Li, M.Q.: Research on air combat decision-making for cooperative multiple target attack using heuristic ant colony algorithm. Acta Aeronautica et Astronautica Sinica **27**(6), 1166–1170 (2006)
11. Luo, D.L., Yang, Z., Duan, H.B., Wu, Z.G., Shen, C.L.: Heuristic particle swarm optimization algorithm for air combat decision-making on CMTA. Trans. Nanjing Univ. Aeronaut. Astronaut. **23**(1), 20–26 (2006)
12. Zhang, X.P, Yu, W.H., Liang, J.J., Liu, B.: Entropy regularization for coordinated target assignment. In: Proceedings of 3rd IEEE Conference on Computer Science and Information Technology, pp. 165–169 (2010)
13. Vasirani, M., Ossowski, S.: Collective-based multiagent coordination: a case study. LNAI **4995**, 240–253 (2008)
14. Modi, P., Shen, W., Tambe, M., Yokoo, M.: Adopt: asynchrous distributed constraint optimization with quality guarantees. Artif. Intell. **161**, 149–180 (2005)
15. Mohammad, H.A., Babak, H.K.: A distributed probability collectives optimization method for multicast in CDMA wireless data networks. In: Proceedings of 4th IEEE International Symposium on Wireless Communication Systems, art. No. 4392414, pp. 617–621 (2007)
16. Ryder, G.S., Ross, K.G.: A probability collectives approach to weighted clustering algorithms for ad hoc networks. In: Proceedings of Third IASTED International Conference on Communications and Computer Networks, pp. 94–99 (2005)
17. Goldberg, D.E., Samtani, M.P.: Engineering optimization via genetic algorithm. In: Proceedings of 9th Conference on Electronic Computation, pp. 471–484 (1986)
18. Ghasemi, M.R., Hinton, E., Wood, R.D.: Optimization of trusses using genetic algorithms for discrete and continuous variables. Eng. Comput. **16**(3), 272–301 (1999)
19. Moh, J., Chiang, D.: Improved simulated annealing search for structural optimization. AIAA J. **38**(10), 1965–1973 (2000)
20. Autry, B.: University course timetabling with probability collectives. Master's thesis, Naval Postgraduate School Montery, CA, USA (2008)

21. Sislak, D., Volf, P., Pechoucek, M., Suri, N.: Automated conflict resolution utilizing probability collectives optimizer. IEEE Trans. Syst. Man Cybern.: Appl. Rev. **41**(3), 365–375 (2011)
22. Arora, J.S.: Introduction to Optimum Design. Elsevier Academic Press (2004)
23. Vanderplaat, G.N.: Numerical Optimization Techniques for Engineering Design. Mcgraw-Hill, New York (1984)
24. Smyrnakis, M., Leslie, D.S.: Sequentially updated probability collectives. In: Proceedings of 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, pp. 5774–5779 (2009)
25. Kulkarni, A.J., Tai, K.: Probability collectives for decentralized, distributed optimization: a collective intelligence approach. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 1271–1275 (2008)
26. Kulkarni, A.J. Tai, K.: Probability collectives: a decentralized, distributed optimization for multi-agent systems. In: Mehnen, J., Koeppen, M., Saad, A., Tiwari, A. (eds.) Applications of Soft Computing, pp. 441–450. Springer (2009)
27. Kulkarni, A.J., Tai, K.: Solving constrained optimization problems using probability collectives and a penalty function approach. Int. J. Comput. Intell. Appl. **10**(4), 445–470 (2011)
28. Kulkarni A.J., Tai, K.: A probability collectives approach with a feasibility-based rule for constrained optimization. Appl. Comput. Intell. Soft Comput. 2011, Article ID 980216
29. Shoham, Y., Powers, R., Grenager, T.: Multi-agent reinforcement learning: a critical survey. www.cc.gatech.edu/∼isbell/reading/papers/MALearning.pdf Accessed 23 July 2011
30. Busoniu, L., Babuska, L., Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst Man Cybern.—Part C: Appl. Rev. **38**(2), 156–172 (2008)
31. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. Artif. Intell. **136** (2), 215–250 (2002)
32. Bowling, M., Veloso, M.: Rational and convergent learning in stochastic games. In: Proceedings of 17th International Conference on Artificial Intelligence, pp. 1021–1026 (2001)
33. Cheng, C.T., Wang, W.C., Xu, D.M., Chau, K.W.: Optimizing hydropower reservoir operation using hybrid genetic algorithm and chaos. Water Resour. Manag. **22**, 895–909 (2008)
34. Blumenthal, H.J., Parker, G.B.: Benchmarking punctuated anytime learning for evolving a multi-agent team's binary controllers. In: Proceedings of World Automation Congress, pp. 1–8 (2006)
35. Roger, L.S., Tan, M.S., Rangaiah, G.P.: Global optimization of benchmark and phase equilibrium problems using differential evolution. http://www.ies.org.sg/journal/current/v46/v462_3.pdf
36. Bouvry, P., Arbab, F., Seredynski, F.: Distributed evolutionary optimization, in manifold: rosenbrock's function case study. Inf. Sci. **122**, 141–159 (2000)

http://www.springer.com/978-3-319-15999-7

Probability Collectives
A Distributed Multi-agent System Approach for Optimization

Kulkarni, A.J.; Tai, K.; Abraham, A.

2015, IX, 157 p. 68 illus., Hardcover
ISBN: 978-3-319-15999-7