A Cooperative Coevolutionary Algorithm with Correlation Based Adaptive Variable Partitioning

Tapabrata Ray and Xin Yao

***Abstract*—A cooperative coevolutionary algorithm (CCEA) is an extension to an evolutionary algorithm (EA); it employs a divide and conquer strategy to solve an optimization problem. In its basic form, a CCEA splits the variables of an optimization problem into multiple smaller subsets and evolves them independently in different subpopulations. The dynamics of a CCEA is far more complex than an EA and its performance can vary from good to bad depending on the separability of the optimization problem. This paper provides some insights into why CCEA in its basic form is not suitable for nonseparable problems and introduces a Cooperative Coevolutionary Algorithm with Correlation based Adaptive Variable Partitioning (CCEA-AVP) to deal with such problems. The performance of CCEA-AVP is compared with CCEA and EA to highlight its benefits. CCEA-AVP offers the possibility to deal with problems where separability among variables might vary in different regions of the search space.**

I. Introduction

Practitioners often encounter varieties of optimization problems ranging from single objective to multi-objective and unconstrained to constrained. Evolutionary algorithms have been quite successful as generic optimizers as they are easy to apply and do not require assumptions on the continuity or the slope of the function and are able to deliver a set of non-dominated solutions for multi-objective problems [1]. The performance of such EAs is known to deteriorate with an increase in the number of variables (commonly referred as the curse of dimensionality), which makes it hard to be applied to problems with larger dimensions.

In an attempt to improve the performance of EAs for problems with a large number of variables or problems with multiple subcomponents, divide and conquer strategy was introduced by Potter and DeJong [2], which has since then been referred as cooperative coevolutionary algorithm (CCEA). While the concept of evolving subcomponents of a problem independently and coadaptively sounds natural and attractive, the dynamics of CCEA is far more complex as compared to EA [3]. In a CCEA, in addition to the parameters of EA (viz. population size, number of generations, crossover and mutation rates), decisions regarding variable partitions, number of generations at each subpopulation level and collaboration strategies (single best, random etc) play a significant role in the algorithm's performance. It is quite likely that the basic implementation without appropriate choices of the above could lead to a large variation in outcome ranging from good to disappointing as echoed in the paper by Popovici and De Jong [4].

In an attempt to better understand the underlying dynamics of CCEA, empirical studies have focused on various aspects – choice of collaborators [5], [6], interaction frequency [7], sequential and parallel versions of information exchange referred as update timing [8], disconnection between the external goal and the internal behavior of CCEA [3], etc. All these studies have highlighted the complexity of the dynamics of CCEA.

On a parallel front, CCEA and its variants such as coevolutionary particle swarms [9], coevolutionary differential evolution [10], cooperative coevolutionary models based on self-adaptive neighborhood search differential evolution [11], and the original study on cooperative coevolutionary algorithm [2] have been applied to a number of single and multi-objective, separable and nonseparable benchmark problems.

It is interesting to note that in these applications very little attention was paid to the selection of the coevolutionary parameters and yet good results were obtained. The purpose of this paper is to investigate the performance of the basic CCEA and the effects on problem size, number of partitions and number of generations allocated to evolve subpopulations on a number of separable and nonseparable problems. The second objective of this paper is to introduce a cooperative coevolutionary algorithm with adaptive variable partitioning (CCEA-AVP) in an attempt to deal with nonseparable problems.

The details of the basic CCEA are described in Section II, while the numerical results are presented in Section III to highlight the deficiencies of the above CCEA. In Section IV, CCEA-AVP (Cooperative Coevolutionary Algorithm with Adaptive Variable Partitioning) is introduced and its performance is studied using several examples. The summary of preliminary findings is listed in Section V.

II. Background

Before getting on with the details of the proposed CCEA, it is necessary to define the following terms.

A function with N-variables said to be *separable* if

$$f(x) = \sum_{i=1}^{m} f_i(x), \quad m > 1$$

where each subfunction $f_i(x)$ depends on atmost $k(< N)$ variables and number of functions that depend on a particular variable is atmost one [12].

Tapabrata Ray is with School of Aerospace, Civil and Mechanical Engineering, University of New South Wales at Australian Defence Force Academy, Canberra ACT, Australia. email: t.ray@adfa.edu.au

Xin Yao is with The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science,University of Birmingham, B15 2TT, United Kingdom. email: X.Yao@cs.bham.ac.uk

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

In a CCEA, a solution vector is split into a number of subcomponents without any overlap among their decision variables. Each subcomponent is then evolved in a subpopulation for a number of generations. The term "subevolve 20" used later in the paper refers to each subpopulation evolving over 20 generations prior to any communication with the next subpopulation. The variable values of the best solution of a subpopulation are used while evolving other subpopulations and is referred as single best collaboration strategy.

To be consistent with comparisons between CCEA and EA, we have used the same underlying evolutionary mechanism in both. The EA used in this study is Non-dominated Sorting Genetic Algorithm (NSGA-II) [1], which for single objective optimization problems ranks the solutions based on objective function value, and treats a feasible individual better than an infeasible one. The recombination mechanism is controlled through simulated binary crossover and polynomial mutation. The pseudo code of the basic CCEA is presented in Algorithm 1.

Algorithm 1 Basic CCEA

Require: N_G (Number of Generations), N_V (Number of variables for the problem), N_P (Population size), N_S (Number of generations which the subpopulations evolve independently), CR (Crossover Rate), MR (Mutation Rate), Distribution index of Crossover and Distribution index of Mutation.

1: Initialize Population (P), $P_{i,j}$ denotes the j^{th} variable of i^{th} solution, $i \in (1, \ldots, N_P)$ and $j \in (1, \ldots, N_V)$
2: **for** generations = 1 to N_G **do**
3: Evaluate (P)
4: Partition population P into K subpopulations $P1, P2, \ldots, PK$ where $PI_{i,j}$ is the first subpopulation with $i \in (1, \ldots, N_P)$ and $j \in (1, \ldots, \frac{N_V}{K})$.
5: Select the best solution obtained in Step 3 and store it as X_{Best}
6: **for** $I = 1$ to K **do**
7: Create a Subpopulation: Make N_P copies of X_{Best} and replace the appropriate variables using PI
8: Evolve the Subpopulation over N_s generations.
9: Update X_{Best}
10: **end for**
11: **end for**

For all the numerical examples studied in this paper, the probability of crossover was set to 1.0, probability of mutation as 0.1, distribution index of crossover as 15 and the distribution index of mutation as 20.

III. Numerical Examples

The test functions chosen for this study are the ones commonly used in cooperative coevolutionary studies. The separable test functions are Rastrigin and Schwefel; the nonseparable functions are Rosenbrock and Ackley. For all the CCEA runs, the variables are divided into 2 subpopulations unless specifically mentioned. The details of the test functions used in this study are listed below.

1) Rastrigin: The function is defined as

$$\text{Minimize} \quad f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

$$-5.12 \leq x_i \leq 5.12$$

The function is separable but has numerous local minima.

2) Schwefel: The function is defined as

$$\text{Minimize} \quad f(x) = 418.9829n - \sum_{i=1}^{n} \left(x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

$$-500 \leq x_i \leq 500$$

The function is separable and multimodal. There are many local minima and the global optimum is located far away from the second best optimum, which often acts as a trap.

3) Rosenbrock: The function is defined as

$$\text{Minimize} \quad f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

$$-30 \leq x_i \leq 30$$

The function is nonseparable [13].

4) Ackley: The function is defined as

$$\text{Minimize} \quad f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) -$$

$$\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

$$-32 \leq x_i \leq 32$$

The function is nonseparable and multimodal.

A. Case 1: Performance on 2D Test functions

The results of EA and CCEA for the 2D test functions with a population size of 20 evolved over 500 generations are presented in the following figures. The averaged objective function values over 20 independent runs are plotted against the number of function evaluations in Figures 1–4. One can observe that EA performs consistently better than CCEA for all four problems although for its rate of convergence is slow at initial stages for Rosenbrock.

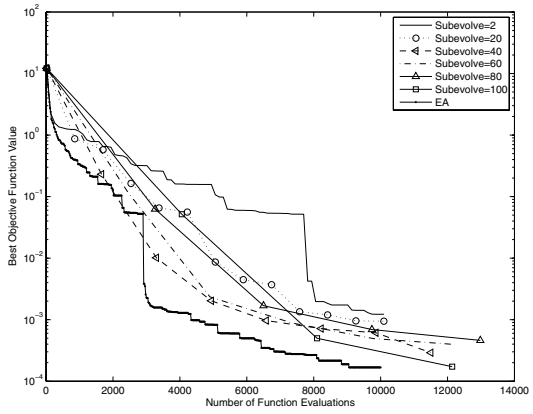

Fig. 1: Performance of CCEA and EA on 2D Rastrigin

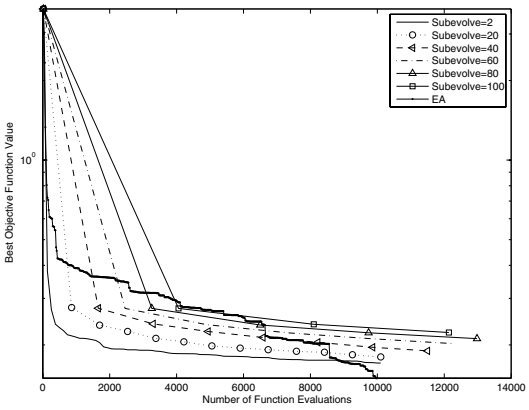

Fig. 3: Performance of CCEA and EA on 2D Rosenbrock

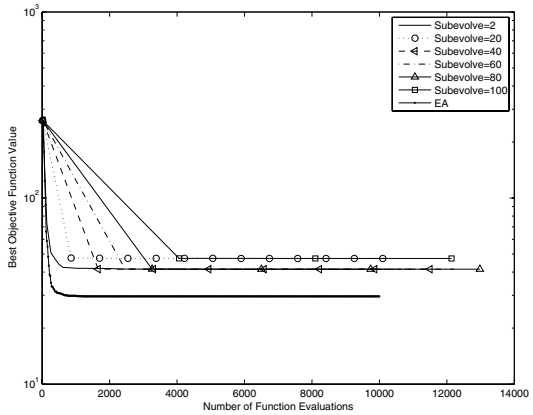

Fig. 2: Performance of CCEA and EA on 2D Schwefel

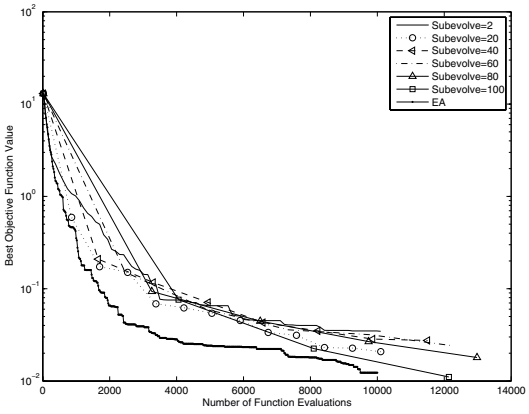

Fig. 4: Performance of CCEA and EA on 2D Ackley

B. Case 2: Performance on 50D and 100D test functions

Since CCEAs are claimed to be attractive for problems with a large number of variables, we conducted tests on 50D and 100D problems using a population size of 100 evolving over 1000 generations. The results for 50D and 100D are presented in Figures 5–12. It is interesting to observe that CCEA performs better than EA for all the problems (50D and 100D) independent of the number of generations allocated to evolve the subpopulations. Potter and DeJong [2] observed the performance of coevolutionary algorithm was better for Ackley function in 30D even though they employed each variable in separate subpopulation and a binary coding for the variables. As we expect a coevolutionary algorithm to face difficulty for nonseparable problems (Rosenbrock and Ackley), a test on highly nonseparable function Shifted Rotated Rastrigin was conducted.

C. Case 3: Performance on Shifted Rotated 50D Rastrigin Function

Shifted Rotated Rastrigin function is a function F10 from CEC-2005 benchmark [14]. The problem is multimodal and nonseparable, with a large number of local optima.

$$\text{Minimize} \quad f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

$$-5.12 \leq x_i \leq 5.12, z = (x - o) \times M$$

where o is the shifted global optimum and M is the linear Transformation Matrix (Rastrigin_M_D50).

Results of CCEA runs with 2 subpopulations are presented in Figure 13 and with 10 subpopulations are presented in Figure 14. It is clear from Figure 14 that CCEA with subpopulations evolving for 2 generations performs better than subpopulations evolving independently for 100 generations. It is also worth taking note from Figure 13 that EA performs better than CCEA or vice versa depending on how long the algorithm is run.

D. Case 4: Constrained optimization problem

In order to further observe the performance of CCEA on constrained optimization problems, a test was conducted using the test function G2 [15] in 50D. The details of the test function are provided below.

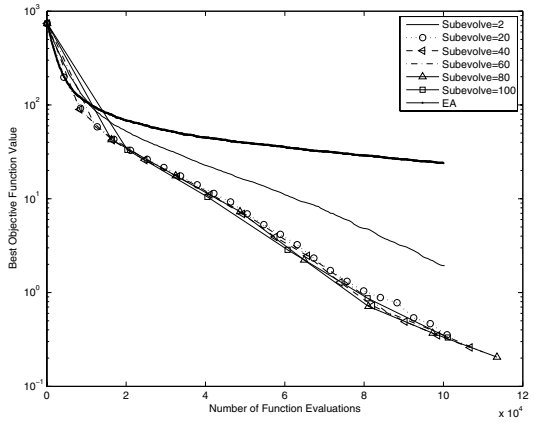

Fig. 5: Performance of CCEA and EA on 50D Rastrigin

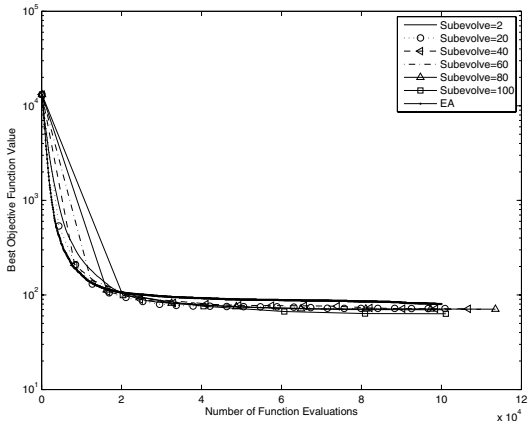

Fig. 7: Performance of CCEA and EA on 50D Rosenbrock

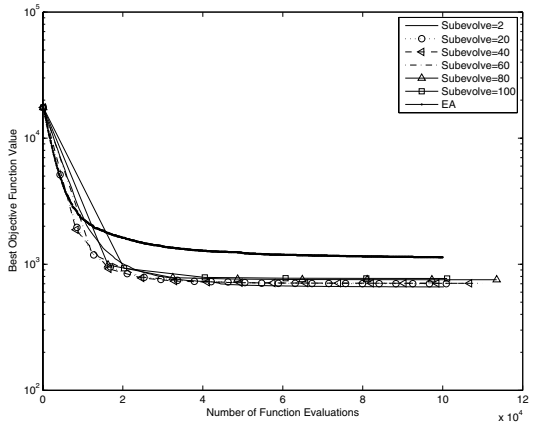

Fig. 6: Performance of CCEA and EA on 50D Schwefel

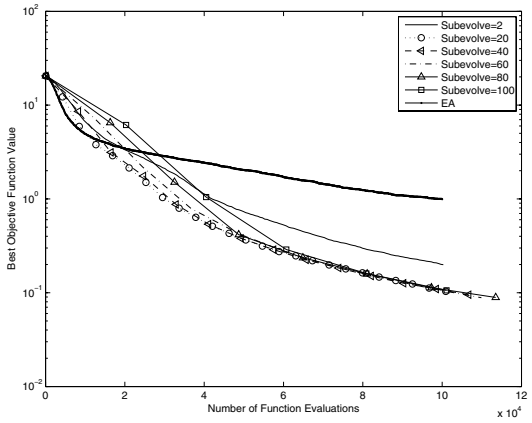

Fig. 8: Performance of CCEA and EA on 50D Ackley

$$\text{Maximize} \quad f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

subject to

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

where $0 \leq x_i \leq 10$ $(i = 1, \ldots, n)$. The global maximum is unknown and the function is nonlinear; the best reported value for 20D is $f(x^*) = 0.803619$.

The study was conducted using 2 fixed subpopulations and the results are shown in Figure 15. It is clear from the results that CCEA is worse off as compared with EA and subevolve 2 is better than subevolve 100. The above studies highlight that the basic form of CCEA has difficulties in solving problems, which are nonseparable (Shifted Rotated Rastrigin and G2) and the choices of the number of partitions and the number of generations allocated to evolve subpopulations have significant effect on the algorithm's performance. This is the motivation to develop a Cooperative Coevolutionary Algorithm with Adaptive Variable Partitioning (CCEA-AVP) which is described in following Section.

IV. Proposed CCEA with adaptive variable partitioning based on correlation

In order to deal with nonseparable problems and to alleviate the problems associated with the selection of number of subpopulations and variable partitioning, a correlation based variable partitioning scheme was designed within CCEA which is referred as CCEA-AVP. In CCEA-AVP, the first M (currently set to 5) generations, all variables evolve within a single partition (single population) similar to an EA. At the $(M+1)^{th}$ generation, the correlation matrix is computed based on the top 50% solutions of the population and the variables are partitioned into multiple subpopulations. The variables having a correlation coefficient greater than a prescribed threshold (0.6 used in this study) are grouped together subject to a predefined maximum number of subpopulations (set to 10 in this study). Correlation based variable partitioning is repeated at every generation subsequently.

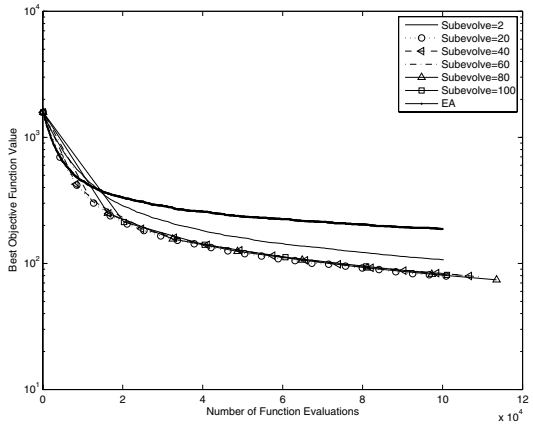

Fig. 9: Performance of CCEA and EA on 100D Rastrigin

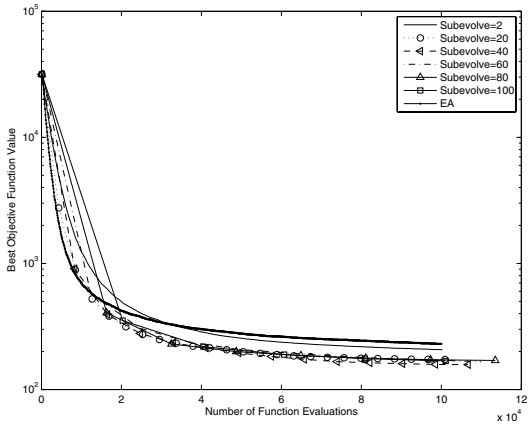

Fig. 11: Performance of CCEA and EA on 100D Rosenbrock

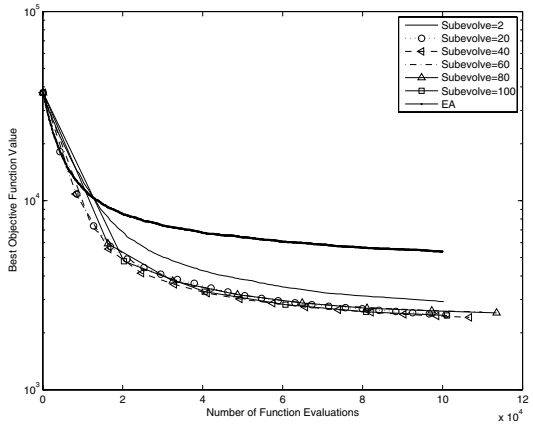

Fig. 10: Performance of CCEA and EA on 100D Schwefel

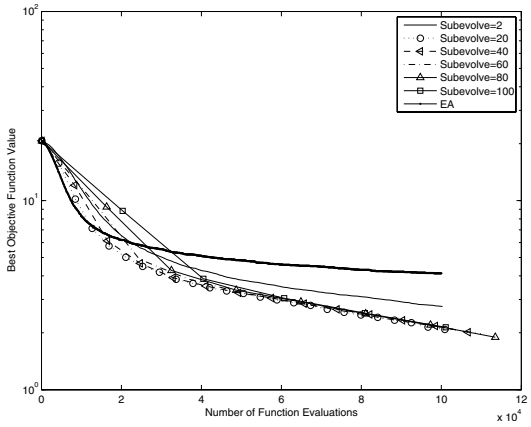

Fig. 12: Performance of CCEA and EA on 100D Ackley

In order to test the efficacy of the proposed CCEA-AVP algorithm for nonseparable problems, the 50D G2 function was first selected as a candidate. The results of CCEA-AVP averaged over 20 runs are listed in Table I. It is clear from the results in Table I (CCEA-AVP) and Figure 15 (CCEA), that CCEA-AVP is better than basic CCEA and marginally worse as compared to an EA. It is also worth mentioning that CCEA-AVP incurs an additional cost of computing correlation matrices. It can also be observed from Figure 16, that CCEA almost performs like an EA with single partition in most of the generations due to the nonseparable nature of the problem.

TABLE I: Results of CCEA-AVP and EA for 50D G2

	Best	Worst	Average
Subevolve 2	-0.7835	-0.6569	-0.7419
Subevolve 20	-0.8104	-0.7305	-0.7739
Subevolve 40	-0.8103	-0.7066	-0.7743
Subevolve 60	-0.8101	-0.6456	-0.7733
Subevolve 80	-0.8015	-0.6039	-0.7537
Subevolve 100	-0.8034	-0.5200	-0.7576
EA	-0.8120	-0.7319	-0.7775

The next example is the highly nonseparable Shifted Rotated Rastrigin function. The performance of CCEA-AVP for this function is listed in Table II. Average results of CCEA-AVP are better than CCEA with 10 subpopulations while the average performance is comparable with a 2 partition CCEA and EA. The performance of CCEA-AVP is identical to a single population EA provided all variables in the population have a correlation coefficient greater than the threshold (0.6 used in the study) at all generations. The number of partitions of the shifted rotated Rastrigin function is significantly less than the number of partitions of the original Rastrigin function indicating a higher variable interaction.

The third example is a separable problem Rastrigin 50D where it is interesting to observe that CCEA-AVP generates comparable results with EA and its performance is worse when compared to a 2 partition CCEA, as seen from Table III. It is counter-intuitive to observe that Subevolve 100 performs worse than Subevolve 20 for this separable problem. A look into Figure 16 explains that the algorithm partitioned variables into 7 subsets on an average and evolving each partition over 100 generations amounts to a

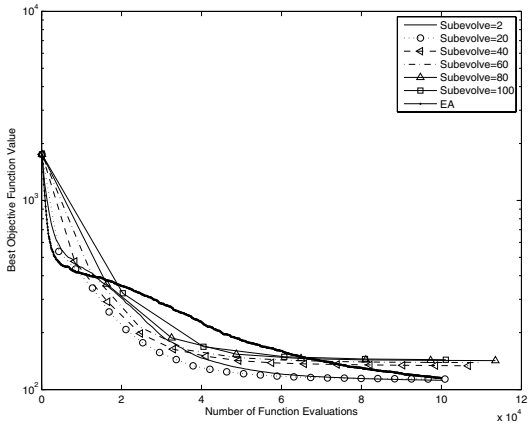

Fig. 13: Performance of CCEA with 2 subpopulations and EA on 50D Rotated Shifted Rastrigin

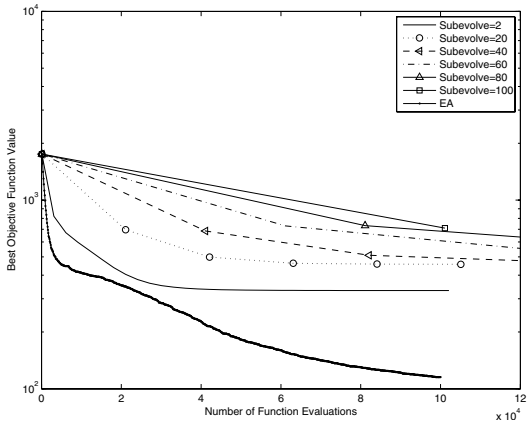

Fig. 14: Performance of CCEA with 10 subpopulations and EA on 50D Rotated Shifted Rastrigin

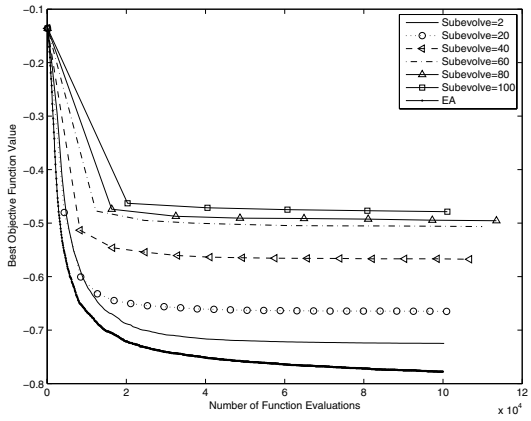

Fig. 15: Performance of CCEA and EA on 50D G2

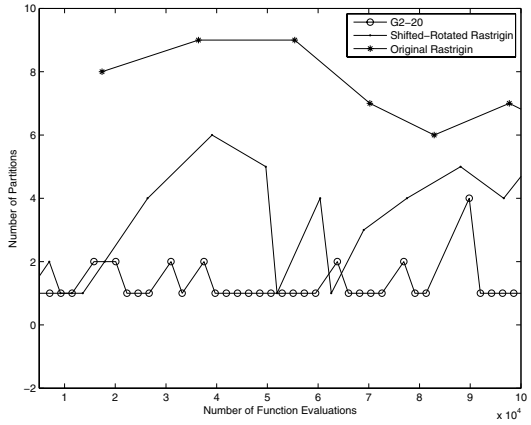

Fig. 16: Number of Partitions created by CCEA-AVP across generations

significant number of function evaluations in each generation and it soon ran out of the maximum allowable number of function evaluations.

V. Summary and Conclusions

This paper investigates the performance of a Single Best Collaboration Strategy Coevolutionary Algorithm for separable and nonseparable test functions. The results of CCEA are compared with an EA using identical evolution mechanism and the same random seeds. The results suggest that the basic CCEA may not be suitable for nonseparable problems and the number of predefined partitions and the number of generations allocated to evolve subpopulations independently play a significant role in its performance. It is imperative that the use of the basic CCEA as a generic optimizer for various classes of problems is likely to produce results ranging from good to poor when compared with EA.

In order to deal with nonseparable optimization problems, CCEA with adaptive variable partitioning (CCEA-AVP) based on correlation was developed. CCEA-AVP partitions the variables into subpopulations based on observed correlation and hence does not require apriori partition rules. The results of CCEA-AVP for nonseparable problems indicate that it is better than the basic CCEA and has comparable average performance as EAs. For the separable problems, CCEA-AVP is better than an EA (subevolve 2 and subevolve 20). The study has also highlighted that even in separable problems, a large number of partitions coupled with large number of subevolutions within each partition could blow out the limit of the computational budget. The result would be poor final solutions due to few generations allocated for the top level evolution process. Since a problem might have different levels of variable correlation in different regions of the search space, CCEA-AVP provides an opportunity to adaptively capture and exploit their relationships. CCEA-AVP would behave identical to a single population EA if all the variables have correlations greater than the user defined threshold. Since separability or nonseparability of an optimization problem cannot be identified apriori, CCEA-AVP offers some hope to deal with black box functions within a cooperative coevolutionary framework. It is also worth high-

TABLE II: Results of CCEA-AVP and EA for Shifted Rotated Rastrigin 50D

	Best	Worst	Average
Subevolve 2	101.96	184.73	123.69
Subevolve 20	93.757	152.59	116.65
Subevolve 40	97.274	146.58	118.27
Subevolve 60	91.082	155.34	116.83
Subevolve 80	102.11	167.88	122.06
Subevolve 100	98.472	209.33	138.84
EA	85.70	169.56	115.27

TABLE III: Results of CCEA-AVP and EA for Original Rastrigin 50D

	Best	Worst	Average
Subevolve 2	0.9430	15.071	3.3412
Subevolve 20	4.5265	31.995	16.872
Subevolve 40	8.0651	95.015	44.346
Subevolve 60	7.5714	102.677	37.143
Subevolve 80	24.335	122.933	56.550
Subevolve 100	44.497	154.75	79.041
EA	18.44	29.19	23.98

lighting that the performance of a CCEA and CCEA-AVP is also dependent on the underlying recombination operator. For a set of positively or negatively correlated variables evolving in a subpopulation of CCEA-AVP, specific operators could be designed to sample along the directions of interest. Further studies are underway to identify the changes in correlation patterns, the effects of partitioning frequency (currently done every generation) and the performance of CCEA-AVP on problems with hundreds of variables.

REFERENCES

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, pp. 182–197, 2002.

[2] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *PPSN III: Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature*, 1994, pp. 249–257.

[3] E. Popovici and K. De Jong, "Relationships between internal and external metrics in co-evolution," in *IEEE Congress on Evolutionary Computation CEC 2005*, vol. 3, Sept. 2005, pp. 2800–2807 Vol. 3.

[4] ——, "Understanding cooperative co-evolutionary dynamics via simple fitness landscapes," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005, pp. 507–514.

[5] R. P. Wiegand, W. C. Liles, and K. A. D. Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *In Proceedings from the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2001, pp. 1235–1242.

[6] J. M. McNamara, Z. Barta, L. Fromhage, and A. I. Houston, "The coevolution of choosiness and cooperation," *Nature*, vol. 451, no. 7175, pp. 189–192, 2008.

[7] E. Popovici and K. De Jong, "The effects of interaction frequency on the optimization performance of cooperative coevolution," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006, pp. 353–360.

[8] ——, "Sequential versus parallel cooperative coevolutionary algorithms for optimization," in *IEEE Congress on Evolutionary Computation CEC 2006*, 0-0 2006, pp. 1610–1617.

[9] C. Tan, C. Goh, K. Tan, and A. Tay, "A cooperative coevolutionary algorithm for multiobjective particle swarm optimization," in *IEEE Congress on Evolutionary Computation CEC 2007*, Sept. 2007, pp. 3180–3186.

[10] E. Popovici and K. De Jong, "A coevolutionary differential evolution algorithm for constrained optimization," in *Proc. of Third International Conference on Natural Computation*, 2007, pp. 51–57.

[11] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985 – 2999, 2008, nature Inspired Problem-Solving. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0C-4S33N9J-1/2/57e2e380756ebd5eb98adc73644a49d6

[12] M. J. Streeter, "Upper bounds on the time and space complexity of optimizing additively separable functions," in *Genetic and Evolutionary Computation GECCO 2004*, 2004, pp. 186–197.

[13] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating Evolutionary Algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.

[14] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur, India, Tech. Rep., 2005.

[15] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.