Vector Evaluated Differential Evolution for Multiobjective Optimization

K.E. Parsopoulos[1,3], D.K. Tasoulis[1,3], N.G. Pavlidis[1,3], V.P. Plagianakos[2,3] and M.N. Vrahatis[1,3]
[1]Department of Mathematics, University of Patras, GR–26110 Patras, Greece
[2]Department of Information and Communication Systems Engineering,
University of the Aegean, GR–83200 Samos, Greece
[3]University of Patras Artificial Intelligence Research Center (UPAIRC)
Email: {kostasp,dtas,npav,vpp,vrahatis}@math.upatras.gr

***Abstract*— A parallel, multi–population Differential Evolution algorithm for multiobjective optimization is introduced. The algorithm is equipped with a domination selection operator to enhance its performance by favoring non–dominated individuals in the populations. Preliminary experimental results on widely used test problems are promising. Comparisons with the VEGA approach are provided and discussed.**

I. Introduction

Multiobjective Optimization (MO) problems consist of several competing and incommensurable objective functions. Such problems are frequently encountered in numerous scientific and engineering applications. The need for the concurrent minimization of more than one objective functions, renders the use of Evolutionary Algorithms (EAs) particularly attractive. In contrast to traditional gradient–based techniques, evolutionary algorithms operate on a set of potential solutions of the problem. Thus, EAs are capable of detecting several solutions of an MO problem in a single run [1], [2], [3], [4], [5], [6]. These solutions are called *Pareto optimal*, and each corresponds to a different trade–off among the objective functions. Typically, a large number of Pareto optimal solutions exist.

Differential Evolution (DE) is an efficient evolutionary optimization algorithm. It has been successfully applied on a plethora of applications [7], [8], [9], [10], [11], [12]. Like other EAs, DE can be easily parallelized [8]. Besides a reduction in execution time, the parallel computation of solutions of an MO problem can also yield a better representation of the possible outcomes, thereby enhancing the performance of the algorithm [4].

This paper introduces a multi–population variant of DE, named *Vector Evaluated Differential Evolution* (VEDE), which is inspired by the *Vector Evaluated Genetic Algorithm* (VEGA) approach [3], [13], [14], [15]. In VEDE, each population is evaluated using one of the objective functions of the problem under consideration. Information sharing among the populations takes place through the migration of the best individuals. The performance of a parallel version of VEDE, which incorporates a domination selection scheme, is investigated on widely used test problems and compared to the VEGA approach.

The rest of the paper is organized as follows; in Section II the basic concepts of MO and the Differential Evolution algorithm, are briefly presented. Section III, is devoted to the description of the proposed algorithm, as well as, to the presentation of the experimental results. A short discussion of the parallel implementation is also included in this section. Finally, the paper ends with a synopsis in Section IV.

II. Background Material

A. Basic Concepts of Multiobjective Optimization

Let $S \subset \mathbb{R}^n$ be an n–dimensional search space, and let k objective functions,

$$f_i(x) : S \to \mathbb{R}, \quad i = 1, \ldots, k, \tag{1}$$

be defined over S. Further assume,

$$g_j(x) \leqslant 0, \quad j = 1, \ldots, m,$$

to be m inequality constraints. Then the MO problem can be stated as finding a vector,

$$x^* = (x_1^*, x_2^*, \ldots, x_n^*)^\top \in S,$$

that satisfies the constraints and minimizes the function,

$$\mathbf{f}(x) = [f_1(x), f_2(x), \ldots, f_k(x)] : \mathbb{R}^n \to \mathbb{R}^k. \tag{2}$$

The goal of MO is to compute a set of Pareto optimal solutions to the aforementioned problem.

Let $u = (u_1, \ldots, u_k)$, and $v = (v_1, \ldots, v_k)$, be two vectors. Then, u *dominates* v if and only if,

$$u_i \leqslant v_i, \quad i = 1, \ldots, k,$$

and

$$u_i < v_i, \quad \text{for at least one } i.$$

This property is known as *Pareto dominance* and it is used to define the Pareto optimal points. A solution, x, of the MO problem is said to be *Pareto optimal* if and only if, there does not exist another solution y, such that $\mathbf{f}(y)$ dominates $\mathbf{f}(x)$. The set of all Pareto optimal solutions of an MO problem is called *Pareto optimal set* and is denoted as $\mathcal{P}^*$. The set,

$$\mathcal{PF}^* = \left\{ (f_1(x), \ldots, f_k(x))^\top \;\middle|\; x \in \mathcal{P}^* \right\},$$

is called *Pareto front*. A Pareto front $\mathcal{PF}^*$ is *convex* if and only if, there exists $w \in \mathcal{PF}^*$, such that,

$$\lambda\|u\| + (1-\lambda)\|v\| \geqslant \|w\|, \quad \forall\ u, v \in \mathcal{PF}^*,\ \forall\ \lambda \in (0,1).$$

Respectively, it is *concave* if and only if, there exists $w \in \mathcal{PF}^*$, such that,

$$\lambda\|u\| + (1-\lambda)\|v\| \leqslant \|w\|, \quad \forall\ u, v \in \mathcal{PF}^*,\ \forall\ \lambda \in (0,1).$$

A Pareto front can be convex, concave or partially convex and/or concave and/or discontinuous.

B. The Differential Evolution Algorithm

Let $S \subset \mathbb{R}^n$ be the search space of the problem under consideration. Then, the Differential Evolution (DE) algorithm utilizes *NP*, n–dimensional vectors,

$$X_i = (x_{i1}, \ldots, x_{in})^\top \in S, \quad i = 1, \ldots, \mathit{NP},$$

as a population for each iteration, called a *generation*, of the algorithm. The initial population is usually taken to be uniformly distributed in the search space. At each generation, two operators, namely *mutation* and *crossover* (recombination), are applied on each individual, thus producing the new population. Then, a *selection* phase takes place, where each individual of the new population is compared to the corresponding individual of the old population, and the best between them is selected as a member of the population in the next generation.

According to the mutation operator, for each individual, $X_i^{(G)}$, $i = 1, \ldots, \mathit{NP}$, at generation G, a *mutant vector*,

$$V_i^{(G+1)} = \left(v_{i1}^{(G+1)}, v_{i2}^{(G+1)}, \ldots, v_{in}^{(G+1)}\right)^\top,$$

is determined using one of the following equations [16], [17]:

$$V_i^{(G+1)} = X_{r_1}^{(G)} + F\left(X_{r_2}^{(G)} - X_{r_3}^{(G)}\right), \quad (3)$$

$$V_i^{(G+1)} = X_{\text{best}}^{(G)} + F\left(X_{r_1}^{(G)} - X_{r_2}^{(G)}\right), \quad (4)$$

$$V_i^{(G+1)} = X_i^{(G)} + F\left(X_{\text{best}}^{(G)} - X_i^{(G)}\right) + F\left(X_{r_1}^{(G)} - X_{r_2}^{(G)}\right), \quad (5)$$

$$V_i^{(G+1)} = X_{\text{best}}^{(G)} + F\left(X_{r_1}^{(G)} - X_{r_2}^{(G)}\right) + F\left(X_{r_3}^{(G)} - X_{r_4}^{(G)}\right), \quad (6)$$

$$V_i^{(G+1)} = X_{r_1}^{(G)} + F\left(X_{r_2}^{(G)} - X_{r_3}^{(G)}\right) + F\left(X_{r_4}^{(G)} - X_{r_5}^{(G)}\right), \quad (7)$$

where, $X_{\text{best}}^{(G)}$ is the best individual of the population at generation G; $F > 0$ is a real parameter, called *mutation constant*, which controls the amplification of the difference between two individuals so as to avoid search stagnation; and r_1, r_2, r_3, r_4, r_5, are mutually different integers, randomly selected from the set $\{1, 2, \ldots, i-1, i+1, \ldots, \mathit{NP}\}$.

Following the mutation phase, the crossover (recombination) operator is applied on the population. For each mutant vector, $V_i^{(G+1)}$, an index $\text{rnbr}(i) \in \{1, 2, \ldots, n\}$ is randomly chosen, and a *trial vector*,

$$U_i^{(G+1)} = \left(u_{i1}^{(G+1)}, u_{i2}^{(G+1)}, \ldots, u_{in}^{(G+1)}\right)^\top,$$

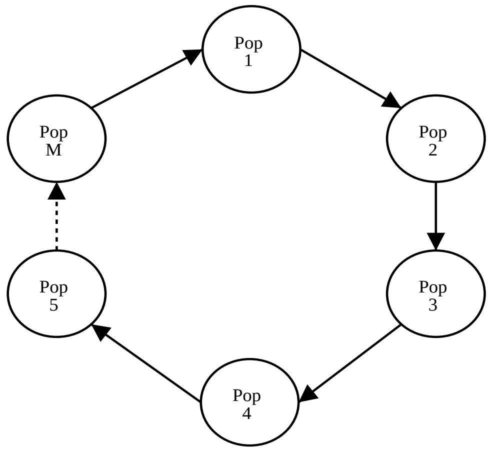

Fig. 1. The ring topology.

is generated, with

$$u_{ij}^{(G+1)} = \begin{cases} v_{ij}^{(G+1)}, & \text{if } (\text{randb}(j) \leqslant \mathit{CR}) \text{ or } (j = \text{rnbr}(i)), \\ x_{ij}^{(G)}, & \text{if } (\text{randb}(j) > \mathit{CR}) \text{ and } (j \neq \text{rnbr}(i)), \end{cases}$$

where, $j = 1, 2, \ldots, n$; $\text{randb}(j)$ is the j–th evaluation of a uniform random number generator within $[0,1]$; and *CR* is a user–defined *crossover constant* in the range $[0,1]$ [16], [17]. In other words, the trial vector consists of some of the components of the mutant vector, and at least one of the components of a randomly selected individual of the population (i.e. the individual with index $\text{rnbr}(i)$).

To decide whether the vector $U_i^{(G+1)}$ should be a member of the population comprising the next generation, it is compared to the corresponding vector $X_i^{(G)}$. Thus, if f denotes the objective function under consideration, then,

$$X_i^{(G+1)} = \begin{cases} U_i^{(G+1)}, & \text{if } f\left(U_i^{(G+1)}\right) < f\left(X_i^{(G)}\right), \\ X_i^{(G)}, & \text{otherwise}. \end{cases} \quad (8)$$

III. Proposed Algorithm and Experimental Results

A. The Proposed Algorithm

Vector Evaluated Differential Evolution (VEDE) is a multi–population DE approach, inspired by VEGA [3], [13], [14], [15]. Specifically, a number, M, of populations are considered in a prespecified ring topology, as depicted in Fig. 1. Each population is evaluated using as fitness function, one of the objective functions of the problem at hand. If k is the number of the objective functions, and $k < M$, then the i–th population is evaluated according to the j–th objective function, where,

$$j \equiv \begin{cases} i \bmod k, & \text{if} \quad i \neq rk, \quad r = 1, 2, \ldots, \\ k, & \text{otherwise}, \end{cases}$$

and $i = 1, \ldots, M$.

In every generation, the best individual, $X_{i,\text{best}}^{(G)}$, of the i–th population, migrates to the $(i+1)$–th population of the ring. Then, the $(i+1)$–th population uses $X_{i,\text{best}}^{(G)}$ as the best individual to produce its mutant vectors at generation $(G+1)$. Obviously, only the DE operators that use the best individual in the mutations, i.e. the variants described in Eqs. (4), (5), and (6), can take full advantage of this information exchange procedure.

Moreover, a domination selection procedure, similar to that of Abbass [18], is applied, i.e. instead of using the plain DE selection operator of Eq. (8), we use the following one:

$$X_i^{(G+1)} = \begin{cases} U_i^{(G+1)}, & \text{if } \mathbf{f}\left(U_i^{(G+1)}\right) \text{ dominates } \mathbf{f}\left(X_i^{(G)}\right), \\ X_i^{(G)}, & \text{otherwise,} \end{cases}$$

where, $\mathbf{f}$ is the vector function defined in Eq. (2). This selection scheme favors non–dominated individuals in the population and it has proved to perform better in practice.

VEDE can be easily parallelized. The populations can be distributed in several machines, with migrations taking place from node to node. For this purpose, the Parallel Virtual Machine (PVM) was used [8], [19].

A high level description of the parallel algorithmic scheme follows:

```
At the master node
1. Spawn $M$ populations,
   each one on a different processor.
2. For each generation
3.     Receive an individual from each population.
4.     Send the individual that will migrate to the
       next population of the ring topology.
At each population
1. For each generation
2.     Perform a complete DE generation.
3.     Send the best individual to the master node.
4.     Receive a migrated individual
       and assign it to the best individual.
```

B. Presentation of Experimental Results

Four well–known MO benchmark problems were used as a first step in the investigation of VEDE's performance. Each test problem consists of two objective functions of the form

$$f_1(x_1) = x_1, \tag{9}$$
$$f_2(x_1,\ldots,x_n) = g(x_2,\ldots,x_n) \times h(f_1,g). \tag{10}$$

Specifically, we considered the following problems [6]:

TEST PROBLEM 1. This test problem is defined as:

$$f_1(x_1) = x_1, \tag{11}$$
$$g(x_2,\ldots,x_n) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \tag{12}$$
$$h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}}, \tag{13}$$

with $n = 30$ and $x_i \in [0,1]$. The Pareto front for this problem is convex.

TABLE I

THE CHARACTERISTICS OF THE SYSTEM USED.

Characteristic	Description
Number of CPUs	2 to 5
CPU Type	Intel Celeron 900-MHz
Memory	256-MB per machine
Operating System	Red Hat Linux 8.0
Communication Network	Fast Ethernet 100-Mbps
Communication Library	PVM 3.4.4
Compiler	GNU Compiler Collection (gcc) 3.2.2

TEST PROBLEM 2. This test problem is the non–convex counterpart to Test Problem 1. It is defined as:

$$f_1(x_1) = x_1, \tag{14}$$
$$g(x_2,\ldots,x_n) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \tag{15}$$
$$h(f_1,g) = 1 - \left(\frac{f_1}{g}\right)^2, \tag{16}$$

with $n = 30$ and $x_i \in [0,1]$.

TEST PROBLEM 3. This test problem is defined as:

$$f_1(x_1) = x_1, \tag{17}$$
$$g(x_2,\ldots,x_n) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \tag{18}$$
$$h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1), \tag{19}$$

with $n = 30$ and $x_i \in [0,1]$. The Pareto front consists of several convex parts.

TEST PROBLEM 4. This test problem is defined as:

$$f_1(x) = x_1, \tag{20}$$
$$\begin{aligned} g(x_2,\ldots,x_n) = {} & 1 + 10(n-1) + \\ & + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right), \end{aligned} \tag{21}$$
$$h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}}, \tag{22}$$

and it has 21^9 local Pareto fronts.

All experiments were performed in parallel, using the PVM communication library. The key characteristics of the system used, are reported in Table I. In addition to the reported hardware, a Pentium III machine with 512-MB of memory, running under Red Hat Linux 8.0, was used as a server.

For the maintenance of the Pareto optimal set, the archiving technique described in [20], which uses an external archive, was employed.

The obtained results were compared to that of VEGA, reported in *http://www.tik.ee.ethz.ch/∼zitzler/testdata.html*. For this purpose, two established measures, namely the $\mathcal{C}$ measure [6], [21], and the $\mathcal{V}$ measure [21], [22] were employed.

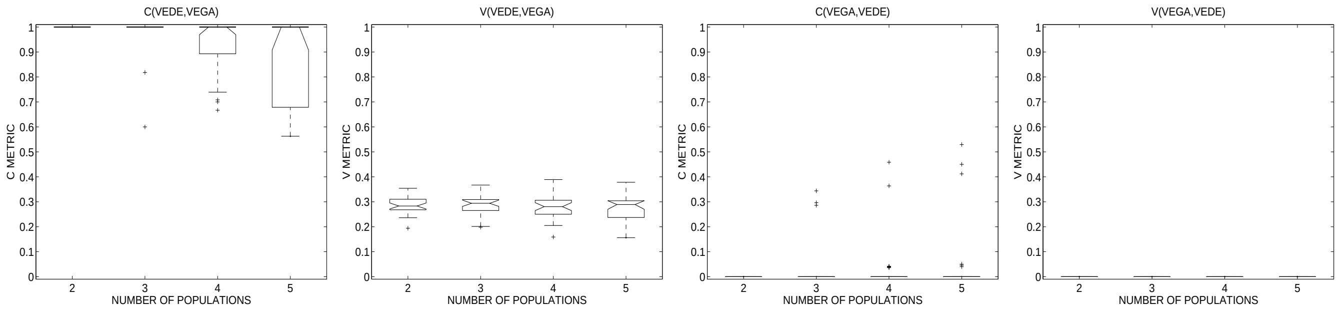

Fig. 2. Results of VEDE1 for the Test Problem 1.

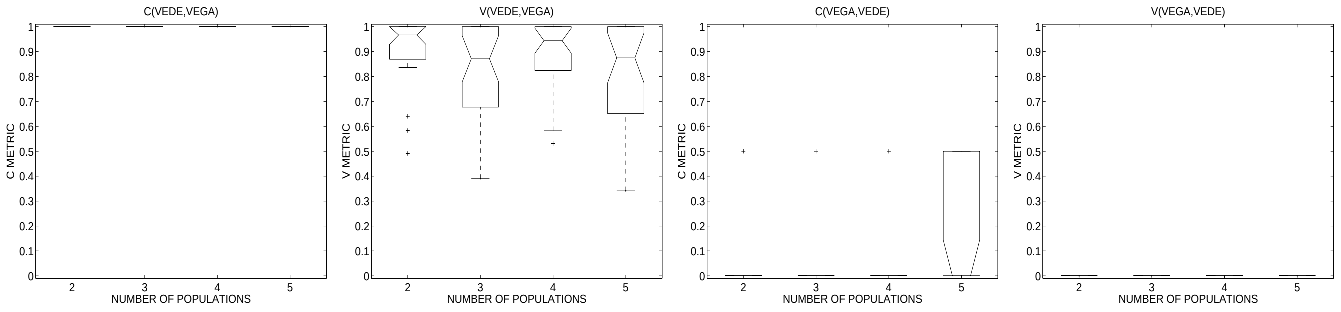

Fig. 3. Results of VEDE1 for the Test Problem 2.

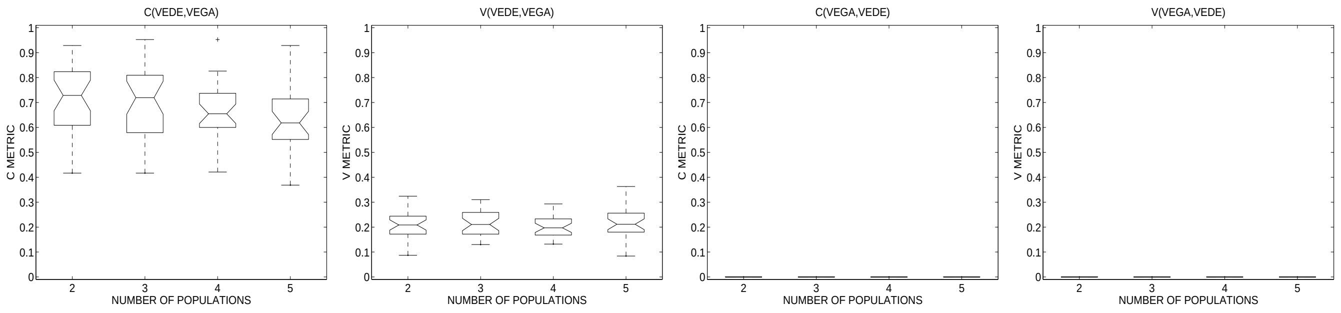

Fig. 4. Results of VEDE1 for the Test Problem 3.

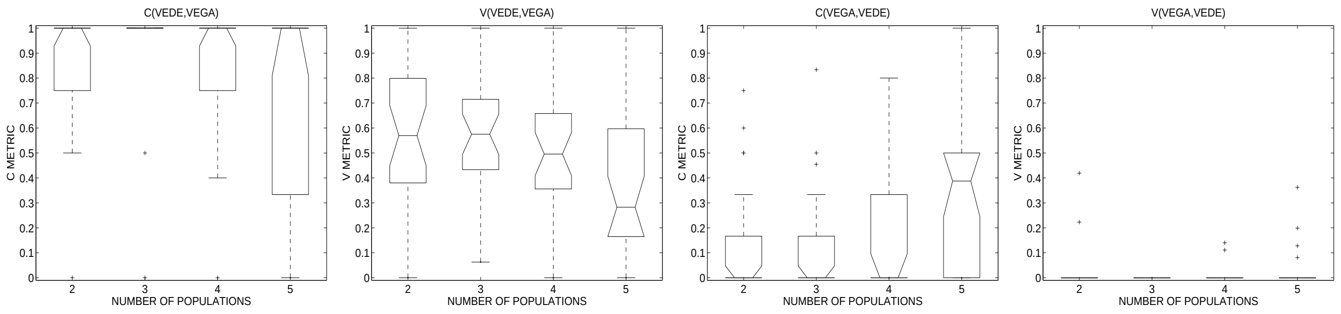

Fig. 5. Results of VEDE1 for the Test Problem 4.

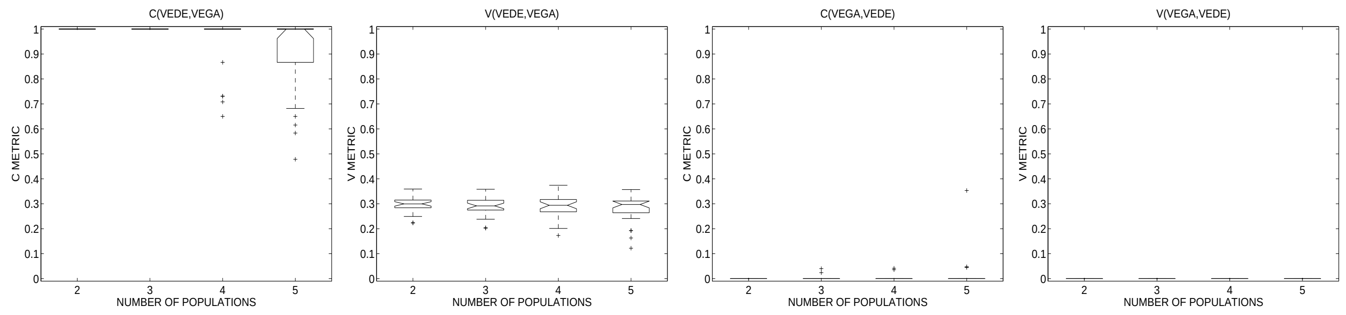

Fig. 6. Results of VEDE2 for the Test Problem 1.

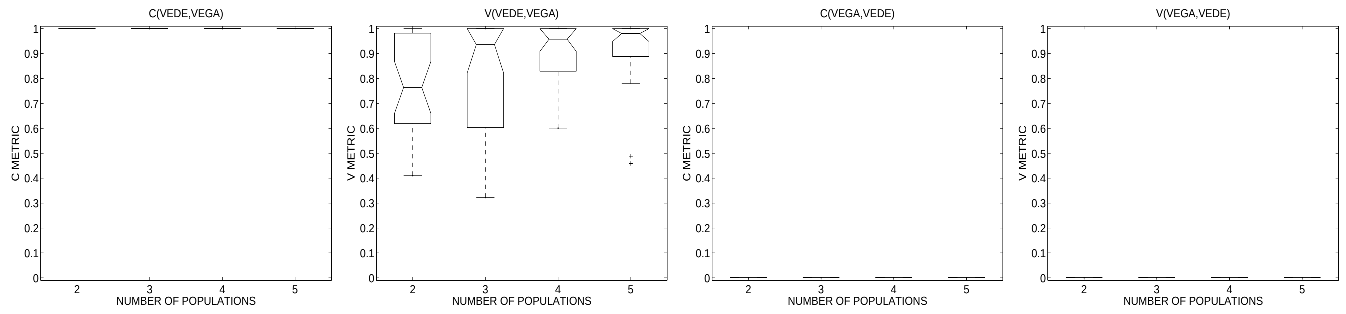

Fig. 7. Results of VEDE2 for the Test Problem 2.

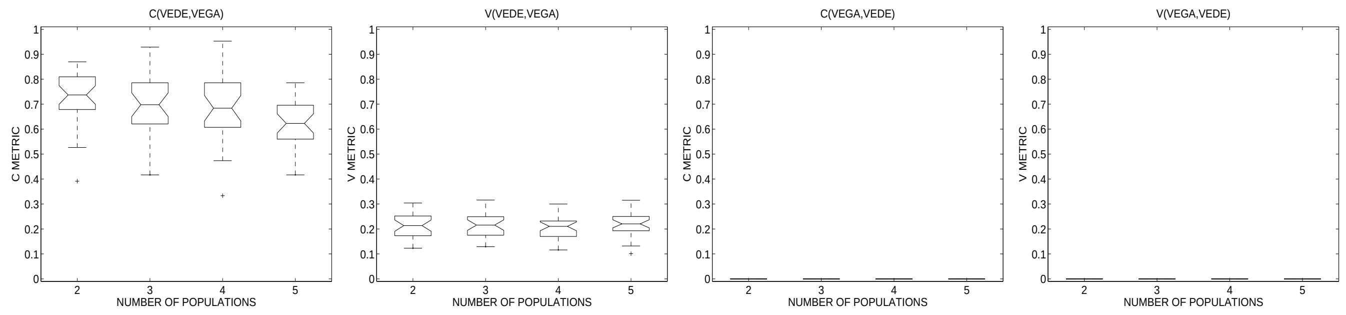

Fig. 8. Results of VEDE2 for the Test Problem 3.

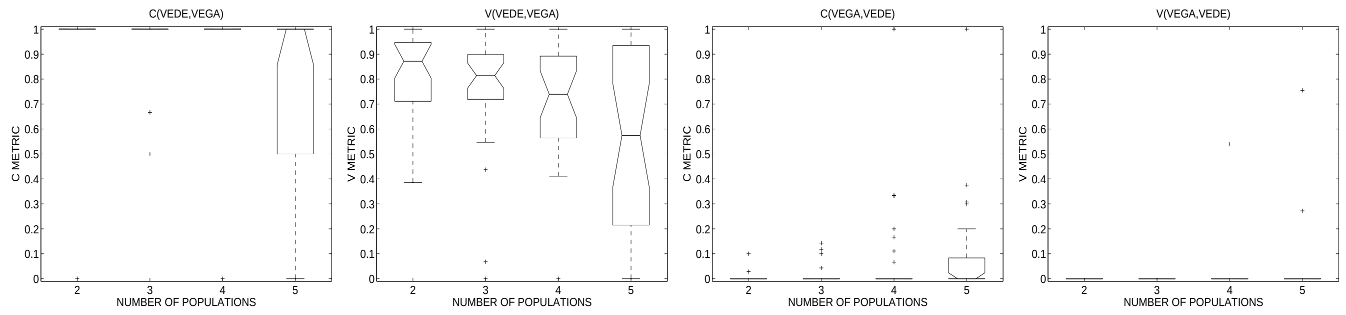

Fig. 9. Results of VEDE2 for the Test Problem 4.

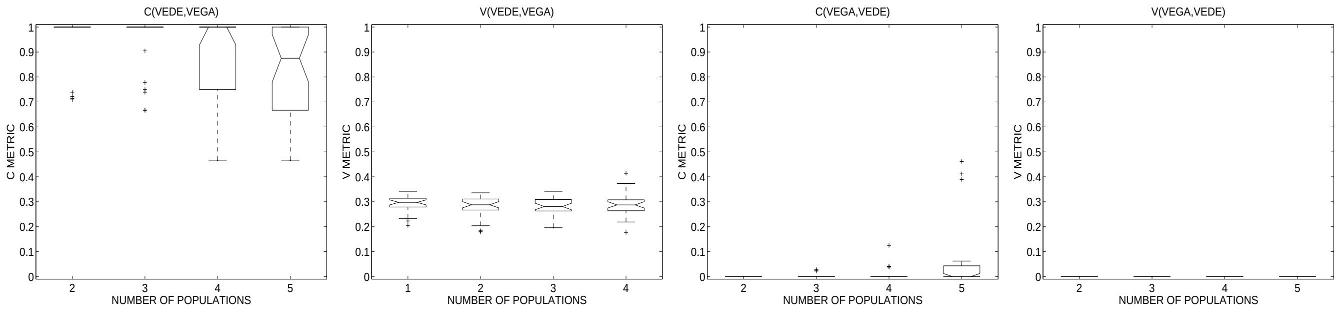

Fig. 10. Results of VEDE3 for the Test Problem 1.

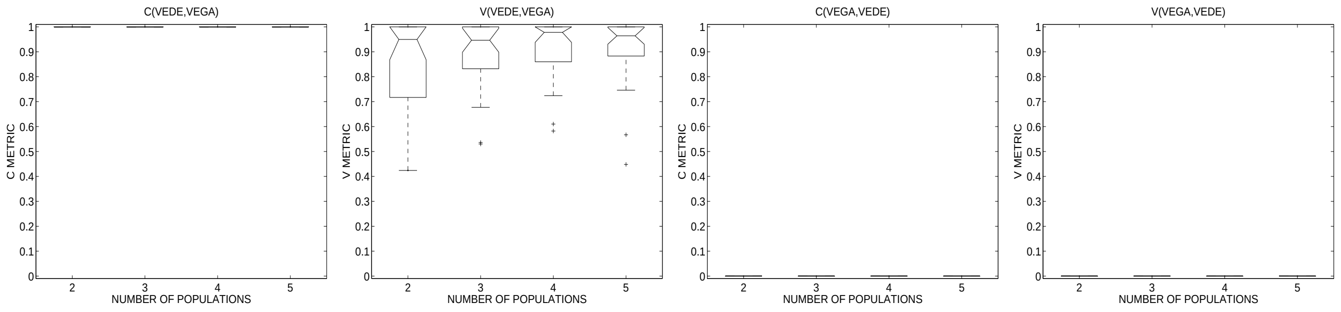

Fig. 11. Results of VEDE3 for the Test Problem 2.

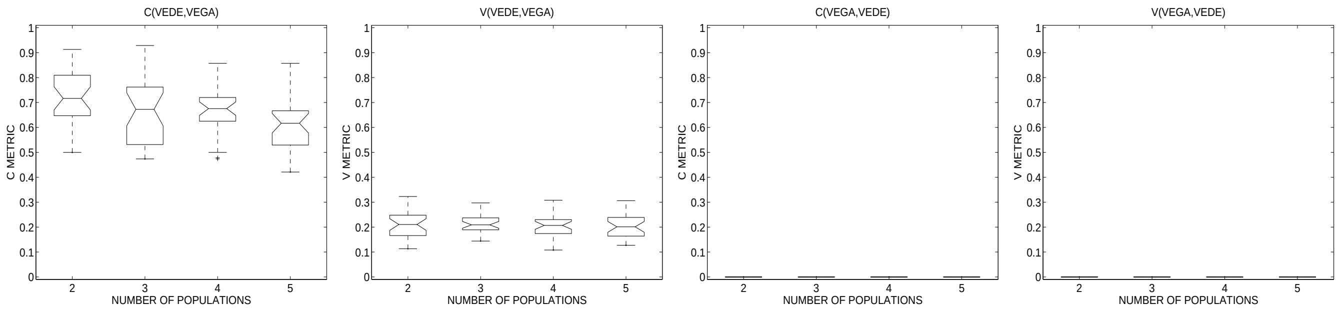

Fig. 12. Results of VEDE3 for the Test Problem 3.

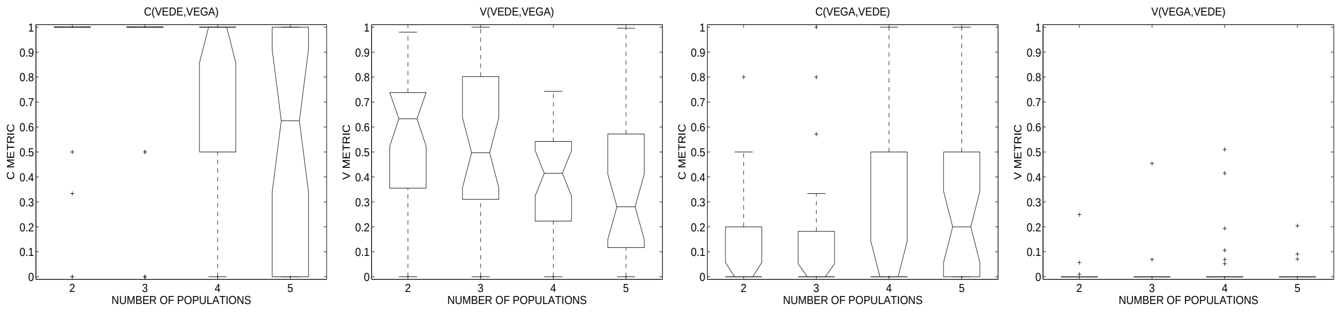

Fig. 13. Results of VEDE3 for the Test Problem 4.

Metric $\mathcal{C}(A,B)$ measures the fraction of members of the Pareto front B that are dominated by members of the Pareto front A, while $\mathcal{V}(A,B)$ is the fraction of the volume of the minimal hypercube containing both fronts, that is strictly dominated by members of A but is not dominated by members of B [21]. Following the analysis presented in [6], a total number of 100 individuals divided in several populations, as well as a maximum of 250 iterations per population per run, were used. We performed 30 experiments for each test problem, using the DE variants described in Eqs. (4), (5), and (6), respectively, because they suit better the migration scheme described in the previous section. The three variants are denoted as VEDE1, VEDE2, and VEDE3, respectively. All results are reported in the boxplots of Figs. 2–13. Each boxplot depicts the obtained values of the corresponding measure, in 30 experiments. The box has lines at the lower quartile, median, and upper quartile values. The lines extending from each end of the box (whiskers) show the extent of the rest of the data. The outliers, i.e. the values that lie beyond the ends of the whiskers, are denoted with crosses.

DE is quite sensitive to population size, especially when the number of individuals becomes small. This was verified in our preliminary experiments with VEDE. Dividing the 100 individuals into more than 5 populations (less than 20 individuals per population) resulted in substantial performance decline. Thus, our experiments were performed using 2 up to 5 populations. In Test Problems 1 to 3, standard values for the F and *CR* parameters, equal to 0.7 and 0.9, respectively, were used. These values have proved to be good default values for the DE algorithm in many applications [8]. In Test Problem 4, the aforementioned values proved inappropriate. Good parameter values proved to be $F = 0.5$ and $\mathit{CR} = 0.6$, for VEDE1 and VEDE3, while for VEDE2, F was set to 0.1 and *CR* was set to 0.7.

The speedup gained from the parallel implementation using up to 5 nodes is depicted in Fig. 14. As illustrated, there is a linearly increasing speedup rate using up to 4 nodes. Beyond 4 nodes, the speedup rate increases marginally. This effect can be attributed to the small number of individuals per population, which falls under 20.

In all cases, VEDE outperformed the VEGA with respect to the two metrics, $\mathcal{C}$ and $\mathcal{V}$. As seen in the first two boxplots of Figs. 2, 6, and 10, all three VEDE variants performed similarly in Test Problem 1. However, VEDE2 seems more robust, since the boxes are shorter and they lie closer to the upper bound, 1.0. The same can be noticed in the results for Test Problem 4. In the other two problems, the algorithms performed similarly, with VEDE3 having a slightly better performance, with respect to the $\mathcal{V}$ measure, in Test Problem 2. In all cases, increasing the number of populations resulted in a decrease of the overall performance of the algorithm with respect to the metric $\mathcal{C}$. An exception is Test Problem 2, where increasing the number of populations improved the $\mathcal{V}$ metric. The results support the claim that VEDE, like DE, is sensitive to population size.

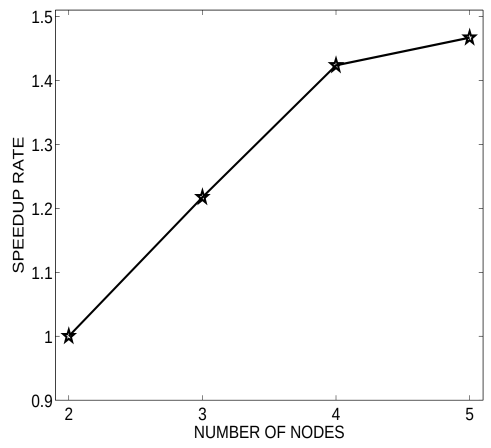

Fig. 14. Speedup gained using up to 5 nodes.

IV. Synopsis

This paper introduces a parallel, multi–population Differential Evolution algorithm, called Vector Evaluated Differential Evolution (VEDE), for multiobjective optimization. The algorithm uses a domination selection operator to enhance its performance by favoring non–dominated individuals in the populations. Preliminary experimental results on widely used test problems, as well as comparisons with the VEGA approach, are promising. The algorithm's sensitivity posed by the inherent sensitivity of the DE algorithm to its parameters (most notably population size) requires further investigation. This issue, along with alternatives to the ring topology, will be addressed in a future work.

Acknowledgment

The authors would like to thank the anonymous referees for their useful comments and suggestions. This work was partially supported by the "Karatheodoris" research grant, awarded by the Research Committee of the University of Patras, and the "Pythagoras" research grant, awarded by the Greek Ministry of Education and Religious Affairs, as well as, the European Union.

References

[1] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi–Objective Problems*. Kluwer, New York, 2002.

[2] K. Deb. Multi–objective genetic algorithms: Problem diffi culties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[3] J. D. Schaffer. *Multiple Objective Optimization With Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, USA, 1984.

[4] D. A. Van Veldhuizen, J. B. Zydallis, and G. B. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comp.*, 7(2):144–173, 2003.

[5] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zürich, Switzerland, 1999.

[6] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolution algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[7] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. Evolutionary operators in global optimization with dynamic search trajectories. *Numerical Algorithms*, 34(2–4):393–403, 2003.

[8] V. P. Plagianakos and M. N. Vrahatis. Parallel evolutionary training algorithms for "hardware–friendly" neural networks. *Natural Computing*, 1(2–3):307–322, 2002.

[9] R. Storn. System design by constraint adaptation and differential evolution. Technical Report Technical Report TR-96-039, ICSI, ICSI, 1996.

[10] J. Rajive and A. C. Sanderson. Minimal representation multisensor fusion using differential evolution. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997.

[11] B. V. Babu and K. K. N. Sastry. Estimation of heat transfer parameters in a trickle–bed reactor using differential evolution and orthogonal collocation. *Computers & Chemical Engineering*, 23(3):327–339, 1999.

[12] M. M. Fischer, K. Hlavackova-Schindler, and M. Reismann. A global search procedure for parameter estimation in neural spatial interaction modelling. *Papers in Regional Science*, 78(2):119–134, 1999.

[13] K. E. Parsopoulos, D. K. Tasoulis, and M. N. Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, Innsbruck, Austria, 2004.

[14] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedins of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, pages 603–607, Madrid, Spain, 2002. ACM Press.

[15] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2–3):235–306, 2002.

[16] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11:341–359, 1997.

[17] K. Price. Differential evolution: A fast and simple numerical optimizer. In *Proceedings NAFIPS'96*, pages 524–525, 1996.

[18] H. Abbass. Self–adaptive pareto differential evolution. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*, pages 831–836, Honolulu, Hawaii, 2002. IEEE Press.

[19] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, 1994.

[20] Y. Jin, M. Olhofer, and B. Sendhoff. Evolutionary dynamic weighted aggregation for multiobjective optimization: Why does it work and how? In *Proceedings GECCO 2001 Conference*, pages 1042–1049, San Francisco, CA, 2001.

[21] J. E. Fieldsend, R. M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimization. *IEEE Trans. Evol. Comp.*, 7(3):305–323, 2003.

[22] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multiobjective evolutionary algorithms with elitism. In *Proc. IEEE Congr. Evol. Comp.*, pages 46–53, Piscataway, NJ, 2000. IEEE Press.