MULTIOBJECTIVE OPTIMIZATION USING PARALLEL VECTOR EVALUATED PARTICLE SWARM OPTIMIZATION

K.E. Parsopoulos, D.K. Tasoulis, M.N. Vrahatis
Department of Mathematics, University of Patras Artificial Intelligence
Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece
email: {kostasp, dtas, vrahatis}@math.upatras.gr

ABSTRACT
This paper studies a parallel version of the Vector Evaluated Particle Swarm Optimization (VEPSO) method for multiobjective problems. Experiments on well known and widely used test problems are performed, aiming at investigating both the efficiency of VEPSO as well as the advantages of the parallel implementation. The obtained results are compared with the corresponding results of the Vector Evaluated Genetic Algorithm approach, yielding the superiority of VEPSO.

KEY WORDS
Particle Swarm Optimization, Multiobjective Optimization, PVM

1 Introduction

Multiobjective optimization (MO) problems consist of several objectives that need to be achieved simultaneously. Such problems arise in many applications, where two or more, sometimes competing and/or incommensurable objective functions have to be minimized concurrently. Due to the multicriteria nature of MO problems, the "optimality" of a solution has to be redefined, giving rise to the concept of *Pareto optimality*. In contrast to the single–objective optimization case, MO problems are characterized by trade–offs and, thus, a multitude of *Pareto optimal* solutions.

Traditional gradient–based optimization techniques can be used to detect Pareto optimal solutions. However, these techniques suffer from two critical drawbacks; (I) the objectives have to be aggregated in a single objective function, and, (II) only one solution can be detected per optimization run. The inherent difficulty to foreknow which aggregation of the objectives is appropriate in addition to the heavy computational cost of gradient–based techniques, necessitates the development of more efficient and rigorous methods. Evolutionary Algorithms (EAs) seem to be particularly suited to MO problems due to their ability to synchronously search for multiple Pareto optimal solutions and perform better global exploration of the search space [1, 2, 3]. Moreover, EAs are easily parallelized, thus, decreasing the computational load and the required execution time. The parallel computation of many solutions may also result in a better representation of the possible outcomes, enhancing the performance of the EA [4].

Particle Swarm Optimization (PSO) is a swarm intelligence method that roughly models the social behavior of swarms [5]. PSO is characterized by its simplicity and straightforward applicability, and it has proved to be efficient on a plethora of problems in science and engineering. Several studies have been recently performed with PSO on MO problems, and new variants of the method, which are more suitable for such problems, have been developed [6, 7, 8, 9].

Vector Evaluated Particle Swarm Optimization (VEPSO) is a multi–swarm variant of PSO, which is inspired by the Vector Evaluated Genetic Algorithm (VEGA) [3, 8]. In VEPSO, each swarm is evaluated using only one of the objective functions of the problem under consideration, and the information it possesses for this objective function is communicated to the other swarms through the exchange of their best experience.

In this paper, a study of the performance of VEPSO, using more than two swarms, as well as a parallel implementation of this approach, is presented. The efficiency of the algorithm, as well as the advantages of the parallel implementation are investigated and the results are reported and compared with the corresponding results of the VEGA approach. The rest of the paper is organized as follows; in Section 2 the basic MO concepts are described, and, in Section 3, the PSO and the VEPSO algorithms are briefly presented and, also, a description of the parallel implementation is provided. Experimental results are reported in Section 4, followed by conclusions in Section 5.

2 Basic Concepts of Multiobjective Optimization

Let $S \subset \mathbb{R}^n$ be an n–dimensional search space and

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

be k objective functions defined over S. Assuming,

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

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

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

that satisfies the constraints and optimizes (without loss of generality we consider only the minimization case) the function

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

The objective functions may be in conflict, thus, in most cases it is impossible to obtain the global minimum at the same point for all the objectives. The goal of MO is to provide a set of Pareto optimal solutions to the aforementioned problem.

Let $u = (u_1, \dots, u_k)$, and $v = (v_1, \dots, v_k)$, be two vectors. Then, u *dominates* v if and only if $u_i \leqslant v_i$, $i = 1, \dots, k$, and $u_i < v_i$ for at least one component. This property is known as *Pareto dominance* and it is used to define the Pareto optimal points. Thus, 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 it is denoted as $\mathcal{P}^*$. The set $\mathcal{PF}^* = \big\{\big(f_1(x), \dots, f_k(x)\big) \mid x \in \mathcal{P}^*\big\}$ is called *Pareto front*. A Pareto front $\mathcal{PF}^*$ is called *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 called *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. The last three cases present the greatest difficulty for most MO techniques.

3 Particle Swarm Optimization and Vector Evaluated Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a *swarm intelligence* algorithm, inspired by the social dynamics and emergent behavior that arises in socially organized colonies [5, 8, 10]. PSO is a population based algorithm, i.e. it exploits a population of individuals to probe promising regions of the search space. In this context, the population is called a *swarm* and the individuals (i.e. the search points) are called *particles*. Each particle moves with an adaptable velocity within the search space, and retains a memory of the best position it ever encountered. In the *global* variant of PSO, the best position ever attained by all individuals of the swarm is communicated to all the particles. In the *local* variant, each particle is assigned to a topological neighborhood consisting of a prespecified number of particles. In this case, the best position ever attained by the particles that comprise the neighborhood is communicated among them [5]. In this paper only the global variant is considered.

Assume an n–dimensional search space, $S \subset \mathbb{R}^n$, and a swarm consisting of N particles. The i–th particle is in effect an n–dimensional vector $X_i = (x_{i1}, x_{i2}, \dots, x_{in})^\top \in S$. The velocity of this particle is also an n–dimensional vector, $V_i = (v_{i1}, v_{i2}, \dots, v_{in})^\top \in S$. The best previous position encountered by the i–th particle is a point in S, denoted by $P_i = (p_{i1}, p_{i2}, \dots, p_{in})^\top \in S$. Assume g to be the index of the particle that attained the best previous position among all the particles in the swarm (global version), and t to be the iteration counter. Then, the swarm is manipulated by the equations [11, 12]:

$$\begin{aligned} V_i(t+1) &= \chi\Big[wV_i(t) + c_1 r_1\big(P_i(t) - X_i(t)\big) + \\ &\quad + c_2 r_2\big(P_g(t) - X_i(t)\big)\Big], \qquad (2) \\ X_i(t+1) &= X_i(t) + V_i(t+1), \qquad (3) \end{aligned}$$

where $i = 1, \dots, N$; c_1 and c_2 are two parameters called the *cognitive* and the *social* parameter, respectively, and they are used to bias the search of a particle toward its best experience and the best experience of the whole swarm, respectively; r_1, r_2, are random numbers uniformly distributed within $[0, 1]$. The parameters χ and w are called the *constriction factor* and the *inertia weight*, respectively, and they are used alternatively as mechanisms for the control of the velocity's magnitude, giving rise to the two different PSO versions. The selection of the aforementioned parameters has been widely discussed and studied in the relative literature [11, 13, 14].

The Vector Evaluated Particle Swarm Optimization (VEPSO) algorithm [8] has been inspired by the concept of the Vector Evaluated Genetic Algorithm (VEGA) [3]. In VEGA, fractions of the next generation or subpopulations are selected from the previous generation according to each of the objectives, separately. After shuffling all these sub–populations together, crossover and mutation are applied to generate the new population. These ideas have been adopted and modified to fit the PSO framework. Specifically, in VEPSO, two or more swarms are employed to probe the search space and information is exchanged among them [8]. Each swarm is exclusively evaluated with one of the objective functions, but, information coming from other swarm(s) is used to influence its motion in the search space. The best position attained by each particle (the particle's memory) separately as well as the best among these positions are the main guidance mechanisms of the swarm. Thus, exchanging this information among swarms can lead to Pareto optimal points.

Let the problem at hand consist of k objective functions, $f_l(x)$, $l = 1, \dots, k$, as defined in Eq. (1), and assume that M swarms, $\mathbb{S}_1, \mathbb{S}_2, \dots, \mathbb{S}_M$, of size N, are employed to address it. Each swarm is evaluated according to one of the objective functions. Let also $X_i^{[j]}, V_i^{[j]}$, and $P_i^{[j]}$, $i = 1, \dots, N$, $j = 1, \dots, M$, be the current position, the velocity and the best previous position of the i–th particle in the j–th swarm, respectively, at a given time. Assuming that $g_{[j]}$ denotes the index of the particle that attained the best previous position in the j–th swarm, then the VEPSO

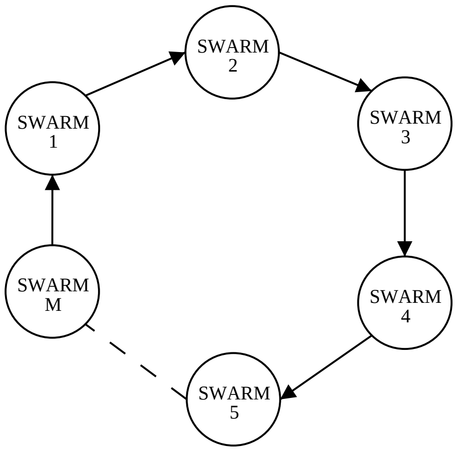

Figure 1. The ring migration scheme.

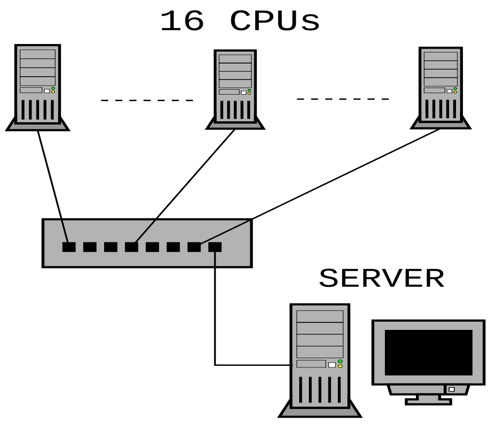

Figure 2. The PVM system used.

swarms are, in general, manipulated according to the equations [8]:

$$V_i^{[j]}(t+1) = \chi^{[j]} \Big[w^{[j]} V_i^{[j]}(t) + c_1^{[j]}\, r_1 \big(P_i^{[j]}(t) - X_i^{[j]}(t) \big) + + c_2^{[j]}\, r_2 \big(P_{g_{[s]}}^{[s]}(t) - X_i^{[j]}(t) \big) \Big], \tag{4}$$

$$X_i^{[j]}(t+1) = X_i^{[j]}(t) + V_i^{[j]}(t+1), \tag{5}$$

where $i = 1, \ldots, N$; $j = 1, \ldots, M$; $c_1^{[j]}$ and $c_2^{[j]}$ are the cognitive and social parameters of the j–th swarm; r_1, r_2, are random numbers uniformly distributed within $[0, 1]$; $\chi^{[j]}$ and $w^{[j]}$ are the constriction factor and the inertia weight of the j–th swarm, respectively; and s is an index taking values in $\{1, \ldots, j-1, j+1, \ldots, M\}$, i.e. the velocities of the j–th swarm are updated using the best previous position of another (the s–th) swarm. The case of two swarms with two objective functions has been presented and investigated in [8]. The procedure of exchanging information among swarms can be clearly viewed as a *migration* scheme in the parallel computation framework. The parameter s can be selected in a number of ways resulting in different VEPSO variants. For example, selecting s according to

$$s = \begin{cases} M, & \text{if} \quad j = 1, \\ j-1, & \text{if} \quad j = 2, \ldots, M, \end{cases} \tag{6}$$

corresponds to the “ring” migration topology [15], which is depicted in Fig 1. An alternative choice is to select s randomly. Further constraints may also be posed on the selection of s, e.g. allow the best particle of a swarm to migrate only to one swarm (this holds for the ring topology but not for the random selection).

This paper aims at investigating the efficiency of the VEPSO method as well as possible benefits obtained by its parallel implementation. Specifically, the main goals are to investigate VEPSO’s performance using different numbers of swarms on a single machine, as well as the time acceleration obtained if more than one machines are used.

Characteristic	Description
Number of CPUs	2 to 10
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

Table 1. The characteristics of the system used for the parallel experiments.

The VEPSO approach can be straightforwardly parallelized by distributing the swarms in many machines and allowing migration from node to node. For this purpose the Parallel Virtual Machine (PVM) has been used [16]. The key characteristics of the system used in the parallel implementation of VEPSO are reported in Table 1 and its topology is depicted in Fig. 2. In addition to the reported hardware, a Pentium III machine with 512-MB of memory, running under Red Hat Linux 8.0, has been used as a server. The single–machine experiments have also been performed on one of the aforementioned systems. Regarding VEPSO’s parallelization parameters, a ring migration topology has been selected, with migration taking place at each iteration (synchronized swarms’ move), employing from 2 up to 10 swarms. For the maintenance of the Pareto optimal set, the archiving technique described in [17] has been used.

The obtained results are evaluated using two established measures, the $\mathcal{C}$ metric [18, 19], and the $\mathcal{V}$ measure [18, 20]. The 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 [18].

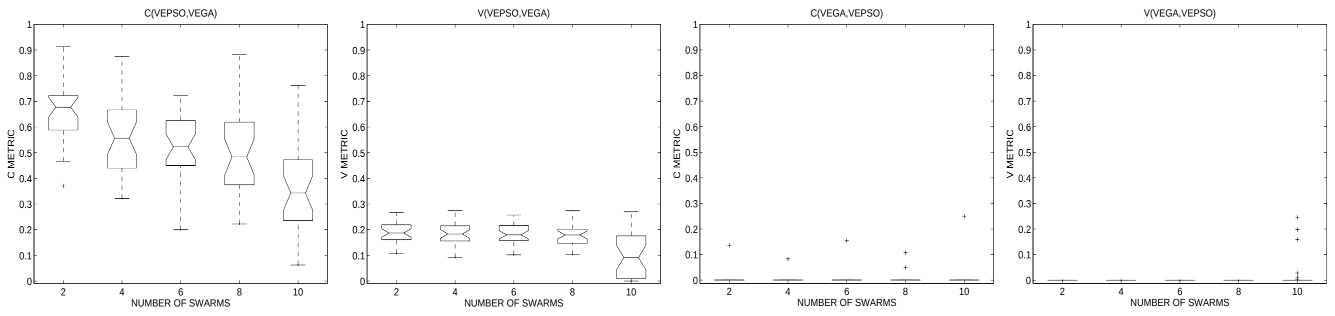

Figure 3. Results for the Test Problem 1.

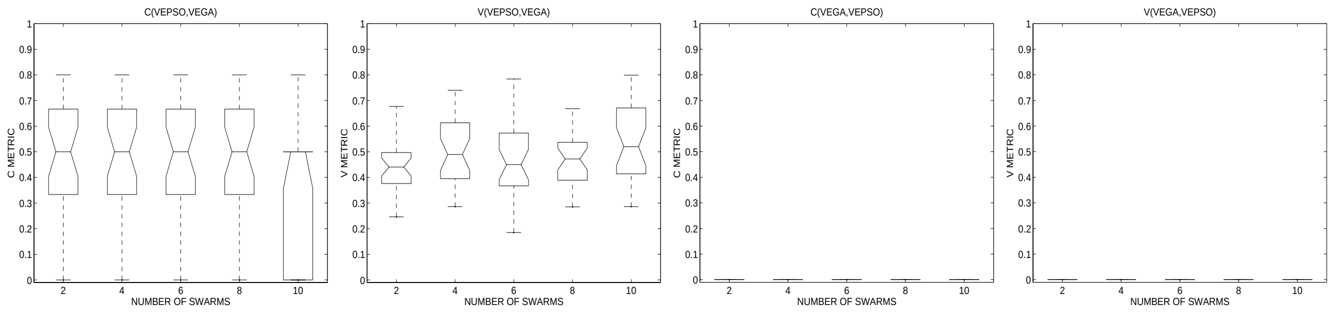

Figure 4. Results for the Test Problem 2.

4 Experimental Results

The following well–known benchmark problems have been used to illustrate the performance of VEPSO:

TEST PROBLEM 1. [19] This problem has a convex Pareto front:

$$f_1(x_1) = x_1, \quad (7)$$

$$g(x_2,\ldots,x_n) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i, \quad (8)$$

$$h(f_1,g) = 1-\sqrt{\frac{f_1}{g}}, \quad (9)$$

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

TEST PROBLEM 2. [19] This is the nonconvex counterpart to the Test Problem 1:

$$f_1(x_1) = x_1, \quad (10)$$

$$g(x_2,\ldots,x_n) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i, \quad (11)$$

$$h(f_1,g) = 1-\left(\frac{f_1}{g}\right)^2, \quad (12)$$

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

TEST PROBLEM 3. [19] This Pareto front consists of several convex parts:

$$f_1(x_1) = x_1, \quad (13)$$

$$g(x_2,\ldots,x_n) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i/(n-1), \quad (14)$$

$$h(f_1,g) = 1-\sqrt{\frac{f_1}{g}}-\frac{f_1}{g}\sin(10\pi f_1), \quad (15)$$

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

TEST PROBLEM 4. [19] This test problem has 21^9 local Pareto fronts:

$$f_1(x_1) = x_1, \quad (16)$$

$$g(x_2,\ldots,x_n) = 1+10(n-1)+ + \sum_{i=2}^{n}\left(x_i^2-10\cos(4\pi x_i)\right), \quad (17)$$

$$h(f_1,g) = 1-\sqrt{\frac{f_1}{g}}, \quad (18)$$

with $n = 10$, $x_1 \in [0, 1]$, and $x_2, \ldots, x_n \in [-5, 5]$.

In all experiments, the global variant of the constriction factor PSO has been used. The PSO parameters have been the same for each swarm and for all problems, equal to: $\chi = 0.729$, $c_1 = c_2 = 2.05$ [11].

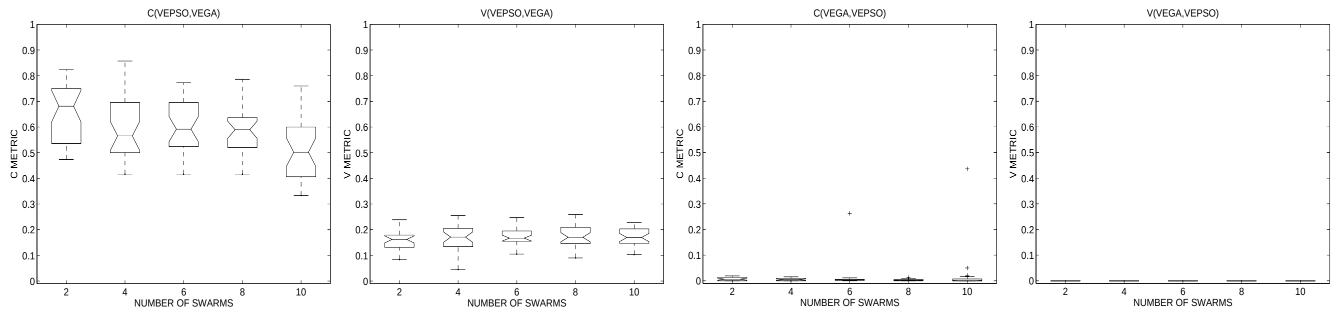

Figure 5. Results for the Test Problem 3.

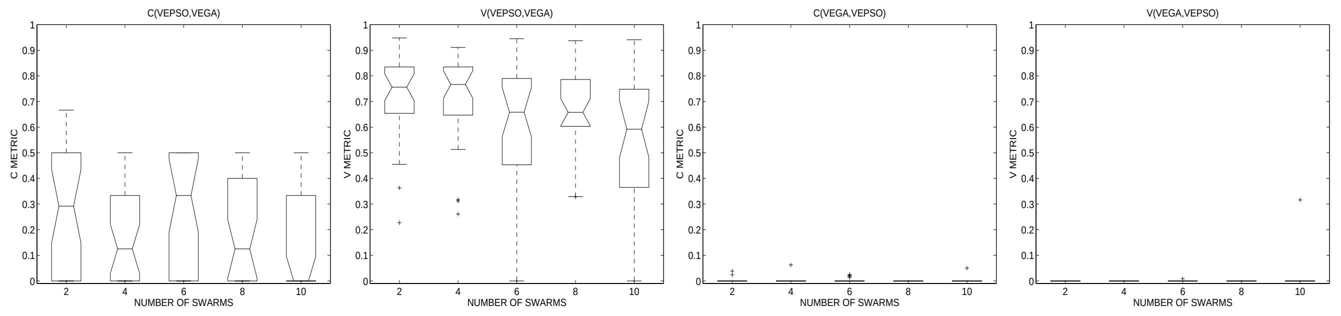

Figure 6. Results for the Test Problem 4.

The results obtained through VEPSO, using 2, 4, 6, 8, and 10 swarms, have been compared with the results obtained through VEGA that are freely available at the web page *http://www.tik.ee.ethz.ch/∼zitzler/testdata.html*. For this purpose, 100 particles have been used, in total, for each experiment, divided in 2 up to 10 sub–swarms. For each case, 30 independent experiments have been performed. The maximum number of iterations of the VEPSO algorithm for each experiment has been set equal to 250. All results are statistically displayed with boxplots in Figs. 3–6. The boxplots are based on the two metrics, $\mathcal{C}$ and $\mathcal{V}$, with respect to the number of swarms that have been used. Each boxplot represents the distribution of the $\mathcal{C}$ or $\mathcal{V}$ values for the ordered pair (VEPSO,VEGA) and vice versa (notice that both the $\mathcal{C}$ and the $\mathcal{V}$ metric are neither symmetrical in their arguments nor satisfy the triangle inequality, thus, in general, $\mathcal{C}(A, B) \neq \mathcal{C}(B, A)$). Each box of the boxplot has lines at the lower quartile, median, and upper quartile values. The lines that extend from each end of the box are the whiskers, and they show the extent of the rest of the data. The outliers lie beyond the ends of the whiskers and they are denoted with crosses. The notches represent a robust estimate of the uncertainty about the medians for box to box comparison.

The obtained results support the claim that the VEPSO algorithm outperforms the VEGA algorithm in all cases. The $\mathcal{C}(\mathrm{VEPSO}, \mathrm{VEGA})$ and the $\mathcal{V}(\mathrm{VEPSO}, \mathrm{VEGA})$ assume relatively high values in all test problems while $\mathcal{C}(\mathrm{VEGA}, \mathrm{VEPSO})$ and $\mathcal{V}(\mathrm{VEGA}, \mathrm{VEPSO})$ are in almost all cases equal to zero. Moreover, it seems that using more than 2 swarms in some cases improves the performance of VEPSO. However, too many swarms do not offer significant performance advantages. Perhaps this happens due to the small size of each swarm, which, for the case of 10 swarms, is equal to 10 particles per swarm. Increasing the swarm size may further enhance the algorithm's performance.

The aforementioned experiments were performed both serially and in parallel. The parallel implementation resulted in an improvement of the performance in terms of the required execution time, which is depicted in Fig. 7. As can be seen, increasing the number of swarms from 2 up to 6 swarms, there is a significant gain in time. However, using more than 6 swarms results in increased time due to the heavy network overhead.

5 Conclusions

The VEPSO approach, which is based on the PSO method, for MO problems has been applied on four well known test problems. Both single–node and parallel implementations of the algorithm have been developed and applied with very promising results. Two widely used metrics have been used for the evaluation of the results and for comparisons with the corresponding results of the VEGA approach. VEPSO

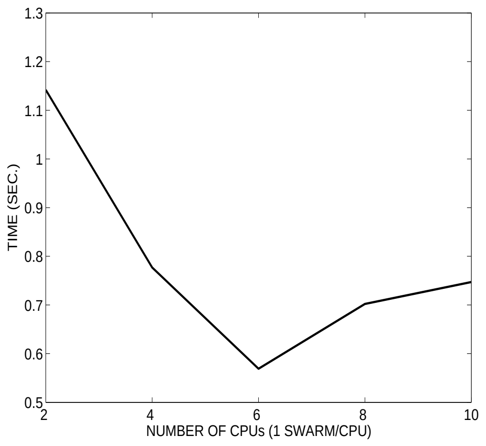

Figure 7. Time for the parallel implementation.

outperformed the VEGA approach in all cases. Future research will include a thorough investigation of the developed approaches as well as a comparison with other parallel EAs for MO problems.

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 difficulties 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] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[6] C. A. Coello Coello and M. S. Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Proc. 2002 IEEE Congress on Evolutionary Computation*, pages 1051–1056, Hawaii, HI, USA, 2002.

[7] X. Hu. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, Honolulu, HI, USA, 2002.

[8] 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.

[9] T. Ray and K. M. Liew. A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, 34(2):141–153, 2002.

[10] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[11] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.

[12] R. C. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming*, volume VII, pages 611–616. Springer, 1998.

[13] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming*, volume VII, pages 591–600. Springer, 1998.

[14] I. C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85:317–325, 2003.

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

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.