

Chapter 2

Socio-Inspired Optimization Using Cohort Intelligence

The nature-/bio-inspired optimization techniques such as genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), simulated annealing (SA), Tabu search, etc., have become popular due to their simplicity to implement and working based on rules. The GA is population based which is evolved using the operators such as selection, crossover, mutation, etc. According to Deb [1] and Ray et al. [2] the performance of GA is governed by the quality of the population being evaluated and may often reach very close to the global optimal solution and necessitates local improvement techniques to incorporate into it. The paradigm of Swarm Intelligence (SI) is a decentralized self organizing optimization approach inspired from social behavior of living organisms such as insects, fishes, etc. which can communicate with one another either directly or indirectly. The techniques such as Particle Swarm Optimization (PSO) is inspired from the social behavior of bird flocking and school of fish searching for food [3]. The Ant Colony Optimization (ACO) works on the ants' social behavior of foraging food following a shortest path [4]. Similar to ACO, the Bee Algorithm (BA) also works on the social behavior of honey bees finding the food; however, the bee colony tends to optimize the use of number of members involved in particular pre-decided tasks [5]. Generally, the swarm techniques are computationally intensive.

Kulkarni et al. [6] proposed an emerging Artificial Intelligence (AI) technique referred to as Cohort Intelligence (CI). It is inspired from the self-supervised learning behavior of the candidates in a cohort. The cohort here refers to a group of candidates interacting and competing with one another to achieve some individual goal which is inherently common to all the candidates. When working in a cohort, every candidate tries to improve its own behavior by observing the behavior of every other candidate in that cohort. Every candidate may follow a certain behavior in the cohort which according to itself may result into improvement in its own behavior. As certain qualities make a particular behavior which, when a candidate follows, it actually tries to adapt to the associated qualities. This makes every candidate learn from one another and helps the overall cohort behavior to evolve. The cohort behavior could be considered saturated, if for considerable number of

learning attempts the individual behavior of all the candidates does not improve considerably and candidates' behaviors become hard to distinguish. The cohort could be assumed to become successful when for a considerable number of times the cohort behavior saturates to the same behavior.

This chapter discusses the CI methodology framework in detail and further validates its ability by solving a variety of unconstrained test problems. This demonstrates its strong potential of being applicable for solving unimodal as well as multimodal problems.

2.1 Framework of Cohort Intelligence

Consider a general unconstrained problem (in the minimization sense) as follows:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{x}) = f(x_1, \dots, x_i, \dots, x_N) \\ &\text{Subject to} && \Psi_i^{\text{lower}} \leq x_i \leq \Psi_i^{\text{upper}}, \quad i = 1, \dots, N \end{aligned} \quad (2.1)$$

As a general case, assume the objective function $f(\mathbf{x})$ as the behavior of an individual candidate in the cohort which it naturally tries to enrich by modifying the associated set of characteristics/attributes/qualities $\mathbf{x} = (x_1, \dots, x_i, \dots, x_N)$.

Having considered a cohort with number of candidates C , every individual candidate c ($c = 1, \dots, C$) belongs a set of characteristics/attributes/qualities $\mathbf{x}^c = (x_1^c, \dots, x_i^c, \dots, x_N^c)$ which makes the overall quality of its behavior $f(\mathbf{x}^c)$. The individual behavior of each candidate c is generally being observed by itself and every other candidate (c) in the cohort. This naturally urges every candidate c to follow the behavior better than its current behavior. More specifically, candidate c may follow $f^*(\mathbf{x}^{(c)})$ if it is better than $f^*(\mathbf{x}^c)$, i.e. $f^*(\mathbf{x}^{(c)}) < f^*(\mathbf{x}^c)$. Importantly, following a behavior $f(\mathbf{x})$ refers to following associated qualities $\mathbf{x} = (x_1, \dots, x_i, \dots, x_N)$ with certain variations t associated with them. However, following better behavior and associated qualities is highly uncertain. This is because; there is certain probability involved by which it selects certain behavior to follow. In addition, a stage may come where the cohort behavior could become saturated. In other words, at a certain stage, there could be no improvement in the behavior of an individual candidate for a considerable number of learning attempts. Such situation is referred to as saturation stage. This makes every candidate to expand its search around the qualities associated with the current behavior being followed. The mathematical formulation of the CI methodology is explained below in detail with the algorithm flowchart in Fig. 2.1 [6, 7].

The procedure begins with the initialization of number of candidates C , sampling interval Ψ_i for each quality x_i , $i = 1, \dots, N$, learning attempt counter $n = 1$, and the setup of sampling interval reduction factor $r \in [0, 1]$, convergence parameter $\varepsilon = 0.0001$, number of variations t . The values of C , t and v are chosen based on preliminary trials of the algorithm.

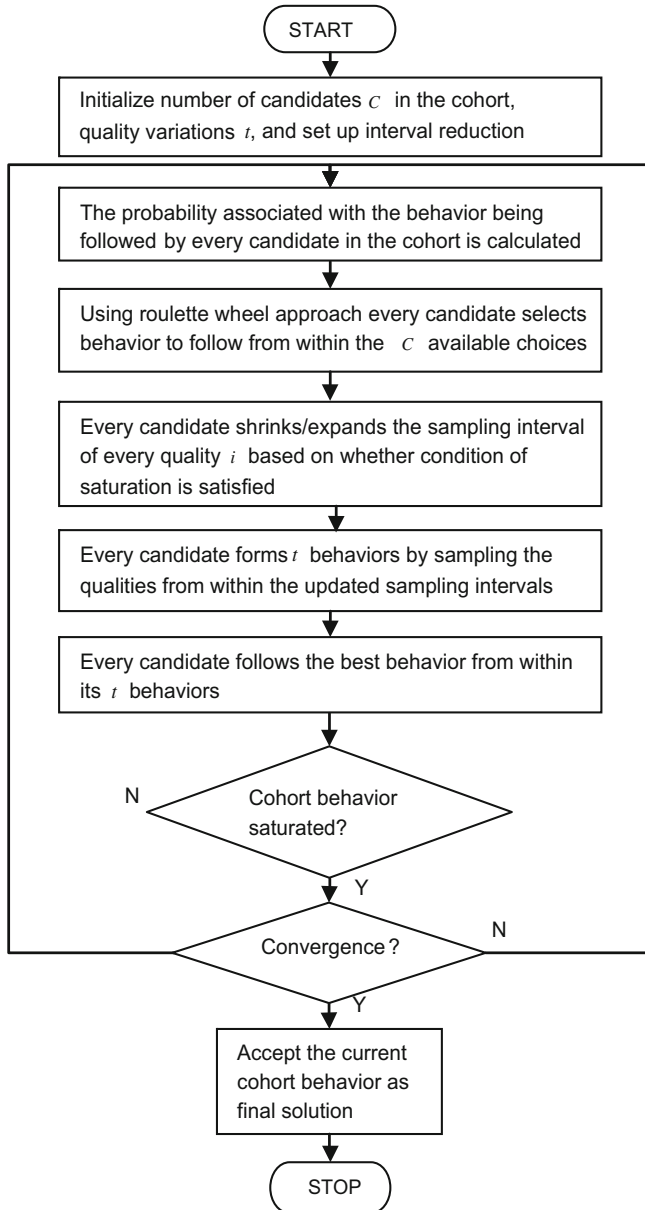


Fig. 2.1 Cohort intelligence (CI) flowchart

Step 1. The probability of selecting the behavior $f^*(\mathbf{x}^c)$ of every associated candidate c ($c = 1, \dots, C$) is calculated as follows:

$$p^c = \frac{1/f^*(\mathbf{x}^c)}{\sum_{c=1}^C 1/f^*(\mathbf{x}^c)} \quad , \quad (c = 1, \dots, C) \quad (2.2)$$

Step 2. Every candidate c ($c = 1, \dots, C$) generates a random number $r \in [0, 1]$ and using a roulette wheel approach decides to follow corresponding behavior $f^*(\mathbf{x}^{c[?]})$ and associated qualities $\mathbf{x}^{c[?]} = (x_1^{c[?]}, \dots, x_i^{c[?]}, \dots, x_N^{c[?]})$. The superscript $[?]$ indicates that the behavior is selected by candidate c and not known in advance. The roulette wheel approach could be most appropriate as it provides chance to every behavior in the cohort to get selected purely based on its quality. In addition, it also may increase the chances of any candidate to select the better behavior as the associated probability stake p^c ($c = 1, \dots, C$) presented in Eq. (2.2) in the interval $[0, 1]$ is directly proportional to the quality of the behavior $f^*(\mathbf{x}^c)$. In other words, better the solution, higher is the probability of being followed by the candidates in the cohort.

Step 3. Every candidate c ($c = 1, \dots, C$) shrinks the sampling interval $\Psi_i^{c[?]}$, $i = 1, \dots, N$ associated with every variable $x_i^{c[?]}$, $i = 1, \dots, N$ to its local neighborhood. This is done as follows:

$$\Psi_i^{c[?]} \in [x_i^{c[?]} - (\|\Psi_i\|/2), x_i^{c[?]} + (\|\Psi_i\|/2)] \quad (2.3)$$

where $\Psi_i = (\|\Psi_i\|) \times r$.

Step 4. Each candidate c ($c = 1, \dots, C$) samples t qualities from within the updated sampling interval $\Psi_i^{c[?]}$, $i = 1, \dots, N$ associated with every variable $x_i^{c[?]}$, $i = 1, \dots, N$ and computes a set of associated t behaviors, i.e. $\mathbf{F}^{c,t} = \{f(\mathbf{x}^c)^1, \dots, f(\mathbf{x}^c)^j, \dots, f(\mathbf{x}^c)^t\}$, and selects the best function $f^*(\mathbf{x}^c)$ from within. This makes the cohort is available with C updated behaviors represented as $\mathbf{F}^C = \{f^*(\mathbf{x}^1), \dots, f^*(\mathbf{x}^c), \dots, f^*(\mathbf{x}^C)\}$.

Step 5. The cohort behavior could be considered saturated, if there is no significant improvement in the behavior $f^*(\mathbf{x}^c)$ of every candidate c ($c = 1, \dots, C$) in the cohort, and the difference between the individual behaviors is not very significant for successive considerable number of learning attempts, i.e. if

1. $\left\| \max(\mathbf{F}^C)^n - \max(\mathbf{F}^C)^{n-1} \right\| \leq \varepsilon$, and
2. $\left\| \min(\mathbf{F}^C)^n - \min(\mathbf{F}^C)^{n-1} \right\| \leq \varepsilon$, and

3. $\|\max(\mathbf{F}^C) - \min(\mathbf{F}^C)\| \leq \varepsilon$, every candidate c ($c = 1, \dots, C$) expands the sampling interval $\Psi_i^{c[?]}$, $i = 1, \dots, N$ associated with every quality $x_i^{c[?]}$, $i = 1, \dots, N$ to its original one $\Psi_i^{lower} \leq x_i \leq \Psi_i^{upper}$, $i = 1, \dots, N$.

Step 6. If either of the two criteria listed below is valid, accept any of the C behaviors from current set of behaviors in the cohort as the final objective function value $f^*(\mathbf{x})$ as the final solution and stop, else continue to Step 1.

- (a) If maximum number of attempts exceeded.
- (b) If cohort saturates to the same behavior (satisfying the conditions in Step 5) for τ_{\max} times.

2.2 Theoretical Comparison with Contemporary Techniques

Particle swarm optimization PSO is a population-based stochastic search algorithm developed by Kennedy and Eberhart [3]. Due to its simple concept, it has been applied to many optimization problems. The PSO itself did not work well in solving constrained problems [8]. To overcome this shortcoming, many modified PSO techniques such as Quantum-behaved PSO [9], Improved Vector PSO (IVPSO) [10] and other techniques which controlled the velocity of the swarm were used. All these techniques depend upon the control parameters in the velocity updating model, which are the inertia weight and acceleration coefficients. Another technique referred to as the Barebones PSO (BPSO) [11] used Gaussian normal distribution to update the values of the particles in the solution. This removed the necessity of inertia weight and acceleration coefficients. In the PSO variations the entire swarm has a collective intelligence. While the individual particle keeps track of its own best solution, every particle in the swarm is also aware of the best solution found by the entire swarm [3]. The movement of each particle is some function of this individual best and the group's best values. The Fully Informed PSO (FIPSO) is one technique that does not merely depend on the best solution offered globally. This technique samples the solutions offered by its entire neighborhood and follows a point in space that is calculated using this complete information [12].

Another technique popular today is the Genetic Algorithm (GA). This technique follows the principle of survival of the fittest. The best solutions in a particular generation are taken forward to the next one, and new solutions are generated by using crossover as well as applying mutations on it. Ant Colony optimization (ACO) [4, 13] follows autocatalytic behavior which is characterized by a positive feedback, where the probability with which an agent chooses a path increases with

the number of agents that previously chose the same path [14]. However it is difficult to solve continuous optimization problem using ACO directly, as there are limitations in number of choices for ants at each stage. Some recent research has extended classical ACO to solve continuous optimization problem.

In CI, however, the collective effort of the swarm is replaced by the competitive nature of a cohort. Every candidate tries to follow the behavior of a candidate that has shown better results in that particular iteration. In following this behavior, it tries to incorporate some of the qualities that made that behavior successful. This competitive behavior motivates each candidate to perform better, and leads to an eventual improvement in the behaviors of all the candidates. This technique differs from PSO and barebones PSO in that it does not check merely the best solution, but is fully informed of the activities of its fellow candidates and follows a candidate selected using the roulette wheel approach. However, it is also different from the FIPSO which keeps track of the entire swarm in that it follows the behavior of only one candidate, not a resultant of the results presented by the entire swarm. CI also differs from GA, as there is no direct exchange of certain properties or even mutation. Rather, candidates decide to follow a fellow candidate, and try to imbibe the qualities that led that candidate to reach its solution. The values of these qualities are not replicated exactly. They are instead taken from a close neighborhood of the values of the qualities of the candidate being followed. This gives a variation in the solutions obtained and this is how the cohort can avoid getting trapped in local minima. CI differs from ACO as the autocatalytic nature of the ants is replaced by competitive nature of the cohorts. Instead of having tendency to follow most followed behavior, candidates in CI try to incorporate the best behavior in every iteration. This prevents the algorithm from getting caught into local minima by not relying on the behavior that is locally optimal. The CI algorithm has shown itself to be comparable to the best results obtained from the various techniques. The sharing of the best solution among candidates in CI gives a direction for all the candidates to move towards, but the independent search of each candidate ensures that the candidates come out of local minima to get the best solutions.

2.3 Validation of Cohort Intelligence

The performance of the proposed CI algorithm was tested by solving unconstrained well known test problems such as Ackley, Dixon and Price, Griewank, Hartmann, Levy, Michalewicz, Perm, Powell, Powersum, Rastrigin, Schwefel, Sphere, Sum Square, Trid, Zakhrov, etc. with different problem sizes. The algorithm was coded in MATLAB 7.8.0 (R2009A) on Windows platform using Intel Core 2 Duo T6570, 2.10 GHz processor speed and 2 GB RAM. Every test problem was solved 20 times with number of candidates C , number of variations in the behavior t chosen as 5 and 7, respectively. The values of the reduction factor r chosen for

Table 2.1 Summary of unconstrained test problem solutions with 5 variables

| Problem | True optimum | CI algorithm Best Mean Worst | Function evaluations | Standard deviation | Time (s) |
|-----------------|--------------|---------------------------------------|----------------------|--------------------|----------|
| Ackley | 0.0000 | 2.04E-10 5.58E-09 2.70E-08 | 43,493 | 1.06E-08 | 1.42 |
| Dixon and Price | 0.0000 | 9.86E-32 1.43E-09 4.28E-09 | 82,770 | 4.28E-09 | 1.83 |
| Griewank | 0.0000 | 0.00E+00 1.80E-02 3.70E-02 | 163,485 | 9.73E-03 | 5.15 |
| Levy | 0.0000 | 1.28E-21 6.22E-20 2.18E-19 | 38,993 | 7.70E-20 | 1.43 |
| Michalewicz | -4.687658 | -3.96E+00 -3.40E+00 -2.80E+00 | 188,700 | 3.61E-01 | 5.61 |
| Perm | 0.0000 | 4.20E-01 2.82E+00 9.83E+00 | 225,960 | 2.79E+00 | 8.07 |
| Powell | 0.0000 | 8.73E-09 2.31E-06 1.88E-05 | 162,510 | 5.54E-06 | 5.65 |
| Rastrigin | 0.0000 | 9.95E-01 1.50E+00 2.00E+00 | 277,440 | 4.91E-01 | 5.99 |
| Schwefel | 0.0000 | 2.83E-06 6.03E-06 8.86E-06 | 1,570,718 | 1.70E-06 | 38.73 |
| Sphere | 0.0000 | 2.69E-29 1.58E-28 2.55E-28 | 12,345 | 5.84E-29 | 0.35 |
| Sum Square | 0.0000 | 1.56E-18 2.96E-18 6.70E-18 | 7470 | 1.37E-18 | 0.22 |
| Zakhrov | 0.0000 | 8.38E-19 1.95E-18 3.20E-18 | 7470 | 6.62E-19 | 0.23 |

every unconstrained test problem are listed in Table 2.6. These parameters were derived empirically over numerous experiments.

The CI performance solving a variety of unconstrained test problems is presented in Tables 2.1, 2.2, 2.3, 2.4 and 2.5 with increase in the number of variables

Table 2.2 Summary of unconstrained test problem solutions with 10 variables

| Problem | True optimum | CI algorithm Best Mean Worst | Function evaluations | Standard deviation | Time (s) |
|-----------------|--------------|--|----------------------|--------------------|----------|
| Ackley | 0.0000 | 8.97E-08 4.58E-07 9.30E-07 | 30,765 | 3.60E-07 | 1.39 |
| Dixon and Price | 0.0000 | 6.67E-01 6.67E-01 6.67E-01 | 359,640 | 1.48E-14 | 12.11 |
| Griewank | 0.0000 | 7.48E-03 2.50E-02 4.68E-02 | 432,368 | 1.18E-02 | 17.49 |
| Levy | 0.0000 | 2.02E-06 7.34E-06 1.12E-05 | 44,798 | 2.78E-06 | 2.29 |
| Powell | 0.0000 | 7.73E-06 6.28E-05 2.13E-04 | 186,570 | 7.29E-05 | 9.46 |
| Rastrigin | 0.0000 | 6.96E+00 1.06E+01 1.49E+01 | 261,998 | 2.31E+00 | 10.04 |
| Rosenbrock | 0.0000 | 0.0000E+00 0.0000E+00 0.0000E+00 | 13,605 | 0.0000E+00 | 0.49 |
| Schwefel | 0.0000 | 1.20E-06 1.52E-06 1.74E-06 | 2,023,103 | 1.66E-07 | 84.55 |
| Sphere | 0.0000 | 7.17E-22 9.47E-22 1.62E-21 | 18,668 | 2.73E-22 | 0.76 |
| Sum Square | 0.0000 | 1.32E-16 2.37E-15 2.21E-14 | 14,948 | 6.58E-15 | 0.61 |
| Zakhrov | 0.0000 | 3.22E-12 5.15E-12 6.68E-12 | 22,365 | 1.24E-12 | 0.90 |

associated with the individual problem. It is observed that with increase in number of variables, the computational cost, i.e. function evaluations and computational time was increased. However, the small standard deviation values for all the

Table 2.3 Summary of unconstrained test problem solutions with 20 variables

| Problem | True optimum | CI algorithm Best Mean Worst | Function evaluations | Standard deviation | Time (s) |
|-----------------|--------------|---------------------------------------|----------------------|--------------------|----------|
| Ackley | 0.0000 | 2.97E-11 6.04E-11 2.96E-10 | 329,745 | 7.88E-11 | 21.87 |
| Dixon and Price | 0.0000 | 7.47E-01 7.77E-01 8.22E-01 | 358,800 | 2.31E-02 | 20.65 |
| Griewank | 0.0000 | 0.00E+00 7.40E-04 7.40E-03 | 187,763 | 2.22E-03 | 14.00 |
| Levy | 0.0000 | 7.31E-15 1.73E-13 8.79E-13 | 329,768 | 3.28E-13 | 27.01 |
| Powell | 0.0000 | 8.46E-05 2.18E-04 3.60E-04 | 539,640 | 8.02E-05 | 44.86 |
| Rastrigin | 0.0000 | 2.19E+01 3.88E+01 5.57E+01 | 408,758 | 7.87E+00 | 24.60 |
| Rosenbrock | 0.0000 | 0.00E+00 3.96E-30 1.98E-29 | 17,288 | 7.92E-30 | 1.01 |
| Schwefel | 0.0000 | 4.38E-06 5.56E-06 6.12E-06 | 2,023,950 | 5.03E-07 | 134.37 |
| Sphere | 0.0000 | 6.22E-14 7.60E-14 9.40E-14 | 26,183 | 1.11E-14 | 1.78 |
| Sum Square | 0.0000 | 8.88E-11 3.78E-10 1.88E-09 | 37,335 | 5.17E-10 | 2.50 |
| Zakhrov | 0.0000 | 1.00E-06 2.19E-06 4.97E-06 | 37,290 | 1.08E-06 | 2.53 |

functions independent of the number of variables highlighted its robustness. The effect of CI parameters such as number of candidates C , reduction rate r and number of variations in behavior t was also analyzed on unimodal as well as multimodal

Table 2.4 Summary of unconstrained test problem solutions with 30 variables

| Problem | True optimum | CI algorithm Best Mean Worst | Function evaluations | Standard deviation | Time (s) |
|-----------------|--------------|---------------------------------------|----------------------|--------------------|----------|
| Ackley | 0.0000 | 1.59E-07 5.91E-07 1.79E-06 | 299,835 | 6.41E-07 | 28.91 |
| Dixon and Price | 0.0000 | 9.89E-01 1.10E+00 1.23E+00 | 357,413 | 7.25E-02 | 29.07 |
| Griewank | 0.0000 | 2.90E-06 1.82E-03 7.57E-03 | 398,918 | 2.86E-03 | 39.91 |
| Levy | 0.0000 | 4.89E-07 4.62E-05 2.47E-04 | 674,363 | 9.15E-05 | 80.24 |
| Powell | 0.0000 | 7.23E-02 1.84E-01 4.23E-01 | 743,595 | 1.23E-01 | 89.86 |
| Rastrigin | 0.0000 | 5.57E+01 8.77E+01 1.00E+02 | 296,745 | 1.41E+01 | 26.40 |
| Rosenbrock | 0.0000 | 0.00E+00 7.22E-30 3.61E-29 | 19,470 | 1.44E-29 | 1.58 |
| Schwefel | 0.0000 | 1.06E-05 1.20E-05 1.37E-05 | 2,023,290 | 8.74E-07 | 180.62 |
| Sphere | 0.0000 | 1.91E-13 2.66E-13 4.51E-13 | 26,235 | 7.38E-14 | 2.51 |
| Sum Square | 0.0000 | 1.28E-10 2.52E-04 9.75E-04 | 55,433 | 3.79E-04 | 5.23 |
| Zakhrov | 0.0000 | 1.63E-04 6.93E-04 1.15E-03 | 221,798 | 3.03E-04 | 20.98 |

functions. The effect is visible in Fig. 2.3 where effect of these parameters on Sphere function and Ackley Function is presented as a representative to unimodal and multimodal function, respectively. The visualization for the convergence of the

Table 2.5 Summary of solution to Powersum, Hartmann and Trid function

| Problem | No of variables | True optimum | CI algorithm Best Mean Worst | Function evaluations | Standard deviation | Time (s) |
|----------|-----------------|--------------|---------------------------------------|----------------------|--------------------|----------|
| Powersum | 4 | 0.0000 | 1.38E-06 6.74E-05 1.34E-04 | 619,125 | 4.44E-05 | 17.48 |
| Hartmann | 6 | -3.86278 | -3.32E+00 -3.32E+00 -3.32E+00 | 710,483 | 1.67E-03 | 31.53 |
| Trid | 6 | -50.0000 | -4.87E+01 -4.87E+01 -4.87E+01 | 177,090 | 4.92E-03 | 4.27 |

representative Ackley function is presented in Fig. 2.2 for learning attempts 1, 10, 15 and 30. For both types of functions, the computational cost, i.e. function evaluations and computational time was observed to be increasing linearly with increasing number of candidates C (refer to Fig. 2.3a, b) as well as number of variations in behavior t (refer to Fig. 2.3e, f, k and l). This was because, with increase in number of candidates, number of behavior choices i.e. function evaluations also increased. Moreover, with fewer number of candidates C , the quality of the solution at the end of first learning attempt referred to as initial solution as well as the converged solution were quite close to each other and importantly, the converged solutions and the converged solution was suboptimal. The quality of both the solutions improved with increase in number of candidates C . This was because fewer number of behavior choices were available with fewer number of candidates, whereas with increase in number of candidates the total choice of behavior also increased as a result initial solution worsened whereas converged solution improved as sufficient time was provided for saturation. Due to this a widening gap between initial solution and converged solution was observed in the

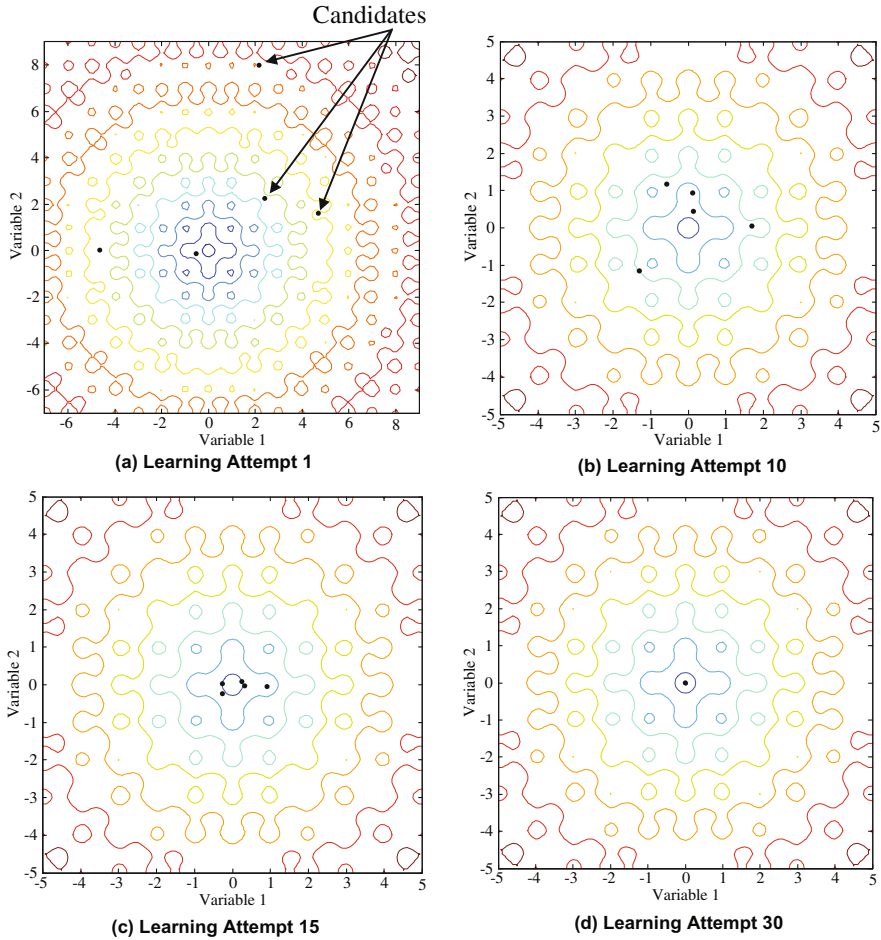


Fig. 2.2 Visualization of convergence of solutions at various learning attempts for Ackley function

trend of initial solution and final solution. It is evident in Fig. 2.3g, h. In addition, it is important observation from Table 2.6 that larger values of the reduction rate r were required as the problem size increased. This could be because as the number of variables increase size of the problem search space also increased and larger values of the reduction rate r were required. However, it is evident from Fig. 2.3c, d, i, j that for the same size of unimodal as well as multimodal problems with the

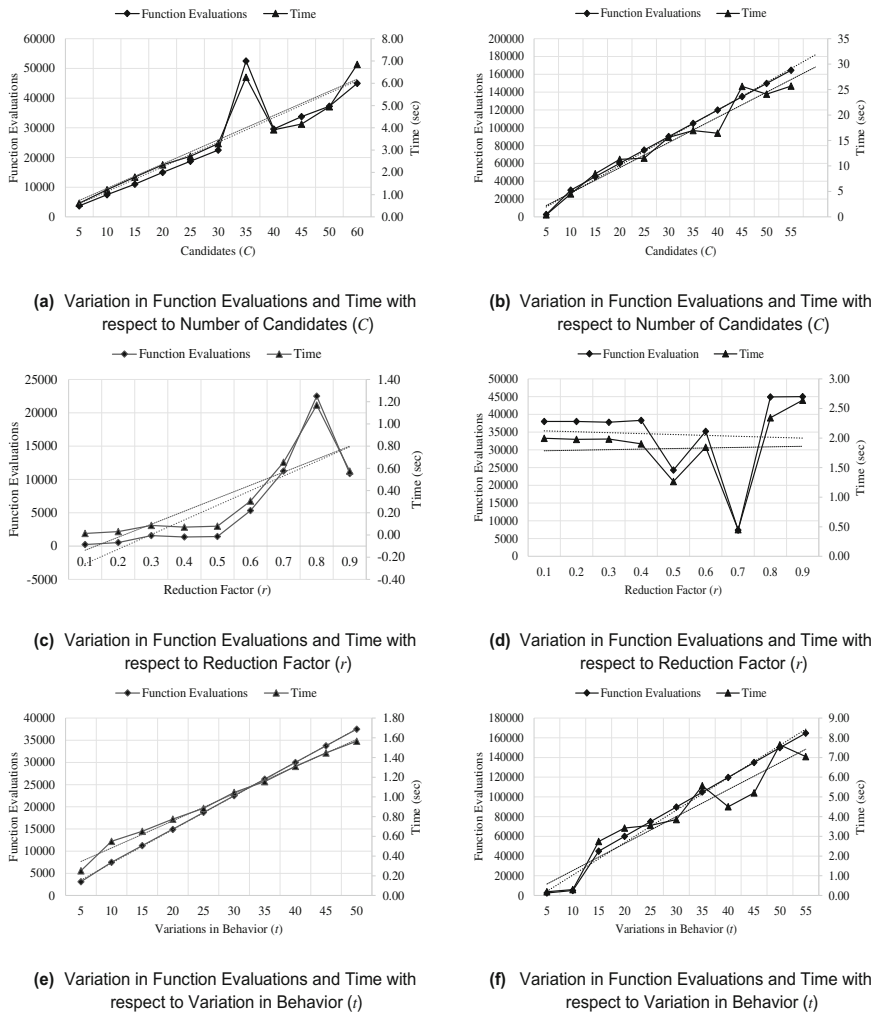
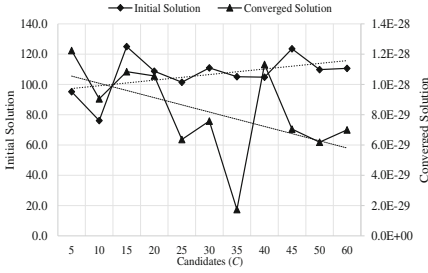
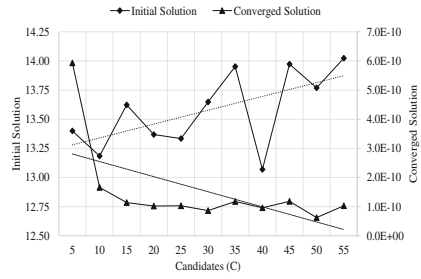


Fig. 2.3 Influence of number of candidates (C), reduction factor (r) and variation in behavior (t) on CI algorithm performance

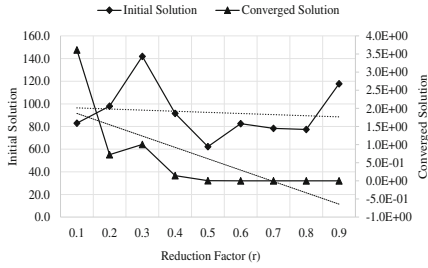
increase in the value of reduction rate r larger solution space was available for exploration which when searched resulted into increased converged solution quality as well as associated computational cost. In addition, similar to the increase in number of candidates C , it was observed that with the increase in number of variations in behavior t the computational cost increased along with the



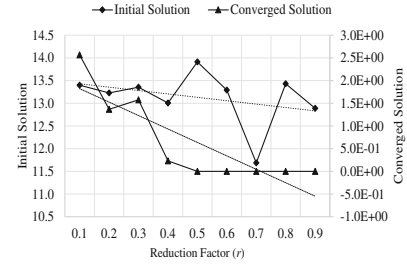
(g) Variation in Initial and Final Solution with respect to Number of Candidates (C)



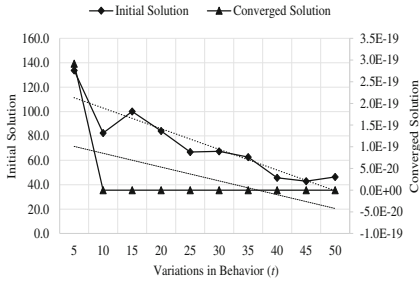
(h) Variation in Initial and Final Solution with respect to Number of Candidates (C)



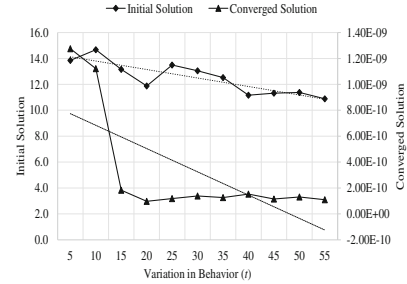
(i) Variation in Initial and Final Solution with respect to Reduction Factor (r)



(j) Variation in Initial and final Solution with respect to Reduction Factor (r)



(k) Variation in Initial and Final Solution with respect to Variation in Behavior (t)



(l) Variation in Initial and Final Solution with respect to Variation in Behavior (t)

Fig. 2.3 (continued)

improvement in converged solution quality. Furthermore, the converged solution quality did not improve significantly after a certain number of variations in behavior t . Moreover, with the increase in variations in behavior t , the difference between the initial solution and the converged solution gradually decreased.

Table 2.6 Summary of reduction factor r values for unconstrained test problems

| Problem | Reduction factor r | | | | | |
|-----------------|----------------------|---------|---------|----------|----------|----------|
| | $n = 4$ | $n = 5$ | $n = 6$ | $n = 10$ | $n = 20$ | $n = 30$ |
| Ackley | – | 0.800 | – | 0.850 | 0.970 | 0.980 |
| Dixon and Price | – | 0.850 | – | 0.980 | 0.997 | 0.997 |
| Griewank | – | 0.950 | – | 0.997 | 0.950 | 0.980 |
| Hartmann | – | – | 0.997 | – | – | – |
| Levy | – | 0.800 | – | 0.950 | 0.980 | 0.995 |
| Michalewicz | – | 0.950 | – | 0.970 | 0.980 | 0.970 |
| Perm | – | 0.997 | – | – | – | – |
| Powell | – | 0.850 | – | 0.990 | 0.950 | 0.980 |
| Powersum | 0.997 | – | – | – | – | – |
| Rastrigin | – | 0.980 | – | 0.900 | 0.980 | 0.980 |
| Rosenbrock | – | – | – | – | – | – |
| Schwefel | – | 0.997 | – | 0.997 | 0.997 | 0.997 |
| Sphere | – | 0.800 | – | 0.900 | 0.950 | 0.950 |
| Sum Square | – | 0.800 | – | 0.900 | 0.970 | 0.980 |
| Trid | – | – | 0.800 | 0.980 | 0.980 | 0.980 |
| Zakharov | – | 0.800 | – | 0.950 | 0.980 | 0.980 |

References

1. Deb, K.: An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **186**, 311–338 (2000)
2. Ray, T., Tai, K., Seow, K.C.: Multiobjective design optimization by an evolutionary algorithm. *Eng. Optim.* **33**(4), 399–424 (2001)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948 (1995)
4. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization: artificial ants as a computational intelligence technique. *IEEE Comput. Intell. Mag.*, 28–39 (2006)
5. Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M.: The bees algorithm. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK (2005)
6. Kulkarni, A.J., Durugkar, I.P., Kumar, M.R.: Cohort intelligence: a self-supervised learning behavior. In: *Proceedings of the IEEE Conference on Systems, Man and Cybernetics 2013*, pp. 1396–1400 (2013)
7. Kulkarni, A.J., Shabir, H.: Solving 0-1 Knapsack problem using cohort intelligence algorithm. *Int. J. Mach. Learn. Cybernet.* (2014). doi:[10.1007/s13042-014-0272-y](https://doi.org/10.1007/s13042-014-0272-y)
8. Wang, H.: Opposition-based barebones particle swarm for constrained nonlinear optimization problems. *Math. Probl. Eng.* **2012**, Article ID 761708 (2010)
9. Coelho, L.D.S.: Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Syst. Appl.* **37**(2), 1676–1683 (2010)
10. Sun, C.L., Zeng, J.C., Pan, J.S.: An new vector particle swarm optimization for constrained optimization problems. In: *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO'09)*, pp. 485–488 (2009)
11. Kennedy, J.: Bare bones particle swarms. In: *Proceedings of the IEEE Swarm Intelligence Symposium (SIS'03)*, pp. 80–87 (2003)

12. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. *IEEE Trans. Evolut. Comput.* **8**(3), 204–210 (2004)
13. Chen, L., Sun, H., Wang, S.: Solving continuous optimization using ant colony algorithm. In: *Second International Conference on Future Information Technology and Management Engineering*, pp. 92–95 (2009)
14. Dorigo, M., Maniezzo, V., Coloni, A.: Ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybernet Part B: Cybernet* **26**(1), 29–41 (1996)

Cohort Intelligence: A Socio-inspired Optimization
Method

Kulkarni, A.J.; Krishnasamy, G.; Abraham, A.

2017, XI, 134 p. 29 illus., Hardcover

ISBN: 978-3-319-44253-2