# Competitive Island-Based Cooperative Coevolution for Efficient Optimization of Large-Scale Fully-Separable Continuous Functions

Kavitesh K. Bali[1] Rohitash Chandra[1] Mohammad N. Omidvar[2]

[1] School of Computing Information and Mathematical Sciences,
University of South Pacific, Suva, Fiji.
[2] School of Computer Science and IT, RMIT University
Melbourne, Australia.
bali.kavitesh@gmail.com, c.rohitash@gmail.com,
mohammad.omidvar@rmit.edu.au

**Abstract.** In this paper, we investigate the performance of introducing competition in cooperative coevolutionary algorithms to solve large-scale fully-separable continuous optimization problems. It may seem that solving large-scale fully-separable functions is trivial by means of problem decomposition. In principle, due to lack of variable interaction in fully-separable problems, any decomposition is viable. However, the decomposition strategy has shown to have a significant impact on the performance of cooperative coevolution on such functions. Finding an optimal decomposition strategy for solving fully-separable functions is laborious and requires extensive empirical studies. In this paper, we use a competitive two-island cooperative coevolution in which two decomposition strategies compete and collaborate to solve a fully-separable problem. Each problem decomposition has features that may be beneficial at different stages of optimization. Therefore, competition and collaboration of such decomposition strategies may eliminate the need for finding an optimal decomposition. The experimental results in this paper suggest that competition and collaboration of suboptimal decomposition strategies of a fully-separable problem can generate better solutions than the standard cooperative coevolution with standalone decomposition strategies. We also show that a decomposition strategy that implements competition against itself can also improve the overall optimization performance.

## 1 Introduction

Various meta-heuristic algorithms have been developed for continuous global function optimization. However, one of the core issues associated with these techniques is their scalability to higher dimensions [1]. *Divide-and-conquer* is an effective technique for solving large-scale complex problems. Cooperative Coevolution (CC) [2] is an explicit means of problem decomposition in the context of evolutionary algorithms (EAs) [3].

A major challenge in using CC for large-scale optimization is decomposition of a given problem into smaller sub-problems. Variable interaction [4] is a major constraint that governs the decomposition of a problem [5]. It is generally believed that placement of interacting variables into separate subcomponents degrades the optimization performance significantly [6, 2]. For this reason, many decomposition methods have been proposed for automatic variable interaction detection [5, 7, 8, 9].

For some classes of problems such as fully-separable functions or overlapping functions [10], there is no unique decomposition. In principle, for a fully-separable function, all of the decision variables can be optimized independently, hence any decomposition is viable. This may suggest that a complete decomposition in which each variables is placed in a separate subcomponent is the most efficient decomposition. However, a recent study [11] showed that the performance of CC is very sensitive to decomposition, even on fully-separable problems. Some partially separable functions may also contain a relatively high dimensional fully-separable subcomponent. Poor decomposition strategies of such subcomponents may hinder the convergence of CC to a high quality solutions [11]. Unfortunately, finding an effective decomposition strategy for fully-separable functions is a laborious task, which requires extensive experimentation [11]. To alleviate the need for finding the optimal decomposition, Omidvar et al. used a very simple reinforcement learning approach to dynamically adapt the decomposition strategy [11].

Recently, competitive island-based cooperative coevolution (CICC) algorithm was proposed for global optimization problems that gave promising results [12]. CICC has been originally designed for training recurrent neural networks on chaotic time series problems [13, 14]. Neuron and synapse level problem decomposition strategies were implemented as islands that competed and collaborated with each other. CICC has shown to be a promising approach for solving large scale fully-separable functions for which there is no unique decomposition strategy [12].

In this paper, we apply CICC algorithm to eliminate the need for finding an optimal decomposition in the context of fully-separable problems. We speculate that competition and collaboration of decomposition strategies exhibiting various features can yield solutions with a quality better than individual decompositions used in isolation. In particular, the aim of this paper is to answer the following questions:

- How effective is the CICC algorithm when applied to large-scale fully-separable function optimization ?
- Can CICC with two *same* effective decomposition strategies adapted from [11], competing against itself improve the overall optimization performance ?
- Can competition and collaboration of two *different* suboptimal decomposition strategies yield solutions with a quality better than the near-optimal standalone decomposition strategies used in isolation ?

The organization of the rest of this paper is as follows. Section 2 describes the proposed method and its application to large-scale fully-separable continuous

functions. Experimental results and their analyses are provided in Sections 3 and 4. Section 5 concludes the paper and outlines possible future extensions.

## 2 Competitive Island Cooperative Coevolution for Fully-Separable Continuous Functions

In this section, we provide details of cooperative coevolution method that features competition and collaboration with species, motivated by evolution in nature. In nature, competition in an environment of limited resources is mandatory for survival. Collaboration enforces interaction and sharing of resources between the different species having distinct characteristics for adaptation with respect to challenges such as environmental changes [13, 14]. Interaction and migration of genetic material or information between the sub-populations can be advantageous in the evolutionary process. Hence, competition and collaboration are vital aspects of the evolutionary process where different groups of species compete for resources in the same environment.

**Algorithm 1:** Competitive Two-Island Cooperative Coevolution algorithm CICC [12].

```
Stage 1: Initialization:
i. Cooperatively evaluate Island One
ii. Cooperatively evaluate Island Two
Stage 2: Evolution:
while FE ≤ Global-Evolution-Time do
    while FE ≤ Island-Evolution-Time do
        foreach Sub-population at Island-One do
            foreach Depth of n Generations do
                Create new individuals using genetic operators
                Cooperative Evaluation of Island One
            end
        end
    end
    while FE ≤ Island-Evolution-Time do
        foreach Sub-population at Island-Two do
            foreach Depth of n Generations do
                Create new individuals using genetic operators
                Cooperative Evaluation of Island Two
            end
        end
    end
    Stage 3: Competition: Compare and mark the island with best fitness.
    Stage 4: Collaboration: Inject the best individual from the island with better fitness into the other
    island.
    if ErrorIslandOne ≤ ErrorIslandTwo then
        Inject Island One's best individual into Island Two.
    end
    else
        Inject Island Two's best individual into Island One.
    end
end
```

In the proposed competitive algorithm, two problem decomposition strategies are implemented as separate islands and evolved by an independent evolutionary algorithm. These islands enforce competition by comparing their solutions after a fixed time (fitness evaluations), and exchange the best solution between the islands[13, 14, 12]. Interaction between the two islands occur after separate evolutionary processes are executed in phases that are defined by fitness evaluations

or generations. After a phase of evolution is completed, the algorithm migrates feasible solutions from the winner island into the others. The proposed CICC method for fully-separable problems is presented in Algorithm 1 where the key aspects are initialization, evolution, competition and collaboration.

For this paper, we focus on a two-island competition algorithm [14, 12]. This algorithm can be extended to more islands in further studies.

### 2.1 Initialization

In CICC, a problem decomposition strategy is implemented as an island. To enforce an unbiased competition, we ensure that both islands (Island One and Island Two) begin search with the same genetic materials in the sub-populations and cooperatively evolve them in isolation.

Initially, all the sub-populations of Island One are initialized with random-real number values from a domain specified in Table 1. These real values (from Island One) are copied into the sub-populations of Island Two. A problem decomposition (configuration) for an island can either have same sized *(uniform)* or varied sized *(non-uniform)* subcomponents. Since we are utilizing the problem decomposition strategies from [11], we employ uniform subcomponent sizes for this study. The highest level of decomposition for an island would have one subcomponent for each variable. A study has concluded that such extreme decompositions do not quite perform well as the rest of the effective decomposition configurations and they should be avoided [11]. In CICC, the number of fitness evaluations depend on the number of sub-populations used in the island. Therefore, an island with higher number of subcomponents will acquire more fitness evaluations for each cycle. We would like to assign each island with the similar time for evolution and encourage a fair competition. Since each island is simultaneously evolved for complete cycles, the number of fitness evaluations cannot be exactly the same for each island if they are defined by different problem decomposition strategies. Therefore, different islands adapt to different times (fitness evaluations) because the search difficulties along different dimensions of each island are different [12]. The islands compete and collaborate with each other to optimize a problem until the termination criteria is reached.

### 2.2 Coevolution in CICC

Once both islands have been initialized with the same search space, they are evolved simultaneously for a predefined time in the usual round robin fashion of the cooperative coevolution algorithm. According to Algorithm 1, this predefined time is termed as *island-evolution time*. The island evolution time is established by the number of cycles that makes the required number of fitness evaluations for each of the two islands. Basically, a cycle is complete when all the sub-populations of an island have been cooperatively evolved for $n$ number of generations in the conventional round-robin fashion of the CC algorithm. Cooperative evaluation of individuals in the respective sub-populations is done by

concatenating the chosen individual from a given sub-population with the best individuals from the rest of the sub-populations [2].

### 2.3 Competition and Collaboration

In Stage 3 of the CICC algorithm, a simple yet efficient competition strategy is implemented. After evolution of each of the islands, through a ranking process, the algorithm marks the island with the best fitness. The island producing the minimum fitness error is the winner island and the individual with the best fitness is copied to the other islands. The migration of the best feasible solution is able to assist and motivate the other islands to compete fairly in the next phase of competition.

As an island wins, the best individuals from each of the subcomponents need to be carefully concatenated into a context vector [15]. The best solutions are then split from the context vector and are then injected into one of the runner-up island(s). The algorithm must be implemented in such a way that it ensures that the solutions are transferred without losing any genotype to phenotype mapping [13, 14].

In the conventional CC algorithm, each sub-population contains individuals that each have a unique fitness, which are cooperatively evaluated with the best solutions from the rest of the subcomponents. Taking that into consideration, there can be many distinct fitness values for the best solutions in each of the different sub-populations. Since the fitness of the best solution from the last sub-population carries a stronger solution, this fitness value is transferred (migrated) and is used to override the fitness of the best solutions of all the sub-populations of the runner-up islands.

## 3 Simulation and Analysis

In this section, we compare the performance of the competition enforced algorithm, CICC against the standalone CC algorithm for fully-separable problems.

### 3.1 Problem Decomposition Strategies

Two sets of experiments are conducted in this paper. Firstly, the best effective static decomposition strategies (near optimal) are selected from [11] and implemented as a potential island. The best effective decomposition strategies that have been identified empirically through previous studies are shown in Table 1. In this scenario, the two islands of CICC algorithm are constructed with the same problem decomposition strategies (best) which compete and collaborate to optimize a fully-separable function. It should be noted that the problem decomposition strategy for each island does not need to be the same as highlighted in [13, 12]. In the next set of experiments, we extend the study by competing two different problem decomposition strategies.

### 3.2 Benchmark Problems and Parameter Settings

The experimental results in this paper are based on eight fully-separable functions taken from previous work [11] and listed in Table 1. Functions $f_1$ and $f_2$ were selected from De Jong suite [16], and the remaining are commonly used functions for benchmarking continuous optimization algorithms defined in [17, 18, 19]. Functions $f_1$, $f_3$ and $f_7$ are uni-modal and the remaining five functions are multi-modal [11]. The total number of fitness evolutions is set to $3 \times 10^6$. The number of individuals in each of the respective sub-populations are fixed at 100.

The generalized generation gap with parent-centric crossover evolutionary algorithm (G3-PCX) [20] is used as the sub-population optimization algorithm. We use a pool size of 2 parents and 2 offspring as presented in [20]. In this generalized generation gap model selection criteria, several individuals are replaced at every generation and only those that are replaced are evaluated.

**Table 1.** A list of fully-separable and scalable benchmark problems.

| Function | Equation | Domain | Optimum | Best Decomp. [11] |
|---|---|---|---|---|
| Sphere Function | $f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$ | $[-100, 100]^n$ | $\mathbf{x}^* = \mathbf{0}, f_1(\mathbf{x}^*) = 0$ | $10 \times 100$ |
| Quadratic Funciton | $f_2(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + \mathcal{N}(0,1)$ | $[-100, 100]^n$ | $\mathbf{x}^* = \mathbf{0}, f_2(\mathbf{x}^*) = 0$ | $5 \times 200$ |
| Elliptic Function | $f_3(\mathbf{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} x_i^2$ | $[-100, 100]^n$ | $\mathbf{x}^* = \mathbf{0}, f_3(\mathbf{x}^*) = 0$ | $10 \times 100$ |
| Rastrigin's Function | $f_4(\mathbf{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$ | $[-5, 5]^n$ | $\mathbf{x}^* = \mathbf{0}, f_4(\mathbf{x}^*) = 0$ | $500 \times 2$ |
| Ackley's Function | $f_5(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$ | $[-32, 32]^n$ | $\mathbf{x}^* = \mathbf{0}, f_5(\mathbf{x}^*) = 0$ | $10 \times 100$ |
| Schwefel's Function | $f_6(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$ | $[-512, 512]^n$ | $\mathbf{x}^* = \mathbf{1}, f_6(\mathbf{x}^*) = 0$ | $500 \times 2$ |
| Different Powers | $f_7(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$ | $[-1, 1]^n$ | $\mathbf{x}^* = \mathbf{0}, f_7(\mathbf{x}^*) = 0$ | $10 \times 100$ |
| Styblinski-Tang | $f_8(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i) + 38.16599n$ | $[-5, 5]^n$ | $\mathbf{x}^* = -2.903534 \times \mathbf{1}$, $f_8(\mathbf{x}^*) = 0$ | $500 \times 2$ |

**Table 2.** Competition of same problem decomposition strategies against itself (CICC) compared with standalone CC with standalone decomposition strategies.

| Functions | Decomposition | Stats. | CC | CICC |
|---|---|---|---|---|
| $f_1$ | $10 \times 100$ | Median | 4.21e-23 | 3.76e-26 |
| | | Mean | 4.23e-23 | 8.29e-26 |
| | | StDev | 9.17e-24 | 2.69e-26 |
| $f_2$ | $5 \times 200$ | Median | 7.77e+03 | 1.39e-09 |
| | | Mean | 4.43e+08 | 1.60e-09 |
| | | StDev | 3.53e+03 | 2.73e-03 |
| $f_3$ | $10 \times 100$ | Median | 2.53e-18 | 5.50e-20 |
| | | Mean | 2.46e-18 | 5.45e-20 |
| | | StDev | 3.36e-19 | 4.06e-20 |
| $f_4$ | $500 \times 2$ | Median | 8.74e+02 | 1.40e+02 |
| | | Mean | 8.66e+02 | 1.45e+02 |
| | | StDev | 3.00e+01 | 2.02e+01 |
| $f_5$ | $10 \times 100$ | Median | 2.86e+00 | 1.26e+00 |
| | | Mean | 2.78e+00 | 1.30e+00 |
| | | StDev | 1.11e-01 | 2.40e-01 |
| $f_6$ | $500 \times 2$ | Median | 2.26e+04 | 1.30e+04 |
| | | Mean | 3.48e+04 | 1.35e+04 |
| | | StDev | 8.69e+02 | 6.17e+02 |
| $f_7$ | $10 \times 100$ | Median | 6.61e-01 | 1.00e-09 |
| | | Mean | 7.79e-01 | 4.00e-04 |
| | | StDev | 1.06e-01 | 1.01e-04 |
| $f_8$ | $500 \times 2$ | Median | 1.20e+01 | 0.00+e00 |
| | | Mean | 1.65e+02 | 0.00+e00 |
| | | StDev | 1.55e+02 | 0.00+e00 |

## 4 Results and Analyses

### 4.1 Competition Between Same Problem Decomposition Strategies

In this section, we evaluate if CICC with the best decomposition strategy competing against itself can improve the performance given by the best problem decomposition strategy used in isolation. The results are given in Table 2 that shows the median, mean and the standard deviation of the final results obtained by 25 independent runs. In this scenario, CICC implements the best problem decomposition strategies for each of the functions determined in [11]. This essentially means that the two islands are constructed having the same problem decomposition that compete with each other. The results of the standalone CC with the same problem decomposition strategy is also presented for comparison. Generally, these results in Table 2 show that the proposed CICC algorithm outperforms the standalone CC in each of the eight fully separable functions $f_1$ - $f_8$. It can be noted that CICC performed fairly well on the three unimodal functions ($f_1$,$f_3$,$f_7$) recording optimal solutions within the max $3 \times 10^6$ fitness evaluations. Additionally, the CICC algorithm performed considerably well on the multi-modal Quadratic function- $f_2$ than the standalone CC counterpart. CICC recorded better solutions for $f_4$, $f_5$, $f_6$ and outperformed its respective

**Table 3.** Competition results of two different problem decomposition strategies (CICC) compared to individual decompositions used in isolation (standalone CC)

| | | Standard CC | | |
|---|---|---|---|---|
| Functions | Stats. | $100 \times 10$ | $50 \times 20$ | CICC |
| $f_1$ | Median | 2.02e-47 | 1.56e-51 | 1.79e-79 |
| | Mean | 1.02e-47 | 3.21e-51 | 1.08e-79 |
| | StDev | 1.38e-47 | 3.94e-51 | 6.04e-80 |
| $f_2$ | Median | 5.40e-02 | 5.42e-03 | 1.96e-09 |
| | Mean | 3.68e-01 | 1.81e-03 | 3.11e-10 |
| | StDev | 5.50e-01 | 2.89e-03 | 9.83e-10 |
| $f_3$ | Median | 1.57e-45 | 3.74e-49 | 1.26e-78 |
| | Mean | 9.25e-46 | 1.37e-49 | 9.78e-78 |
| | StDev | 4.05e-46 | 1.19e-49 | 1.42e-77 |
| $f_4$ | Median | 3.53e+03 | 3.70e+03 | 3.47e+03 |
| | Mean | 3.50e+03 | 3.85e+03 | 3.86e+03 |
| | StDev | 1.39e+02 | 2.23e+02 | 5.39e+01 |
| $f_5$ | Median | 9.64e-02 | 9.78e-01 | 4.44e-16 |
| | Mean | 1.60e+00 | 1.37e+00 | 4.44e-16 |
| | StDev | 1.20e+00 | 7.52e-01 | 1.00e-16 |
| $f_6$ | Median | 1.19e+05 | 1.17e+05 | 1.06e+05 |
| | Mean | 1.26e+05 | 1.14e+05 | 1.06e+05 |
| | StDev | 3.27e+03 | 3.97e+03 | 2.37e+03 |
| $f_7$ | Median | 5.36e-03 | 1.60e+00 | 3.28e-04 |
| | Mean | 1.38e-03 | 1.48e+00 | 4.00e-04 |
| | StDev | 7.75e-05 | 2.93e-01 | 1.00e-04 |
| $f_8$ | Median | 4.82e+03 | 5.80e+03 | 4.11e+03 |
| | Mean | 4.12e+03 | 5.78e+03 | 3.75e+03 |
| | StDev | 1.89e+02 | 2.06e+02 | 1.08e+02 |

CC configurations. For the Styblinski-Tang function - $f_8$, CICC performed substantially better.

The competition of the two islands with the same decomposition scheme does not follow the motivation of the original CICC method [13, 12], where only different decomposition strategies were competing in order to exchange their unique features during evolution. In the case of competition using the same decomposition strategies, it can be noted that each island has features or solutions at different landscape of the problem that may be beneficial to the other island which may be struggling in a local optimum. These features are acquired through diversity and execution of genetic operators in the different islands that evolve in isolation for short span of time until there is comparison and then collaboration.

### 4.2 Competition Between Different Problem Decomposition Strategies

To further evaluate the efficiency of the CICC framework, we run an experiment with a set of two different problem decomposition strategies ($100 \times 10$) and ($50 \times 20$) that competes with each other to optimize a function. The results are

presented in Table 3. Once again, it is clear that using CICC to compete two different sets of problem decomposition strategies generates better solutions than the standard cooperative coevolution with standalone decompositions strategies. CICC performed better on the fully-separable functions $f_1$ - $f_8$. It generated near-optimum solutions for the uni-modal functions $f_1$ and $f_3$ and performed significantly better than the standalone CC for $f_7$. Multi-modal functions such as $f_5$ and $f_2$ were well optimized by CICC. The CICC algorithm managed to outperform the standalone CC implementations of $f_4$, $f_6$, $f_8$ and generated better quality solutions for these multi-modal functions.

In summary, it can be observed that CICC has performed well without having the need to find an optimal decomposition strategy to optimize large-scale fully-separable functions. It has generated solutions of equal quality and at times better than those found through optimal decomposition schemes. If a competition algorithm, competing problem decomposition strategies can perform better, then we do not have to empirically find the best decomposition strategy in the first instance. This can help save time and computational resources.

## 5 Conclusions and Future Work

In this paper, we have applied an island-based competitive cooperative coevolution algorithm to large-scale fully-separable continuous optimization problems. The results show that CICC can significantly improve the performance when compared to optimal problem decomposition strategies of standalone cooperative coevolution method. We found that competition and collaboration of two different suboptimal problem decomposition strategies of a fully-separable problem can also generate better solutions than the standard cooperative coevolution with standalone decomposition strategies.

Furthermore, we have shown that if competition of problem decomposition strategies can perform equally better than that of the best performing decomposition, then there is not a need to empirically find the best decomposition strategy. In other words, CICC helps us to eliminate the need for finding the optimal decomposition strategy and yet we can have solutions with similar quality.

In future work, CICC can be extended to a wide range of problems such as partially-separable functions as well as the recently introduced overlapping functions [10]. Further improvement of the results can also be achieved by implementing a multi-island CICC algorithm where more than two islands are considered.

## References


[1] T. Weise, R. Chiong, and K. Tang, "Evolutionary Optimization: Pitfalls and Booby Traps," *Journal of Computer Science and Technology (JCST)*, vol. 27, no. 5, pp. 907–936, 2012, special Issue on Evolutionary Computation.

[2] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature*. Springer, 1994, pp. 249–257.

[3] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1997.
[4] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, pp. 263–278, 1995.
[5] M. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 3, pp. 378–393, June 2014.
[6] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 2. IEEE, 2001, pp. 1101–1108.
[7] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Cooperative co-evolution with a new decomposition method for large-scale optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014*, 2014, pp. 1285–1292.
[8] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Proc. of International Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, vol. 6239. Springer Berlin / Heidelberg, 2011, pp. 300–309.
[9] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. of IEEE Congress on Evolutionary Computation*, 2010, pp. 1762–1769.
[10] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Information Sciences*, 2015.
[11] M. N. Omidvar, Y. Mei, and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *Proc. of IEEE Congress on Evolutionary Computation*, 2014, pp. 1305–1312.
[12] R. Chandra and K. Bali, "Competitive two island cooperative coevolution for real parameter global optimization," in *IEEE Congress on Evolutionary Computation*, Sendai, Japan, May 2015, pp. 93–100.
[13] R. Chandra, "Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction," *Neural Networks and Learning Systems, IEEE Transactions on*, p. In Press, 2015.
[14] ——, "Competitive two-island cooperative coevolution for training Elman recurrent networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, July 2014, pp. 565–572.
[15] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 225–239, 2004.
[16] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," 1975.
[17] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization," *gene*, vol. 7, p. 33, 2013.
[18] N. Hansen, S. Finck, R. Ros, A. Auger *et al.*, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," 2009.
[19] M. Molga and C. Smutnicki, "Test functions for optimization needs (2005)," *URL http://www. zsd. ict. pwr. wroc. pl/files/docs/functions. pdf*.
[20] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evolutionary computation*, vol. 10, no. 4, pp. 371–395, 2002.