HANDLING DYNAMIC MULTIOBJECTIVE PROBLEMS WITH PARTICLE SWARM OPTIMIZATION

Alan Díaz Manríquez, Gregorio Toscano Pulido and José Gabriel Ramírez Torres
Laboratorio de Tecnologías de Información, CINVESTAV-Tamaulipas
Km. 6 carretera Cd. Victoria-Monterrey. Cd. Victoria, Tamaulipas, 87267, Mexico

Keywords: Dynamic multi-objective optimization, Particle swarm optimization, Multi-objective optimization.

Abstract: In this paper the hyperplane distribution and Pareto dominance were incorporated into a particle swarm optimization algorithm in order to allow it to handle dynamic multiobjective problems. When a change in a dynamic multiobjectve function is detected, the proposed algorithm reinitializes (in different ways) the PSO's velocity parameter and the archive where the non-dominated solutions are beeing stored such that the algorithm can follow the dynamic Pareto front. The proposed approach is validated using two dynamic multiobjective test functions and an standard metric taken from the specialized literature. Results indicate that the proposed approach is highly competitive which can be considered as a viable alternative in order to solve dynamic multiobjective optimization problems.

1 INTRODUCTION

Since life is dynamic, it is only natural to expect that the problems from daily life are dynamics. Robot path planning, or the selection of routes in a communications network are some problems whose function fluctuates over time or incorporates some kind of noise. These problems are called "non-stationary" or "dynamic".

A dynamic optimization problem (DOP) may involve two or more functions to be optimized simultaneously (also known as dynamic multiobjective optimization problems, or DMOP for short), as well as constraints and parameters which can be changed over time. Dealing with a DMOP increases the complexity of a dynamic problem, since it is imperative to detect the change in order to re-evaluate the previously stored solutions. Also, it can increase the number of objective functions, and therefore the optimization process may change dramatically.

Although the study of this type of problems is not new, most of the proposed approaches transform the original dynamic problem into many static optimization problems. The scientific community of evolutionary computation has focused their efforts on designing approaches to manage a set of valid solutions (population) to solve these problems without performing any transformation. The resulting approaches can take advantage of previous knowledge to direct the search.

Kennedy and Eberhart proposed an approach called "Particle Swarm Optimization" (PSO) which was inspired on the choreography of a bird flock. Like other evolutionary algorithms, PSO uses a set of possible solutions which will be "evolved" until an optimal solution or a termination criteria is reached. In this case, each solution (x) is represented by a particle, and a set of particles are represented by a swarm. The responsibility of evolving (moving) the swarm to the optimal region corresponds to the velocity equation. This equation is usually composed by three elements: a velocity inertia, a cognitive component (pbest) and a social component (gbest). The entire approach can be seen as a distributed behavioral algorithm that performs (in its more general version) a multidimensional search. In the simulation, the behavior of each particle is affected by either the best local particle (i.e., within a certain neighborhood) or the best global particle (Kennedy and Eberhart, 2001).

An interesting aspect of PSO is that it allows individuals to benefit from their past experiences (note that in other approaches such as the genetic algorithm, normally the current population is the only "memory" used by the individuals).

The remain of this paper is organized as follows: Basic concepts are given in Section 2. In Section 3, we present the dynamic multiobjective state-of-the-art. Section 4 describes the proposed algorithm and the components that it is conformed by. Section 5 presents the experiments and the comparison of results. Finally, Section 6 provides the concluding remarks and future work.

2 BASIC CONCEPTS

Definition 1 (Dynamic Multiobjective Optimization Problem (DMOP)). *Find $\vec{x}$ which minimizes: $\vec{f}(\vec{x},t) = [f_1(\vec{x},t), f_2(\vec{x},t), ..., f_k(\vec{x},t)]^T$, subject to m inequality constraints: $g_i(\vec{x},t) \leq 0 \quad i = 1,2,...,m$, and p equality constraints: $h_j(\vec{x},t) = 0 \quad j = 1,2,...,p$, where $\vec{x}$ is the vector of decision variables; $\vec{f}$ is the set of objective functions to be minimized in time t. The functions g and h, represent the set of constraints, that define the feasible region $\mathcal{F}$ in time t.* □

Definition 2 (Pareto Optimality). *A point $\vec{x}^* \in \mathcal{F}$ is* **Pareto optimal** *in time t, if for every $\vec{x} \in \mathcal{F}$ and $I = \{1,2,\ldots,k\}$ either, $\forall_{i \in I}(f_i(\vec{x},t) = f_i(\vec{x}^*,t))$ or, there is at least one $i \in I$ such that $f_i(\vec{x},t) > f_i(\vec{x}^*,t)$* □

Definition 3 (Pareto Dominance). *A vector $\vec{x} = [x_1,\ldots,x_k]^T$ is said to dominate $\vec{y} = (y_1,\ldots,y_k)$ (denoted by $\vec{x} \preceq \vec{y}$) if and only if x is partially less than y, i.e., $\forall i \in \{1,\ldots,k\}$, $x_i \leq y_i \wedge \exists i \in \{1,\ldots,k\} : x_i < y_i$.* □

Definition 4 (Pareto-Optimal Set). *For a given time t and a given MOP $\vec{f}(x,t)$, the Pareto optimal set ($\mathcal{P}^*$) is defined as:*

$$\mathcal{P}^t := \{x \in \mathcal{F} \mid \nexists\, x' \in \mathcal{F} \;\; \vec{f}(x',t) \preceq \vec{f}(x,t)\}. \quad (1)$$

□

Definition 5 (Pareto Front). *For a given MOP $\vec{f}(x,t)$, and Pareto optimal set $\mathcal{P}^t$, in time t, the Pareto front ($\mathcal{PF}^t$) is defined as:*

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x,t),\ldots,f_k(x,t)) \mid x \in \mathcal{P}^*\}. \quad (2)$$

□

In the general case, it is impossible to find an analytical expression of the line or surface that contains these points. The normal procedure to generate the Pareto front is to compute the feasible points $\mathcal{F}$ and their corresponding $f(\mathcal{F})$. When there is a sufficient number of these, it is then possible to determine the non-dominated points and to produce the Pareto front.

Furthermore, DMOP can be clustered into four types (Farina et al., 2004):

- **Type I.** Change on the Pareto-optimal set ($\mathcal{P}^t$), whereas the Pareto-optimal front (optimal objective values) ($\mathcal{PF}^t$) does not change.
- **Type II.** Both $\mathcal{P}^t$ and $\mathcal{PF}^t$ change.
- **Type III.** $\mathcal{P}^t$ does not change, whereas $\mathcal{PF}^t$ changes.
- **Type IV.** Both $\mathcal{P}^t$ and $\mathcal{PF}^t$ do not change, although the problem can dynamically change (e.g., the constraints can vary).

Since Types II and III are the most challenging type of DMOP, we focused in solving these kind of problems.

3 RELATED WORK

Evolutionary algorithms have been successfully applied to solve DMOPs. Their succeed might be directed by their population-based nature, since this allows to use the most of the previous discovered knowledge in order to follow the change in the environment.

Bingul (Bingul, 2007) solved a dynamic multiobjective optimization problem (DMOP) using an aggregating function approach with a Genetic Algorithm (GA). Zeng *et al.* (Zeng et al., 2006) introduced a dynamic orthogonal multiobjective evolutionary called DMOEA. Their approach selects randomly between an orthogonal crossover operator and a linear crossover operator. The first operator was proposed with the aim to enhance the fitness of the population while the process is stabilized between two changes. Hatzakis and Wallace (Hatzakis and Wallace, 2006) proposed a forward-looking approach which combines a forecasting technique with an evolutionary algorithm. Deb *et al.* (Deb et al., 2000) proposed two modifications to the NSGA-II in order to be able to handle DMOPs. In the first modification, the population is reinitialized whilst in the second the population is hyper-mutated depending on the type of change in the environment (Deb et al., 2006). Talukder and Kirley (Talukder and Kirley, 2008) used a new variation operator to follow the Pareto front in DMOPs. Zhou *et al.* (Zhou et al., 2006) proposed two strategies to perform a population re-initialization when a change in the environment is detected. Wang and Dang transform the objectives of the original DMOP in a set of static bi-objective (Yuping Wang, 2008). Ray *et al.* (Ray et al., 2009) introduced a memetic algorithm which employs an orthogonal ε-constrained formulation to deal with multiple objectives and a sequential quadratic programming (SQP) solver is embedded as its local search mechanism in order to improve the convergence rate.

4 PROPOSED APPROACH

PSO has been successfully used for both continuous nonlinear and discrete binary single objective optimization (Kennedy and Eberhart, 2001). The pseudocode of the PSO pseudocode is shown in Algorithm 1.

Algorithm 1. PSO Algorithm.

```
$\vec{gbest} \leftarrow \vec{x_0}$
for $i = 0$ to $nparticles$ do
    $\vec{pbest_i} \leftarrow \vec{x_i} \leftarrow initialize\_randomly()$
    $fitness_i \leftarrow f(\vec{x_i})$
    if $fitness_i < f(\vec{gbest})$ then
        $\vec{gbest} \leftarrow \vec{x_i}$
    end if
end for
repeat
    for $i = 0$ to $nparticles$ do
        for $d = 0$ to $ndimensions$ do
            $v_{id} \leftarrow W * v_{id} + C_1 * U(0,1) * (pbest_{id} - x_{id}) + C_2 * U(0,1) *$
            $(gbest - x_{id})$
            $x_{id} \leftarrow x_{id} + v_{id}$
        end for
        $fitness_i \leftarrow f(\vec{x_i})$
        if $fitness_i < f(\vec{pbest_i})$ then
            $\vec{pbest_i} \leftarrow \vec{x_i}$
        end if
        if $fitness_i < f(\vec{gbest_i})$ then
            $\vec{gbest_i} \leftarrow \vec{x_i}$
        end if
    end for
until Termination criterion
```

4.1 Handling Multiple Dynamic Objectives with PSO

PSO seems particularly suitable for dynamic multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single-objective optimization. Based on such behavior, one would expect that a multiobjective PSO (MOPSO) to be very efficient computationally speaking. However, in order to be able to handle dynamic multiobjective problems there is necessary to perform three main modifications to the original algorithm.

- To modify the algorithm to handle multiple objectives and produce a set of non-dominated solutions in a single run.
- To modify the algorithm to obtain a good distribution of solutions.
- To modify the algorithm in order to handle its behavior when a change is detected.

A natural modification to a PSO algorithm aimed handle multiple objectives is replace the comparison operator in order to determine whether a solution *a* is better a solution *b*. The analogy of particle swarm optimization with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme is a straightforward way to extend the approach to handle multiobjective optimization problems. However, if we merge a Pareto ranking scheme with the PSO algorithm a set of non-dominated solutions will be produced (by definition, all non-dominated solutions are equally good). Having several non-dominated solutions implies the inclusion into the algorithm of both: an additional criteria to decide whether a new non-dominated solution is pbest or gbest and a strategy to select the guide particles (pbest and gbest).

In order to select an strategy to choose the gbest solution, we performed three experiments:

1. To choose randomly among all the particles in the archive.
2. To choose the non-dominated particle closer to pbest.
3. To choose the non-dominated particle more distant to the pbest.

Experimental results indicate us that the best strategy to select the gbest was: 1) random selection.

Several approaches have suggest that the use of elitism by means of an external archive in order to store the non-dominated solutions can enable most algorithms to find the Pareto front. However, the size of the archive can grow very fast, and therefore, it is imperative to maintain a small set of non-dominated solutions. We selected the hyper-plane distribution algorithm in order to maintain diversity and to reduce the size of non-dominated solutions stored in the archive(Blinded, 2005).

4.2 Hyper-plane Distribution

The core idea of this proposal is to perform a good distribution of the hyper-plane space defined by the minima (assuming minimization) from the objectives, and use such distribution to select a representative subset from the whole set of non-dominated solutions. The algorithm works as follows: First it accepts a set of non-dominated vectors and a number *n* of solutions of the desirable subset as its input. Then, the algorithm selects those vectors which have the minimum and maximum value of each objective, and it groups them into two sets, the minima set (called *MIN*), and the maxima set (called *MAX*). Using *MIN*, the algorithm creates a hyper-plane, and distributes its space into $n-1$ fixed-size sub hyper-planes. After that, it computes lines on each subdivision; such lines are perpendicular to the hyper-plane. Finally, the algorithm returns the closest vectors to each line.

4.3 Adaption to the Change

Since the natural behavior of a DMOP is to be changing, it is essential that the algorithm performs a good reaction when a change is detected. Therefore, in order to properly follow the Pareto Front, we need to reinitialize the PSO algorithm. However, such reinitialization can be made using four strategies:

- **DPSO-1.** The current solution (after the change) will be taken as *pbest*. The particles are reevaluated and the resulting non-dominated solutions are stored in the archive.
- **DPSO-2.** This is similar to DPSO-1. However, in this strategy, the archive will be updated using the hyper-plane distribution.
- **DPSO-3.** Similar to DPSO-1, but the velocity of each particle will be reinitialized to zero.
- **DPSO-4.** This strategy is similar to DPSO-2, but this strategy reinitializes the velocity of each particle (to zero).

5 EXPERIMENTS AND COMPARISON OF RESULTS

Two test functions were taken from the specialized literature to compare our approaches. In order to allow a quantitative assessment of the performance of a multiobjective optimization algorithm, the Inverted Generational Distance (IGD) metric was adopted.

Inverted Generational Distance (IGD): The concept of generational distance was introduced by Van Veldhuizen & Lamont (Veldhuizen and Lamont, 1998; Veldhuizen and Lamont, 2000) as a way of estimating how far are the elements in the Pareto front produced by our algorithm from those which belongs to the true Pareto front of the problem. This measure is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \qquad (3)$$

where n is the number of non-dominated vectors found by the algorithm being analyzed and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the true Pareto front. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the true Pareto front of the problem.

In order to know how competitive is our approach, we decided to compare our results with respect to those obtained by the NSGA II-A and the NSGA II-B (Deb et al., 2006), these algorithms use the parameter ζ that is the percentage of population which will be reinitialized in the NSGA II-A and hyper-mutated in the NSGA II-B, this variation operators would be applied when a change is detected.

In all the following examples, we report the results obtained from 100 independent runs of each compared algorithm.

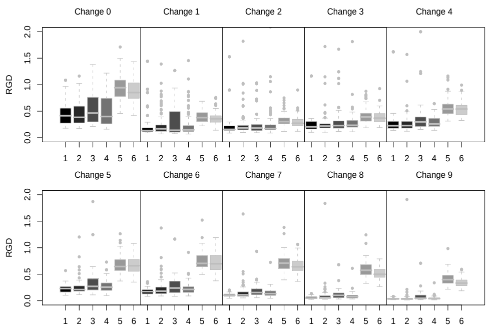
In order to have a fair comparison, we hand-tune each change to be activated every 500 evaluations of the objective function. For such sake, we setup the population of NSGA II-A and NSGA II-B to 52 individuals and $\tau_T = 10$, $n_t = 10$, and $\zeta = 20\%$. The DPSO-1, DPSO-2, DPSO-3 and DPSO-4 were executed using 20 particles, the size of the archive was $n = 100$, and the parameters $W = 0.4$ and $C_1 = C_2 = 1.49$.

The tests functions used for the algorithms are shown in the Table 1.

Table 1: Tests functions.

FDA1
$f_1(X_i) = x_1, g(X_{II}) = 1 + \sum_{x_i \in X_{II}} (x_i - G(t))^2$
$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}, G(t) = sin(0.5\pi t)$
$f_2 = g * h(f_1, g)$
$t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor,$
τ is the generation counter and τ_T is the number of generations that t is fixed.
$X_I = (x_1)^T, x_1 \in [0,1], X_{II} = (x_2, ..., x_{x_n})^T,$ $x_2, ..., x_n \in [-1,1]$
FDA2$_{mod}$
$f_1(X_i) = x_1, g(X_{II}) = 1 + \sum_{x_i \in X_{II}} x_i^2$
$h(X_{III}, f_1, g) = 1 - \sqrt{\frac{f_1}{g}}^{e(t, X_{III})}$
$H(t) = 0.2 + 4.8t^2, t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor,$
$f_2 = g * h(f_1, g)$
τ is the generation counter τ_T is the number of generations that t is fixed.
$X_I = (x_1)^T, x_1 \in [0,1], X_{II} \in [-1,1]^{r_2}$
$X_{III} =\in [-1,1]^{r_3}, 1 + r_2 + r_3 = n$

The algorithms were executed until 10 changes have occurred in each test function. However, since there is a large amount of data, it would be very confused to display it in a numerical mode. Therefore, in order to presente the results in a more compact way, we decided to use box-plot graphics of the application of the IGD metric to the Pareto known before each change.


(a) Inverted generational distance results for FDA1

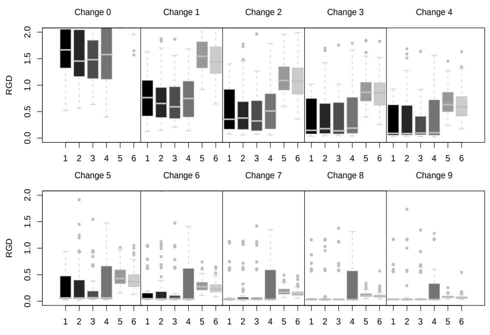

(b) Inverted generational distance results for FDA2$_{mod}$

Figure 1: Box-plot of the application of the inverted generational distance metric to the results produced 100 independent runs by 1) DPSO-1, 2) DPSO-2, 3) DPSO-3, 4) DPSO-4, 5) NSGA II-A and 6) NSGA II-B for both test functions: (a) and (b) refers to the inverted generational distance for FDA1 and FDA2, respectively.

Results obtained when FDA1 was optimized show that all versions of DPSO and the NSGA II-A and NSGA II-B behaved similarly. Results from IGD suggest, that all versions of DPSO had a better a convergence than NSGA II-A and NSGA II-B (see Figure 1(a)). When comparing the DPSO versions, we can say that the best performance were for those solutions algorithm that preserve the velocity value before the change (those who did not initialized the velocity to 0). Also, in Figure 5, we can see the Pareto fronts produced by the different algorithms. From this figure, it is clear that DPSO outperforms the NSGA II-A and NSGA II-B.

The results obtained for FDA2$_{mod}$ show that all versions of DPSO and the NSGA II-A and NSGA II-B also behaved similar. However, from the IGD metric shown in Figure 1(b), we can easily observed that the results from all versions of DPSO outperform to those solutions obtained by NSGA II-A and NSGA II-B. When comparing the DPSO versions, we can say that the reinitialization of the velocity plays a key role in the algorithm. Such that, in the initial changes of the fitness function, the algorithms which did not reinitialize the velocity (to zero), obtained better results.

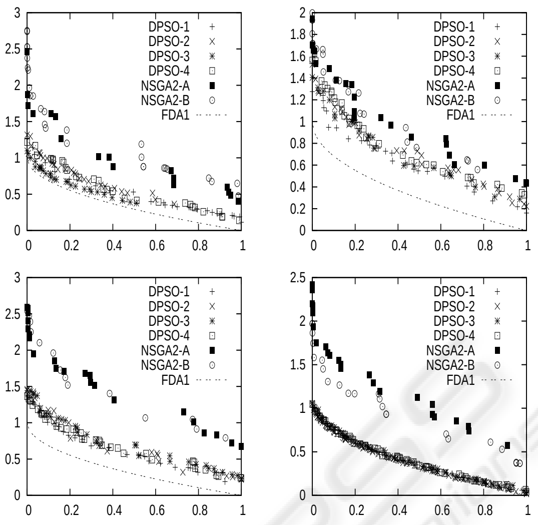


Figure 2: Pareto fronts produced by all versions of DPSO, NSGA II-A and NSGA II-B for FDA1 test function: change 1 is shown in the top left; change 3 is shown in the top right, change 6 is shown in the bottom left, change 9 is shown in the bottom right.

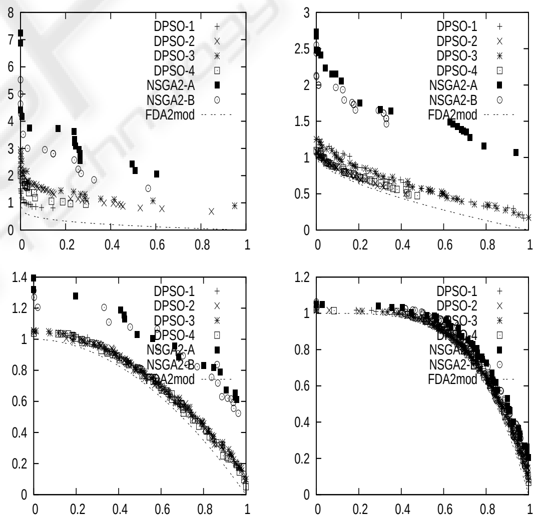


Figure 3: Pareto fronts produced by all versions of DPSO, NSGA II-A and NSGA II-B for FDA2$_{mod}$ test function: change 1 is shown in the top left; change 3 is shown in the top right, change 6 is shown in the bottom left, change 9 is shown in the bottom right.

6 CONCLUSIONS

We have presented a proposal to extend particle swarm optimization to handle dynamic multiobjective problems. The proposed approach was validated using the standard methodology currently adopted for the evolutionary multiobjective optimization community. Results indicate that our approach is a viable alternative since its performance is highly competitive with respect to some of the best dynamic multiobjective evolutionary algorithms known-to-date.

One aspect that we would like to explore in the future is to study how the DPSO behaves with a different the particles' interconnection topology. Furthermore, we would like to explore the use of different values for W, $C1$, and $C2$.

ACKNOWLEDGEMENTS

Acknowledgements The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Information Technology Laboratory at CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT through project 90548. Also, This research was partially funded by project number 51623 from "Fondo Mixto Conacyt-Gobierno del Estado de Tamaulipas". Finally, we would like to thank to Fondo Mixto de Fomento a la Investigación científica y Tecnológica CONACyT - Gobierno del Estado de Tamaulipas for the support to publish this paper.

REFERENCES

Bingul, Z. (2007). Adaptive Genetic Algorithms Applied to Dynamic Multi-ObjectiveProblems. *Appl. Soft Comput.*, 7(3):791–799.

Blinded (2005). *Blinded.* PhD thesis, Blinded, Blinded.

Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India.

Deb, K., N., U. B. R., and Karthik, S. (2006). Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In *EMO*, pages 803–817.

Farina, M., Deb, K., and Amato, P. (2004). Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications. *IEEE Transactions on Evolutionary Computation*, 8(5):425–442.

Hatzakis, I. and Wallace, D. (2006). Dynamic Multi-Objective Optimization with Evolutionary Algorithms: A Forward-Looking Approach. In et al., M. K., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, volume 2, pages 1201–1208, Seattle, Washington, USA. ACM Press. ISBN 1-59593-186-4.

Kennedy, J. and Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ray, T., Isaacs, A., and Smith, W. (2009). A memetic algorithm for dynamic multiobjective optimization. In *Multi-Objective Memetic Algorithms*, volume 171 of *Studies in Computational Intelligence*, pages 353–367. Springer Berlin / Heidelberg.

Talukder, A. K. A. and Kirley, M. (2008). A pareto following variation operator for evolutionary dynamic multi-objective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation 2008 (CEC 2008)*, Hong Kong, China. IEEE Press, Piscataway, NJ.

Veldhuizen, D. A. V. and Lamont, G. B. (1998). Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Veldhuizen, D. A. V. and Lamont, G. B. (2000). On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey. IEEE Service Center.

Yuping Wang, C. D. (2008). An evolutionary algorithm for dynamic multi-objective optimization. *Applied Mathematics and ComputationIn Press*.

Zeng, S., Chen, G., Zheng, L., Shi, H., de Garis, H., Ding, L., and Kang, L. (2006). A Dynamic Multi-Objective Evolutionary Algorithm Based on an Orthogonal Design. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 2588–2595, Vancouver, BC, Canada. IEEE.

Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., and Tsang, E. (2006). Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *The 4th Int. Conf. on Evolutionary Multi-Criterion Optimization*, volume 4403, pages 832–846. Springer.