2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Dynamic Multi-Objective Optimization using Charged Vector Evaluated Particle Swarm Optimization

Kyle Robert Harrison
Department of Computer Science
Brock University
St. Catharines, ON, Canada
kh08uh@brocku.ca

Beatrice M. Ombuki-Berman
Department of Computer Science
Brock University
St. Catharines, ON, Canada
bombuki@brocku.ca

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa
engel@cs.up.ac.za

***Abstract*—The vector evaluated particle swarm optimization (VEPSO) algorithm is a multi-swarm variation of the traditional particle swarm optimization (PSO) used to solve static multi-objective optimization problems (MOOPs). Recently, the dynamic VEPSO (DVEPSO) algorithm was proposed as an extension to VEPSO enabling the algorithm to handle dynamic MOOPs (DMOOPs). While DVEPSO has been successful at handling DMOOPs, the change detection mechanism relied on observing changes in objective space. An alternative strategy is proposed by using charged PSO (CPSO) sub-swarms with decision space change detection to address the outdated memory issue observed in vanilla PSO. This dynamic PSO variant allows for (implicit) decision space tracking not seen in DVEPSO while implicitly handling the diversity issue seen in dynamic environments. The proposed charged VEPSO is compared to DVEPSO on a wide variety of dynamic environment types. Results indicated that, in general, the proposed charged VEPSO outperformed the existing DVEPSO. Further, charged VEPSO exhibited better front-tracking abilities, while DVEPSO was superior with regards to locating the Pareto front.**

I. Introduction

Many real-world problems exhibit some dynamic behavior such that the desired solution changes as time progresses. It is also common for there to be multiple goals which are to be concurrently optimized. These goals are often in direct competition; an improvement in one goal causes a detrimental effect in another. For such problems, a set of optimal trade-offs is desired. However, the dynamic behavior may cause the optimal set of trade-offs to change or may cause the decision variables leading to optimal solutions to be altered. Problems with such characteristics are known as dynamic multi-objective optimization problems (DMOOPs) [1].

Multi-objective optimization problems (MOOPs) are intuitively harder than single objective problems as they have a number of conflicting sub-objectives [1]. Further, MOOPs may have an infinite number of optimal solutions representing trade-offs among different sub-objectives. When a sub-objective cannot be improved further without worsening another sub-objective, the solution is referred to as non-dominated or Pareto optimal. The set of Pareto optimal solutions is called the Pareto optimal front (POF), while the corresponding set of decision variables is referred to as the Pareto optimal set (POS). Thus, the goal of multi-objective optimization is to find a well-distributed set of such Pareto optimal solutions [2].

An additional level of difficulty is added to MOOPs, forming DMOOPs. This difficulty comes in the form of environmental changes to the decision space, objective space, or both. A change in decision space occurs when the set of decision variables which correspond to the optimal set of solutions changes over time. Similarly, a change in objective space occurs when the fitness values of the optimal solutions vary over time. Farina *et al.* [1] provided a classification of DMOOP environments as follows:

- **Type I** – the Pareto optimal front does not change, but the Pareto optimal set does.
- **Type II** – both the Pareto optimal front and set change.
- **Type III** – the Pareto optimal front changes, while the Pareto optimal set remains unchanged.
- **Type IV** – both the Pareto optimal front and set remain unchanged but the problem still changes over time.

The vector evaluated particle swarm optimization (VEPSO) algorithm was introduced by Parsopoulos and Vrahatis [3] as a multi-swarm variant of the original particle swarm optimization (PSO) [4] algorithms, used to optimize MOOPs. VEPSO works by assigning each sub-swarm a dedicated sub-objective as its only optimization task. However, in order to optimize the MOOP as a whole, knowledge transfer strategies (KTSs) [3], [5], [6] are used to propagate information between the sub-swarms. Recently, DVEPSO [7] was proposed as a dynamic variant of VEPSO used to handle the optimization of DMOOPs. DVEPSO was shown to be successful at solving DMOOPs [7], [8] but only tracked changes in objective space. Proposed is a VEPSO variant that uses charged PSO [9] sub-swarms, which use change detection at the decision space level to address the issue of outdated memory [10]. Decision space tracking allows each sub-swarm to manage their own change detection and response mechanisms.

The remainder of the paper is structured as follows. Section II introduces the charged PSO and DVEPSO algorithms, while the charged VEPSO algorithm is given in Section III. The experimental details are outlined in Section IV with a discussion of the results in Section V, followed by concluding remarks in Section VI.

II. Background

Section II-A provides definitions required for the remainder of the paper, while Sections II-B, II-C, and II-D present PSO, CPSO, and DVEPSO, respectively.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

A. Definitions

Let $\vec{f}$ and $\vec{f}^*$ be two objective vectors with n_o sub-objectives, respectively.

Definition 1 (Dominance). A dominance relation, $\prec$, is given by $\vec{f}^* \prec \vec{f} := \forall k : f_k^* \leq f_k \wedge \exists k : f_k^* < f_k$, where $k \in \{1, 2, ..., n_o\}$. An objective vector which is not dominated by any other objective vector is said to be non-dominated.

Definition 2 (Pareto Optimal Front). The Pareto optimal front, $POF(t)^*$, is the solution set of a DMOOP, at time t, and is defined as the set of all non-dominated objective vectors. This definition can be stated more formally as $POF(t)^* = \{\vec{f}^* \in S | \nexists \vec{f} \in S : \vec{f} \prec \vec{f}^*\}$, where S denotes the objective space corresponding to feasible solutions.

Definition 3 (Pareto Optimal Set). The Pareto optimal set, $POS(t)^*$, is the decision-space analogue to the Pareto optimal front. That is, the Pareto optimal set contains decision vectors which correspond to solutions in the Pareto optimal front, given by $POS(t)^* = \{\vec{x}^* \in D | f(\vec{x}^*) \in POF(t)^*\}$ where D denotes the decision space of the DMOOP.

B. Particle Swarm Optimization

PSO was introduced by Kennedy and Eberhart [4] as a model of the social dynamics of a flock of birds. Further examination lead to the discovery of a complex behavior exhibited by the agents, i.e., convergence to a single point. As such, PSO was developed as a population based, stochastic optimization technique initially meant for real-valued, continuous space optimization problems. Particles are "flown" through the search space using an iterative process of calculating and applying a velocity vector. The velocity update equation used by vanilla PSO is given as

$$\begin{aligned}\vec{v}_i(t+1) = \omega\vec{v}_i(t) + c_1\vec{r}_1(t)(\vec{p}_{best} - \vec{x}_i(t)) \\ + c_2\vec{r}_2(t)(\vec{g}_{best} - \vec{x}_i(t)) \ .\end{aligned} \tag{1}$$

In Equation (1), $\vec{v}_i$ is the velocity of particle i, $\vec{x}_i$ is the position of the particle, $\vec{p}_{best}$ is the personal best position of the particle, and $\vec{g}_{best}$ is the swarm's best-found position, assuming a star neighborhood topology. The constants $\omega \in [0, 1], c_1 \geq 0,$ and $c_2 \geq 0$ are the *inertia*, *cognitive*, and *social* weights, respectively, which are user-supplied parameters. Inertia applies a portion of the previous velocity to the current velocity in an attempt to keep the particles direction from changing. The cognitive and social components determine the influence taken from the personal best and global best positions, respectively. Finally, $\vec{r}_1(t)$ and $\vec{r}_2(t)$ are vectors sampled from a uniform distribution in the range [0,1], giving the PSO algorithm its stochastic element.

PSO is known to have two major drawbacks in dynamic environments, namely outdated memory and diversity loss [10]. Outdated memory occurs when an environmental change invalidates a particle's personal best and/or fitness values. Outdated memory may lead to attractors which are no longer in promising areas of the search space. The other major issue for PSO in dynamic environments, i.e., diversity loss, occurs when the swarm has exhibited convergence and little to no variation exists in the attractors. The small velocities later in a run coupled with a drastic shift in the optimum can cause particles to stagnate and oscillate along a linear path, a phenomenon known as linear collapse. Variants of PSO have been proposed to alleviate these two major drawbacks [11], [12], [9].

C. Charged Particle Swarm Optimization

Charged PSO (CPSO), introduced by Blackwell and Bentley [9], is based on the orbit model of an atom where electrons orbit a nucleus. In CPSO, some proportion of the population is given a charge which repels other charged particles. Particles given a charge are referred to as charged particles, while the remainder are referred to as neutral particles. Particle velocities are updated using the vanilla update equation (Equation (1)) with an additional term (acceleration) based on their proximity to other particles. The CPSO velocity update equation is given as

$$\begin{aligned}\vec{v}_i(t+1) = \omega\vec{v}_i(t) + c_1\vec{r}_1(t)(\vec{p}_{best} - \vec{x}_i(t)) \\ + c_2\vec{r}_2(t)(\vec{g}_{best} - \vec{x}_i(t)) + \sum_{\forall j \neq i} a_{ij} \ .\end{aligned} \tag{2}$$

The acceleration between particles i and j is given by

$$a_{ij} = \begin{cases} \frac{Q_i Q_j}{||\vec{x}_i - \vec{x}_j||^3}(\vec{x}_i - \vec{x}_j) & \text{if } p_{core} < ||\vec{x}_i - \vec{x}_j|| \leq p \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where Q_i refers to the charge of particle i, while p and p_{core} control the acceleration term's radius of effect.

Neutral particles behave as they would in vanilla PSO, causing convergence of the neutral particles around the global best to be exhibited by CPSO. However, the acceleration term addresses the issue of diversity loss during convergence by the repulsion of charged particles, which causes enhanced exploration. Thus, CPSO is designed to maintain a higher level of diversity throughout the run, thereby eliminating the issue of linear collapse noted with vanilla PSO. Although the diversity issue is implicitly handled by CPSO, an external strategy is needed to address outdated memory.

D. Dynamic Vector Evaluated Particle Swarm Optimization

VEPSO [3] was introduced as a multi-swarm variant of PSO used to handle MOOPs. Due to the multi-swarm nature of VEPSO, a mechanism is needed to transfer information between sub-swarms allowing the algorithm to optimize the MOOP as a while. VEPSO employs a mechanism known as a knowledge transfer strategy (KTS) to facilitate the transfer of knowledge between sub-swarms. Currently, there are only two such KTSs in use, although various others have been recently proposed [6]. The original ring KTS [3] selects the global best position of the neighboring sub-swarm, using a ring topology, as the global guide, while the random global best KTS selects the global guide of a randomly selected sub-swarm (including itself). This study used the random global best KTS, which has been shown to outperform the original ring KTS in both static and dynamic environments [6], [13].

The DVEPSO algorithm was introduced by Greeff and Engelbrecht [7] as a dynamic variant of VEPSO [3]. The main

contribution to the original VEPSO algorithm was the addition of sentry particles in the archive, which monitored the objective space to determine if an environmental change occurred. If such a change was detected, the archive was cleared and the sub-swarm which observed the change had a portion of its particles reinitialized while all other particles were reevaluated. These two response mechanisms handled both the diversity and outdated memory issues. DVEPSO was observed to be suited for both locating and tracking the Pareto front in various dynamic environments [7], [8]. The DVEPSO algorithm is depicted in Algorithm 1.

Algorithm 1 Dynamic VEPSO Algorithm

```
function DVEPSO
    initialize PSO for each sub-objective
    while stopping criteria not met do
        if change occured then
            clear archive
            reinitialize some particles
            reevaluate all particles
        end if
        perform VEPSO iteration
        for all candidate solutions do
            if non-dominated then
                if archive full then
                    prune archive
                end if
                add new solution to archive
            end if
        end for
    end while
end function
```

III. Charged Vector Evaluated Particle Swarm Optimization

While the DVEPSO algorithm has been shown to be effective at solving DMOOPs [7], [8], it suffered from two drawbacks: change detection mechanisms resided only in objective space while diversity loss was addressed via reinitialization of particles, which cause drastic changes. The exclusive tracking in objective space was alleviated by replacing each vanilla PSO sub-swarm in the DVEPSO algorithm with CPSO, an inherently dynamic PSO variant. CPSO sub-swarms, equipped with sentry particles [12], allowed for implicit change detection in decision space. Futher, CPSO sub-swarms eliminated the need for diversity injection via reinitialization. The proposed charged VEPSO algorithm replaced each of the vanilla PSO sub-swarms in VEPSO with CPSO sub-swarms. Algorithm 2 shows the general approach to the proposed charged VEPSO.

The charged VEPSO algorithm uses sentry points in both the archive as well as each sub-swarm to track changes in objective and decision space, respectively. Each sub-swarm contains sentry particles, allowing changes in each sub-objective to be tracked separately. When a change is detected in decision space, particles are reevaluated to address outdated memory. Similar to DVEPSO, when the archive sentries report changes in objective space, archive solutions are cleared to prevent invalid solutions from being maintained.

Different settings for the charge and core radius were tested for the CPSO sub-swarms of the charged VEPSO algorithm. The charge was selected from $\{8, 16, 32\}$, while the core radius (p_{core}) was selected from $\{0.5, 1, 2\}$.

Algorithm 2 Charged VEPSO Algorithm

```
function CHARGED VEPSO
    initialize CPSO for each sub-objective
    while stopping criteria not met do
        if change occured then
            clear archive
        end if
        perform VEPSO iteration using CPSO sub-swarms
            ▷ CPSO uses sentries and particle reevaluation
        for all candidate solutions do
            if non-dominated then
                if archive full then
                    prune archive
                end if
                add new solution to archive
            end if
        end for
    end while
end function
```

IV. Experimental Setup

This section describes the experimental parameters selected, the benchmark functions examined, the performance measures employed, and finally provides the statistical analysis methodology employed in this paper.

A. Parameterization

All experiments were executed using the Computational Intelligence library (CIlib) [14]. Each algorithm was run for a total of 1000 iterations and kept an archive with maximum size of 100. When a solution was needed to be removed from the archive, the solution with the smallest nearest neighbor distance was chosen for removal. PSO parameters were chosen as values shown to lead to convergent behavior [15], namely ω = 0.729844 and c_1 = c_2 = 1.496180. A clamping boundary condition was enforced, preventing particles from exiting the feasible region. Personal best positions were updated if the new solution dominated the old solution. However, if both solutions were non-dominated with respect to each other, one was randomly selected as the personal best. Each sub-swarm had 20 particles, with CPSO sub-swarms having 10 (50%) charged particles.

A major difference between DVEPSO and charged VEPSO was the change detection mechanisms. DVEPSO used randomly selected archive solutions which reported a change if any component of their fitness vector differed by more than a threshold. Thus the monitoring was done in objective space. When a change occurred, the archive solutions were cleared and 30% of particles were reinitialized. Furthermore, particles were reevaluated when a change was detected. Charged VEPSO also made use of archive sentries, which triggered the archive to be cleared when a change was detected. However, the CPSO sub-swarms also had decision-level change detection

which triggered particle reevaluation. Further, CPSO subswarms did not need reinitialization as diversity is implicitly maintained.

B. Benchmark Functions

Five benchmark functions of varying environment types were used, namely dMOP1 [16], dMOP2 [16], dMOP3 [16], FDA1 [1], and FDA3$_{Camara}$ [17]. dMOP1 is a Type III environment, dMOP2 and FDA3$_{Camara}$ are Type II environments, while FDA1 and dMOP3 are Type I environments. The true POFs of each examined benchmark function are shown in Figure 1. Note that Type IV environments, which arise only in special cases [1], were not considered. Besides, in Type IV environments, both the POF and POS remain static over time.

Several temporal and spatial severity settings were used to determine the effects of varying the degree of dynamism on performance. Temporal severity (τ_t) refers to the number of iterations between environmental changes while spatial severity (n_t) refers to the magnitude of environmental changes. Although previous classification systems exist [18], [19], Duhain's [20] behavioral classes, depicted in Figure 2, are used in this work. The settings for temporal and spatial severities used in this study are presented in Table I.

TABLE I: Temporal and Spatial Severity Settings

n_t	τ_t	**Environment Type**
1	50	Infrequent, small changes (quasi-static)
1	5	Frequent, small changes (progressive)
20	50	Infrequent, large changes (abrupt)
20	5	Frequent, large changes (chaotic)

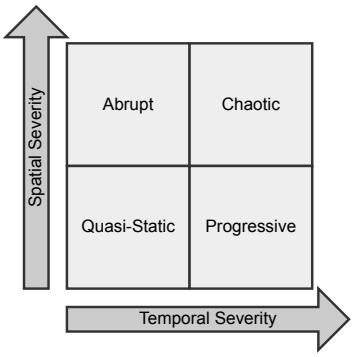


Fig. 2: Duhain's Dynamic Behavioral Classes [20]

C. Performance Measures

The goal of a DMOOP optimizer is two-fold: the optimizer must be able to locate the Pareto optimal front, while also being able to adapt to environmental changes and track the Pareto front over time. As such, the performance measures described below are calculated at each iteration before a change occurs.

Accuracy: Accuracy (*acc*) [21] measures the closeness of an approximation front to the true front. The accuracy measure is calculated as

$$acc(t) = |HV(POF(t)^*) - HV(POF(t))| \quad (4)$$

where $HV(POF^*(t))$ denotes the hypervolume [22] of the true Pareto front at time t. To approximate the true front, 1000 equidistant points were used. For hypervolume calculations, the reference vector was set as the worst observed objective values. A low *acc* value indicates good performance.

Stability: Stability (*stab*) [21] measures the effect of environmental changes on the accuracy. This measure is given as

$$stab(t) = \max\{0, acc(t-1) - acc(t)\} \quad (5)$$

where a low *stab* value indicates good performance.

Number of Solutions: Number of solutions (*NS*) refers to the number of non-dominated solutions in the archive.

D. Statistical Analysis

Each experiment consisted of 30 independent runs. The normalized wins and losses approach by Helbig and Engelbrecht [23] was used to analyze the performance of the optimizers. All statistical tests were performed at the 95% confidence level.

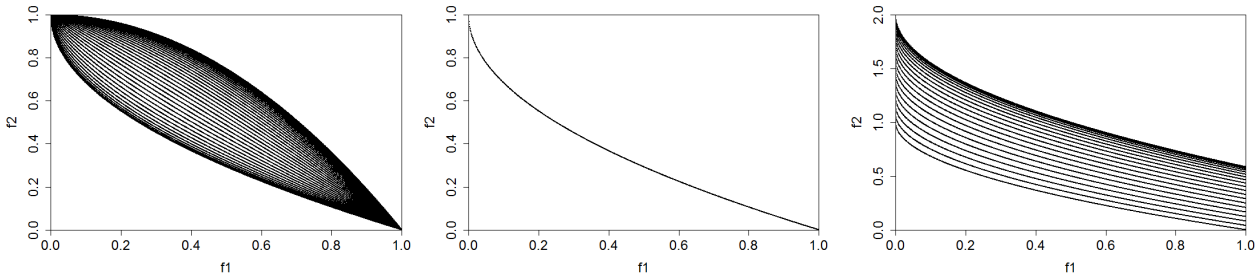
V. Experimental Results and Discussion

This section discusses the results of the examined optimizers. The overall, aggregated performance is presented in Section V-A, while Sections V-B and V-C examine the results with regards to various dynamic environment and DMOOP types, respectively. For the tables presented in this section, PM refers to the performance measure, C refers to a standard charged VEPSO optimizer, with charge of 16 and a core radius of 1, D refers to the DVEPSO optimizer, while C-x-y refers to various configurations of the charged VEPSO optimizer, with a charge of x and a core radius of y.

A. Overall Performance

The results aggregated over all functions, performance measures, and environment types, presented in Table II demonstrate that the standard charged VEPSO (i.e., the charged VEPSO with charge of 16 and core radius of 1) had the smallest difference between the number of wins and losses, and attained 113.49 more wins than losses. Relatively close in performance to the standard charged VEPSO was the DVEPSO optimizer which scored a difference of 101.32 more wins than losses, 10.72% less than that of the standard charged VEPSO. The worst overall optimizer was the charged VEPSO using a charge of 16 and a core radius of 0.5, with 125.70 more losses than wins.

Examining the performance of charged VEPSO using various parameters, as shown in Table II, a trend was quite apparent with respect to core radius and performance. The three variants of charged VEPSO (i.e., including the standard one) which obtained the best performance had a core radius of 1, while the three worst performing variants used a radius of 0.5. Note that only three optimizers obtained more losses than wins – all three being charged VEPSO with a core radius of 0.5. This poor performance can be explained by explosive acceleration attributed to the inverse square law. Conversely,



(a) dMOP1 and dMOP2 (b) dMOP3 and FDA1 (c) FDA3$_{Camara}$

Fig. 1: True POFs using $\tau_t = 5$ and $n_t = 20$ for 1000 iterations

TABLE II: Overall Wins and Losses

n_t	τ_t	PM	Result	C	D	C-8-1	C-32-1	C-8-0.5	C-16-0.5	C-32-0.5	C-8-2	C-16-2	C-32-2
all	all	all	Wins	185.27	191.01	113.82	109.03	53.26	36.46	41.55	96.44	93.23	90.28
all	all	all	Losses	71.78	89.69	62.41	66.57	161.19	162.16	161.23	84.72	76.33	74.27
all	all	all	Diff	113.49	101.32	51.41	42.46	-107.94	-125.70	-119.68	11.72	16.91	16.01
all	all	all	Rank	**1**	2	3	4	8	10	9	7	5	6

the performance of charged VEPSO also degraded when using a core radius of 2.

The performance of the optimizers for various performance measures is summarized in Table III. With respect to the accuracy measure, DVEPSO obtained the highest overall difference score, at 69.61. However, the difference score of the standard charged VEPSO was only 4.05 (5.82%) less than that of DVEPSO, which indicated relatively close performance between the two optimizers. Other variants of charged VEPSO demonstrated significantly worse performance than the variant with standard parameters; the next closest in performance used charge of 32 and a core radius of 1, had obtained only 15.94 more wins than losses, i.e., 77.10% less than that of DVEPSO. The worst overall performance when the *acc* measure was considered was depicted by charged VEPSO with a charge of 32 and a core radius of 0.5. The results for the *acc* measure indicate that DVEPSO would, in general, find approximation fronts which were closer to the true front than those found by charged VEPSO.

When examining the results in Table III with respect the *stab* measure, an opposite trend was observed than with the *acc* measure. Namely, DVEPSO was outperformed by six variants of charged VEPSO when stability was considered. Overall best performance with respect to stability was observed by charged VEPSO using a charge of 8 and a core radius of 1. Interestingly, the standard charged VEPSO was outperformed by the variants which used a core radius of 2. The worst overall performance was obtained when charged VEPSO used a charge of 16 and a core radius of 0.5, with similarly poor performance demonstrated by the other two variants which used a core radius of 0.5. The results for the *stab* measure indicate that, in general, the performance of charged VEPSO was more stable than DVEPSO when tracking the changing Pareto front.

Regarding the results for the number of solutions, as seen in Table III, the overall best performance was obtained by the standard charged VEPSO, which demonstrated 43.59 more wins than losses. As with the stability measure, the worst performance was observed when charged VEPSO with a charge of 16 and core radius of 0.5 was used. Although the standard charged VEPSO was the overall winner, DVEPSO consistently found more solutions than most variants of charged VEPSO.

B. Performance on Various Dynamic Environment Types

This section discusses the results obtained when considering the dynamic behavioral classes defined by Duhain [20]. Table IV presents the performance of the optimizers using various temporal and spatial severity settings.

When considering the quasi-static environments (n_t = 1, τ_t = 50), only two optimizers showed more wins than losses, i.e., DVEPSO and standard charged VEPSO, with DVEPSO obtaining the highest rank. DVEPSO obtained a difference of 29.70, while the standard charged VEPSO scored 20.60. All other optimizers showed significantly worse performance, with the third ranking optimizer having only 2.50 more losses than wins. The lowest ranking optimizer was the charged VEPSO when a charge of 32 and core radius of 1 was used.

Similar trends, as observed with the quasi-static environments, were demonstrated when considering the abrupt environments (n_t = 20, τ_t = 50). DVEPSO and the standard charged VEPSO were, again, the two highest ranked optimizers but had switched ranks when the larger environmental changes were present. As with quasi-static environments, a majority of charged VEPSO optimizers showed more losses than wins. However, when a charge of 8 and core radius of 0.5 were used, charged VEPSO scored 5.30 more wins than losses in the abrupt environments. The lowest ranked optimizer in the abrupt environments was charged VEPSO when the charge was 8 and core radius was 2.

When considering the progressive environments (n_t = 1, τ_t = 5) charged VEPSO with a charge of 32 and core radius of 1 obtained the highest rank and scored 31.96 more wins

TABLE III: Overall Wins and Losses for Various Performance Measures

n_t	τ_t	PM	Result	C	D	C-8-1	C-32-1	C-8-0.5	C-16-0.5	C-32-0.5	C-8-2	C-16-2	C-32-2
all	all	*acc*	Wins	94.28	97.31	44.37	41.97	15.52	13.18	12.44	31.57	34.77	33.62
all	all	*acc*	Losses	28.72	27.70	29.64	26.03	65.49	66.83	67.56	40.43	36.24	30.39
all	all	*acc*	Diff	65.56	69.61	14.73	15.94	-49.97	-53.65	-55.12	-8.86	-1.47	3.23
all	all	*acc*	Rank	2	**1**	4	3	8	9	10	7	6	5
all	all	*stab*	Wins	22.74	21.54	20.32	19.56	11.63	9.68	11.26	18.64	18.83	18.01
all	all	*stab*	Losses	18.39	26.76	5.67	7.71	29.74	29.88	29.23	7.53	8.89	8.41
all	all	*stab*	Diff	4.35	-5.23	14.65	11.86	-18.11	-20.21	-17.97	11.11	9.95	9.60
all	all	*stab*	Rank	6	7	**1**	2	9	10	8	3	4	5
all	all	*NS*	Wins	68.26	72.17	49.13	47.50	26.11	13.61	17.85	46.24	39.64	38.66
all	all	*NS*	Losses	24.67	35.24	27.10	32.84	65.97	65.45	64.44	36.77	31.21	35.48
all	all	*NS*	Diff	43.59	36.94	22.03	14.66	-39.86	-51.84	-46.60	9.47	8.43	3.18
all	all	*NS*	Rank	**1**	2	3	4	8	10	9	5	6	7

than losses. The other two charged VEPSO variants with core radii of 1 obtained second and third ranks, respectively. The performance of charged VEPSO indicated that the optimizer performed well in the faster-paced, progressive environments. For the progressive environments, the DVEPSO optimizer obtained a rank of 4 over all performance measures. The three worst performing optimizers were all charged VEPSO variants with a core radius of 0.5.

Similar to the progressive environments, chaotic environments (n_t = 20, τ_t = 5) were optimized best by charged VEPSO with a core radius of 1. All three of the top optimizers were charged VEPSO with a core radius of 1, where each obtained the same rank as they had in the progressive environments. A charge of 32 and core radius of 1 lead to the highest difference score while a core radius of 0.5 and charge of 8 lead to the worst performance on the chaotic environments. DVEPSO ranked sixth for the chaotic environments, showing degraded performance from the progressive environments, where DVEPSO had ranked fourth.

C. Performance on Various Dynamic Multi-Objective Optimization Problem Types

This section presents the results obtained for the various DMOOP types defined by Farina *et al.* [1] as shown in Tables V to VII.

When considering the performance on Type I DMOOPs, as shown in Table V, DVEPSO significantly outperformed all charged VEPSO variants with respect to the *acc* and *NS* measures, while the standard charged VEPSO ranked second in both these measures. The difference between the number of wins and losses for DVEPSO was more than double that of the standard charged VEPSO with respect to both *acc* and *NS*. However, when considering the *stab* measure, DVEPSO was outperformed by all variants of charged VEPSO. The worst performance with regards to the accuracy and number of solutions were charged VEPSO with a charge of 8 and core radius of 2 and charged VEPSO with a charge of 32 and core radius of 1, respectively. Interestingly, charged VEPSO performed better when using a core radius of 0.5 than with a core radius of 2, a behavior which is reversed from the overall results presented in Table II.

Table VI outlines the performance of the various algorithms on Type II DMOOPs. Contrary to the results on Type I DMOOPs, the standard charged VEPSO attained the highest rank on both the *acc* and *NS* measures, while DVEPSO ranked second on both measures. However, the difference in performance between the two algorithms was noticeably smaller than with Type I DMOOPs. When considering the *stab* measure, charged VEPSO with a charge of 8 and core radius of 0.5 demonstrated the best performance, while DVEPSO showed the worst performance as with Type I DMOOPs. The optimizers which demonstrated the worst performance with respect to the *acc* and *NS* measures were charged VEPSO with a core radius of 0.5 and charges of 16 and 32, respectively.

Performance of the optimizers on a Type III DMOOP is given in Table VII. For all three performance measures, the best optimizer was a variant of charged VEPSO. For the *acc* measure, the standard charged VEPSO ranked highest, while the charged VEPSO which employed a charge of 8 and core radius of 1 performed best with respect to the *stab* and *NS* measures. As previously observed, variants of charged VEPSO with a core radius of 0.5 were the only optimizers to obtain more losses than wins, leading to a negative difference score for all three performance measures. However, the charge value which lead to the worst performance for each measure was not consistent with each of the three charge values being ranked

D. Summary and Discussion of Results

In general, DVEPSO performed best with respect to the accuracy measure. However, charged VEPSO demonstrated better performance in terms of stability. The superior stability and inferior accuracy of the charged VEPSO can be attributed in part to characteristics of the charged VEPSO optimizer. The repulsive behavior of the charged particles would cause some particles to be "repelled" from the Pareto front once located by other particles, leading to a lower accuracy score. Charged VEPSO implicitly addressed diversity via the CPSO sub-swarms and also reduced the drastic changes to particle topology from reinitializing particles, explaining the increased stability. With the above observations, it is shown that DVEPSO was better at locating the Pareto front while charged VEPSO was less susceptible to environmental changes and was better able to track the Pareto front over time. Further evidence is provided when considering the performance on Farina *et al.*'s DMOOP types [1]. DVEPSO demonstrated the highest accuracy on Type I DMOOPs (POF does not change) while on Type II and Type III DMOOPs (POF does change), the standard charged VEPSO attained the best scores for accuracy. This indicates that the accuracy of DVEPSO degrades when the Pareto front is non-static.

When considering Duhain's environmental behavior classes [20], it was observed that charged VEPSO performed significantly better than DVEPSO in the faster-paced progressive

TABLE IV: Overall Wins and Losses for Various Temporal and Spatial Severity Settings

n_t	τ_t	PM	Result	C	D	C-8-1	C-32-1	C-8-0.5	C-16-0.5	C-32-0.5	C-8-2	C-16-2	C-32-2
1	50	all	Wins	33.80	38.85	7.55	3.85	9.10	3.95	3.25	8.80	4.80	3.20
1	50	all	Losses	13.20	9.15	10.55	14.30	11.60	10.75	10.80	13.55	13.40	9.85
1	50	all	Diff	20.60	29.70	-3.00	-10.45	-2.50	-6.80	-7.55	-4.75	-8.60	-6.65
1	50	all	Rank	2	**1**	4	10	3	7	8	5	9	6
1	5	all	Wins	54.77	57.13	56.59	53.88	16.68	16.69	20.27	45.80	48.35	47.00
1	5	all	Losses	24.33	33.92	25.48	21.92	71.41	70.45	71.35	36.48	28.79	33.03
1	5	all	Diff	30.45	23.21	31.11	31.96	-54.74	-53.76	-51.08	9.32	19.57	13.97
1	5	all	Rank	3	4	2	**1**	10	9	8	7	5	6
20	50	all	Wins	60.20	53.65	12.10	10.95	21.65	10.65	11.10	8.15	5.35	8.10
20	50	all	Losses	23.10	24.60	15.40	19.05	16.35	20.35	21.25	24.80	19.20	17.80
20	50	all	Diff	37.10	29.05	-3.30	-8.10	5.30	-9.70	-10.15	-16.65	-13.85	-9.70
20	50	all	Rank	**1**	2	4	5	3	6	8	10	9	6
20	5	all	Wins	36.50	41.39	37.58	40.35	5.83	5.17	6.93	33.69	34.73	31.98
20	5	all	Losses	11.16	22.02	10.98	11.30	61.83	60.61	57.83	9.89	14.94	13.59
20	5	all	Diff	25.35	19.37	26.61	29.05	-56.00	-55.44	-50.91	23.80	19.79	18.39
20	5	all	Rank	3	6	2	**1**	10	9	8	4	5	7

TABLE V: Overall Wins and Losses on FDA1 and dMOP3 (Type I DMOOPs)

n_t	τ_t	PM	Result	C	D	C-8-1	C-32-1	C-8-0.5	C-16-0.5	C-32-0.5	C-8-2	C-16-2	C-32-2
all	all	*acc*	Wins	23.42	31.62	14.48	13.97	12.24	8.56	8.32	5.47	7.14	7.81
all	all	*acc*	Losses	15.58	10.39	11.53	9.03	10.77	11.44	13.69	21.53	15.87	13.20
all	all	*acc*	Diff	7.84	21.23	2.95	4.94	1.47	-2.88	-5.37	-16.06	-8.73	-5.39
all	all	*acc*	Rank	2	**1**	4	3	5	6	7	10	9	8
all	all	*stab*	Wins	6.90	4.13	2.67	2.80	3.92	2.00	3.23	1.12	1.37	1.46
all	all	*stab*	Losses	2.25	8.29	1.72	2.34	1.43	1.44	1.71	3.43	3.02	3.96
all	all	*stab*	Diff	4.65	-4.17	0.95	0.46	2.49	0.56	1.52	-2.31	-1.66	-2.50
all	all	*stab*	Rank	**1**	10	4	6	2	5	3	8	7	9
all	all	*NS*	Wins	23.67	37.67	14.08	9.39	22.91	10.52	11.37	13.68	9.01	7.85
all	all	*NS*	Losses	12.94	11.70	15.31	21.18	14.26	12.04	16.14	20.70	17.43	18.44
all	all	*NS*	Diff	10.73	25.97	-1.23	-11.80	8.65	-1.52	-4.77	-7.02	-8.42	-10.59
all	all	*NS*	Rank	2	**1**	4	10	3	5	6	7	8	9

TABLE VI: Overall Wins and Losses on FDA3 and dMOP2 (Type II DMOOPs)

n_t	τ_t	PM	Result	C	D	C-8-1	C-32-1	C-8-0.5	C-16-0.5	C-32-0.5	C-8-2	C-16-2	C-32-2
all	all	*acc*	Wins	46.58	42.48	17.32	17.74	2.40	2.76	3.25	16.03	16.39	16.08
all	all	*acc*	Losses	8.43	10.52	10.69	10.26	36.61	37.24	35.75	9.98	11.62	9.93
all	all	*acc*	Diff	38.15	31.96	6.63	7.48	-34.21	-34.48	-32.50	6.05	4.77	6.15
all	all	*acc*	Rank	**1**	2	4	3	9	10	8	6	7	5
all	all	*stab*	Wins	8.10	5.33	4.27	4.40	5.52	3.60	4.83	2.72	2.97	3.06
all	all	*stab*	Losses	8.65	14.69	2.02	2.64	1.73	1.74	2.01	3.73	3.32	4.26
all	all	*stab*	Diff	-0.55	-9.37	2.25	1.76	3.79	1.86	2.82	-1.01	-0.36	-1.20
all	all	*stab*	Rank	7	10	3	5	**1**	4	2	8	6	9
all	all	*NS*	Wins	32.91	34.88	17.78	18.06	12.47	5.44	5.81	18.61	11.18	12.33
all	all	*NS*	Losses	8.90	14.42	12.06	8.87	24.34	25.19	25.95	18.04	15.26	16.45
all	all	*NS*	Diff	24.02	20.47	5.73	9.19	-11.87	-19.75	-20.14	0.57	-4.08	-4.13
all	all	*NS*	Rank	**1**	2	4	3	8	9	10	5	6	7

TABLE VII: Overall Wins and Losses on dMOP1 (Type III DMOOP)

n_t	τ_t	PM	Result	C	D	C-8-1	C-32-1	C-8-0.5	C-16-0.5	C-32-0.5	C-8-2	C-16-2	C-32-2
all	all	*acc*	Wins	24.29	23.21	12.58	10.26	0.89	1.86	0.88	10.08	11.25	9.74
all	all	*acc*	Losses	4.72	6.79	7.43	6.74	18.12	18.15	18.13	8.93	8.76	7.27
all	all	*acc*	Diff	19.57	16.42	5.15	3.52	-17.23	-16.29	-17.25	1.15	2.49	2.47
all	all	*acc*	Rank	**1**	2	3	4	9	8	10	7	5	6
all	all	*stab*	Wins	7.22	8.21	7.83	7.96	3.39	3.50	4.11	8.26	10.17	7.93
all	all	*stab*	Losses	3.57	5.49	2.06	4.43	13.99	13.94	13.79	2.62	5.31	3.35
all	all	*stab*	Diff	3.65	2.72	5.77	3.54	-10.61	-10.45	-9.69	5.64	4.86	4.58
all	all	*stab*	Rank	5	7	**1**	6	10	9	8	2	3	4
all	all	*NS*	Wins	11.66	8.56	14.44	12.35	0.68	0.62	2.65	10.87	11.32	11.06
all	all	*NS*	Losses	6.08	8.22	2.39	4.49	15.90	15.95	15.90	5.96	4.53	4.78
all	all	*NS*	Diff	5.58	0.34	12.05	7.87	-15.23	-15.33	-13.25	4.92	6.79	6.28
all	all	*NS*	Rank	5	7	**1**	2	9	10	8	6	3	4

and chaotic environment types. However, when considering the slower-paced quasi-static and abrupt environments, DVEPSO and the standard charged VEPSO performed very similarly. Another observation was made regarding the spatial severity settings. DVEPSO degraded in performance on the environments with larger changes in the environment (i.e., n_t = 20). That is, DVEPSOs rank worsened when switching from quasi-static to abrupt environments as well as when changing from progressive to chaotic environments. Thus, charged VEPSO was more adequate to handle both faster-paced as well as larger changes in the environment.

VI. Conclusion

This paper proposed a new variant of vector evaluated particle swarm optimization (VEPSO) designed to optimize dynamic multi-objective optimization problems (DMOOPs), namely charged VEPSO. Charged VEPSO made use of charged particle swarm optimization (CPSO) sub-swarms to implicitly address decision-space tracking of environmental changes. Various configurations of CPSO parameters were examined to determine their effect on the algorithm's performance. Variants of charged VEPSO were compared to the predecessor, dynamic VEPSO (DVEPSO) on a wide variety of dynamic environments. Five benchmark functions, spanning three of Farina *et al.*'s DMOOP Types, were examined. Each benchmark function was configured with four spatial and temporal severity settings, representing each of Duhain's four dynamic environment behavioral classes. The accuracy, stability, and number of solution measures calculated and used to assign a relative ranking to each optimizer based on normalized, per-iteration performance.

Results over all experiments indicated that, in general, the charged VEPSO optimizer using the standard CPSO parameters outperformed the existing DVEPSO optimizer. However, DVEPSO outperformed all other, non-standard variants of charged VEPSO. Results demonstrated that DVEPSO was more well-suited to locate the Pareto optimal front, while charged VEPSO was better equipped to track the changing Pareto front. Further, DVEPSO was shown to perform better on environments where the Pareto front does not change over time. Charged VEPSO exhibited better performance on the faster-paced, progressive and chaotic environments.

Future work includes further methods to improve the performance of VEPSO in dynamic environments. A wider set of benchmark functions and environmental configurations is needed to further categorize the performance of optimizers. An immediate extension to this research is to compare the performance of charged VEPSO, and other dynamic sub-swarm VEPSO variants, against state-of-the-art dynamic multi-objective optimization algorithms.

Acknowledgment

The authors would like to thank Mardé Helbig of The Council for Scientific and Industrial Research (CSIR), South Africa, for her assistance in generating the true Pareto fronts. Furthermore, the authors thank the anonymous reviewers for their valuable insight and suggestions.

References

[1] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 5, pp. 425–442, 2004.

[2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[3] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proc. of the 2002 ACM Symposium on Applied Computing*. NY: ACM, 2002, pp. 603–607.

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *IEEE Int'l Conference on Neural Networks*, vol. IV, 1995, pp. 1942–1948.

[5] J. Grobler, "Particle swarm optimization and differential evolution for multi-objective multiple machine scheduling," Master's thesis, U. of Pretoria, South Africa, 2008.

[6] K. R. Harrison, B. Ombuki-Berman, and A. P. Engelbrecht, "Knowledge transfer strategies for vector evaluated particle swarm optimization," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 7811, pp. 171–184.

[7] M. Greeff and A. P. Engelbrecht, "Solving dynamic multi-objective problems with vector evaluated particle swarm optimisation," in *IEEE Congress on Evolutionary Computation (CEC 2008)*. IEEE, 2008, pp. 2917–2924.

[8] M. Greeff and A. Engelbrecht, "Dynamic multi-objective optimisation using pso," in *Multi-Objective Swarm Intelligent Systems*, ser. Studies in Computational Intelligence. Springer, 2010, vol. 261, pp. 105–123.

[9] T. M. Blackwell, P. J. Bentley *et al.*, "Dynamic search with charged swarms," in *Proceedings of the genetic and evolutionary computation conference*. Citeseer, 2002, pp. 19–26.

[10] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 459–472, 2006.

[11] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 2. IEEE, 2002, pp. 1666–1670.

[12] A. J. Carlisle, "Applying the particle swarm optimizer to non-stationary environments," Ph.D. dissertation, Auburn University, Auburn, AL, USA, 2002.

[13] M. Helbig and A. P. Engelbrecht, "Analyses of guide update approaches for vector evaluated particle swarm optimisation on dynamic multi-objective optimisation problems," in *Evolutionary Computation*, ser. CEC '12. IEEE, 2012, pp. 1–8.

[14] G. Pampara, A. P. Engelbrecht, and T. Cloete, "Cilib: A collaborative framework for computational intelligence algorithms - part i," in *Proc. of IEEE World Congress on Computational Intelligence*, Hong Kong, 2008, pp. 1750–1757, source code available at: https://github.com/cilib/cilib.

[15] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, U. of Pretoria, South Africa, 2002.

[16] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 1, pp. 103–127, 2009.

[17] M. Cámara, J. Ortega, and F. de Toro, "Approaching dynamic multi-objective optimization problems by using parallel evolutionary algorithms," in *Advances in Multi-Objective Nature Inspired Computing*. Springer, 2010, pp. 63–86.

[18] K. De Jong, "Evolving in a changing world," in *Foundations of Intelligent Systems*, ser. Lecture Notes in Computer Science. Springer, 1999, vol. 1609, pp. 512–519.

[19] K. Weicker, "An analysis of dynamic severity and population size," in *Parallel Problem Solving from Nature PPSN VI*, ser. Lecture Notes in Computer Science, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, Eds. Springer Berlin Heidelberg, 2000, vol. 1917, pp. 159–168.

[20] J. Duhain and A. Engelbrecht, "Towards a more complete classification system for dynamically changing environments," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, June 2012, pp. 1–8.

[21] M. Cámara, J. Ortega, and F. de Toro, "A single front genetic algorithm for parallel multi-objective optimization in dynamic environments," *Neurocomputing*, vol. 72, no. 16, pp. 3570–3579, 2009.

[22] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[23] M. Helbig and A. P. Engelbrecht, "Analysing the performance of dynamic multi-objective optimisation algorithms," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 1531–1539.