Strength Pareto Particle Swarm Optimization and Hybrid EA-PSO for Multi-Objective Optimization

Ahmed Elhossini aelhossi@uoguelph.ca
School of Engineering, University of Guelph, Guelph, ON, N1G 2W1, Canada

Shawki Areibi sareibi@uoguelph.ca
School of Engineering, University of Guelph, Guelph, ON, N1G 2W1, Canada

Robert Dony rdony@uoguelph.ca
School of Engineering, University of Guelph, Guelph, ON, N1G 2W1, Canada

Abstract

This paper proposes an efficient particle swarm optimization (PSO) technique that can handle multi-objective optimization problems. It is based on the strength Pareto approach originally used in evolutionary algorithms (EA). The proposed modified particle swarm algorithm is used to build three hybrid EA-PSO algorithms to solve different multi-objective optimization problems. This algorithm and its hybrid forms are tested using seven benchmarks from the literature and the results are compared to the strength Pareto evolutionary algorithm (SPEA2) and a competitive multi-objective PSO using several metrics. The proposed algorithm shows a slower convergence, compared to the other algorithms, but requires less CPU time. Combining PSO and evolutionary algorithms leads to superior hybrid algorithms that outperform SPEA2, the competitive multi-objective PSO (MO-PSO), and the proposed strength Pareto PSO based on different metrics.

Keywords

Multi-objective optimization, particle swarm optimization, evolutionary algorithms, strength Pareto evolutionary algorithm.

1 Introduction

Particle swarm optimization (PSO) is a relatively new heuristic algorithm first introduced in Kennedy and Eberhart (1995). Similar to evolutionary algorithms (EA), PSO is a population based technique inspired by the swarm behavior of birds and insects during their search for food or new areas for settlement. PSO is a meta heuristic in which each individual, called a particle, flies in its own direction and velocity searching for a good point in the solution space. All particles communicate their information to direct the swarm toward good points in the search space. Each particle represents a specific solution. The particle position is updated according to its own particular history of good points, which is called the individual-best, history of subset of the swarm, neighborhood best, and the overall swarm history, the global-best. While PSO has good global search capabilities, some researchers find an advantage in decreasing the maximum velocity of swarm particles to exploit the local search capability of the algorithm (Boeringer

and Werner, 2004). In addition, PSO, similar to other meta-heuristic techniques, was originally developed to handle a single objective optimization problem.

Most practical (industrial/engineering) problems are multi-objective in nature and the solution of these problems is a set of points that give trade-offs for the different objectives. Introducing multi-objective optimization for PSO is inspired by the early research in evolutionary algorithms (MOEA) for similar problems (Mostaghim and Teich, 2003). The strength Pareto evolutionary algorithm (SPEA) developed by Zitzler and Thiele (1999), its modified versions, SPEA2 (Zitzler et al., 2001, 2002) and the nondominated sorting genetic algorithm (NSGA) (Srinivas and Deb, 1994; Deb et al., 2002a) are examples of such algorithms. These algorithms are shown to be efficient in the field of multi-criteria optimization and many researchers have investigated their use in different applications. The majority of these algorithms make use of the Pareto dominance concept to assign a single fitness value for each individual in the population. This is used to select a set of nondominant solutions called the Pareto front. The diversity of solutions in the Pareto front to cover different trade-offs of the problem objectives and the distance to the actual front (optimal solutions) are two main issues that should be handled carefully and are affected by the fitness assignment and Pareto front individual selection techniques. Using an external memory (archive) to store nondominant solutions found during the search process is a common approach to maintain the Pareto front.

Inspired by these ideas, many multi-objective PSO algorithms have been presented in the literature. Previous work (Li, 2003; Coello et al., 2004; Reyes-Sierra and Coello, 2005; Yin et al., 2007; Mostaghim and Teich, 2003) gives examples of these algorithms. The area of multi-objective PSO is attracting much research interest because of the simple computational model introduced by PSO and exploration capabilities of the algorithm. Although several approaches already used with evolutionary algorithms are applied to PSO, they still suffer from some drawbacks, which include the inability to avoid local fronts and the low diversity of solutions in the Pareto front (Yin et al., 2007).

While evolutionary algorithms, especially genetic algorithms (GA), have good global search capabilities, PSO, on the other hand, can be tuned to be a good local search algorithm as well (Boeringer and Werner, 2004). Combining the features of evolutionary algorithms (in the form of genetic algorithms) with PSO has attracted several researchers to build different forms of hybrids. Previous work (Robinson et al., 2002; Shi et al., 2003; Settles and Soule, 2005; Zhang et al., 2007) gives examples of these approaches. Although these previous authors covered different approaches for hybrid EA-PSO, their use on multi-objective optimization has not been fully exploited.

Handling multiple objectives in PSO and improving the search capabilities for both PSO and EA were the main motivations behind the work presented in this paper. The main contributions of our work can be summarized by the following points:

1. The strength Pareto approach (Zitzler et al., 2001, 2002) is used to handle multiple objectives in PSO. This approach is used to maintain the Pareto front, handle the diversity between solutions and assign fitness values to each particle according to its nondominance status.

2. The strength Pareto PSO is integrated with the SPEA2 algorithm in a single framework where both PSO and EA operators (crossover and mutation) manipulate the same population. A unified external archive is used to store the nondominant solutions for both algorithms. This allows the building of three hybrid forms of EA-PSO algorithms. The EA operators are modified to preserve the position

information required by the PSO algorithm. A modified nondominance comparison operator is used to select P_{ib} (individual-best) for each particle.

The remainder of the paper is organized as follows. Section 2 gives essential background related to this work. Section 3 introduces work on multi-objective optimization based on both evolutionary and swarm optimization. Section 4 introduces the proposed approach for multi-objective PSO and hybrid EA-PSO. The experimental setup and benchmarks used for testing the different proposed algorithms are discussed in Section 5. Results and analysis are presented in Section 6. Section 7 concludes the paper and proposes some interesting future work.

2 Background

Multi-objective optimization is playing an important role in the solution of many real world problems. Almost every design process requires the optimization of several objectives that are usually in conflict with each other. A good practical example of an engineering design problem where multi-objective optimization has been successfully employed is the design space exploration for embedded systems (Givargis et al., 2002; Lieverse et al., 2001; Pimentel et al., 2006; Elhossini et al., 2008). In this application, different conflicting objectives are required to be optimized such as: the system performance, area, power consumption, and flexibility. For example, increasing the system performance results in an increase in the system dynamic power consumption. An efficient heuristic search technique that is able to handle the different trade-offs among these objectives will be helpful in many real world applications. In these systems, multi-objective evolutionary algorithms were the main optimization technique used. The exploration properties of PSO combined with its abilities to perform local search makes it a promising technique for many applications, especially if combined with the current evolutionary algorithms. In the following sections, some basic definitions for multi-objective optimization and PSO are given.

2.1 Multi-Objective Optimization Problem (MOOP)

The multi-objective optimization problem is defined as follows:
Find the vector $\vec{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ that satisfies the set of constraints:

$$g_i(\vec{x}) \geq 0, i = 1, 2, \ldots, m \tag{1}$$

$$h_i(\vec{x}) = 0, i = 1, 2, \ldots, p \tag{2}$$

and optimizes a vector of objective functions

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is a vector of decision variables. Usually $f_1, f_2, \ldots, f_k$ are conflicting, which means satisfying f_1 can result in unsatisfying values for $f_2, \ldots, f_n$ (Coello et al., 2004).

2.1.1 Terms and Definitions

Pareto Dominance. A vector $\vec{a} = \{a_1, a_2, \ldots, a_k\}$ is said to dominate vector $\vec{b} = \{a_1, a_2, \ldots, a_k\}$, or $\vec{a} \preceq \vec{b}$, if $a_j \leq b_j$ for every $j \in \{1, 2, \ldots, k\}$, and $a_j < b_j$ for at least one value of j.

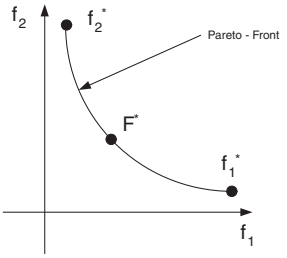


Figure 1: The concept of the Pareto front.

Pareto Optimality. Any point $\vec{x}^* \in \Omega$ (where Ω is a set of visible values of $\vec{x}$) is said to be optimal if for every $\vec{x} \in \Omega$

$$\vec{f}(\vec{x^*}) \preceq \vec{f}(\vec{x}) \tag{4}$$

Pareto Optimal Set. The Pareto optimal set for any multi-objective optimization problem is the set P^* of vectors $\vec{x} \in \Omega$ where there is no other vector $\vec{x'} \in \Omega$ that makes the inequality $\vec{f}(\vec{x'}) \preceq \vec{f}(\vec{x})$ true.

Pareto Front. The Pareto front of any multi-objective optimization problem is the set PF^* of vectors $\vec{f}(\vec{x})$ corresponding to each $\vec{x} \in P^*$. In other words, the Pareto front is the set of function vectors in the solution space that represents the trade-offs for the different objectives. In multi-objective optimization, it is necessary to find the Pareto optimal set of solutions, P^*, in the decision variables space that results in the Pareto front set, PF^*, in the objective space. The concept of the Pareto front is shown in Figure 1.

2.2 Particle Swarm Optimization (PSO)

In particle swarm optimization, a group of particles is formed where each particle represents a possible solution for the optimization problem. Each particle contains a decision variable vector $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ which is also considered the position vector for the particle in the search space. It also contains a velocity vector $\vec{v} = [v_1, v_2, \ldots, v_n]^T$ that represents the velocity of each variable. In the standard PSO proposed in Kennedy and Eberhart (1995) and Eberhart and Kennedy (1995), both the position $\vec{x}$ and velocity $\vec{v}$ vectors are updated according to Equations (5) and (6), respectively.

$$\begin{aligned} \vec{v}_{m+1} &= \omega \vec{v}_m + \Phi_1 \vec{r_1}(P_{\text{ib}} - \vec{x}_m) + \Phi_2 \vec{r_1}(P_{\text{gb}} - \vec{x}_m) \\ \vec{x}_{m+1} &= \vec{x}_m + \vec{v}_{m+1} \end{aligned} \tag{5}$$

$$\begin{aligned} \vec{v}_{\min} \preceq \vec{v}_{m+1} \preceq \vec{v}_{\max} \\ \vec{x}_{\min} \preceq \vec{x}_{m+1} \preceq \vec{x}_{\max} \end{aligned} \tag{6}$$

In Equation (5), the individual-best vector, P_{ib}, is the best solution vector ever found by the particle. In a single objective optimization problem, this vector is updated if the new position results in a better value for the objective function. The value of

global-best (neighborhood-best) vector, P_{gb}, depends on the chosen topology of the particle swarm (Bratton and Kennedy, 2007) which defines the communication strategy among particles.

In the *gbest* (global-best) topology, which is initially adopted by the standard PSO, the global-best vector contains the best solution ever found by the entire swarm. Accordingly, all particles communicate their information with each other to update this vector. The *gbest* topology provides a quick solution but could suffer from premature convergence, which results in low quality solutions. On the other hand, in multi-objective optimization, there is no single solution for the problem as discussed in Section 2.1. For this reason, the *gbest* topology needs to be modified for multi-objective optimization so that the global-best vector is linked to one solution in the Pareto front (Coello et al., 2004).

The *lbest* topology is another technique used to describe any swarm topology in which there is no global communication (Bratton and Kennedy, 2007). In this case, the global-best vector P_{gb} is updated by the best solution in a subset of the swarm that ranges from one particle to many. The global-best vector holds the best solution in this subset, which is sometimes referred to as the particle neighborhood and the vector in this case is called the neighborhood-best. The *lbest* topology has the advantage of a slower convergence rate compared to the *gbest* topology, which leads to finding a global optimum with higher probability. This topology has been recently used to overcome premature convergence of the *gbest* topology (Bratton and Kennedy, 2007).

Three constants control the contribution of each of the three components of the velocity update in Equation (5). The constant ω, or inertia weight, controls the contribution of the previous velocity. The constant Φ_1, or the self-knowledge factor and Φ_2, or the social-knowledge factor, specify the relative importance of the individual-best and global-best solutions, respectively, in the velocity update.

These constants control the behavior of the algorithm and careful choice of these constants should be considered. The variables r_1, r_2 are randomly chosen between 0 and 1 and are different for each direction. These two variables ensure the stochastic behavior of the algorithm.

The vector $\vec{v}_{\max}$ defines the maximum velocity in each direction and is used to limit the particle in the problem space. If the updated velocity vector exceeds $\vec{v}_{\max}$, it is then limited to this value. The choice of $\vec{v}_{\max}$ is problem dependent and it can greatly affect the performance of the optimizer. Controlling the velocity vector using the inertia weight is more preferable. The vector $\vec{x}_{\max}$ defines the range of values for the decision variables. Particles can be allowed to fly outside the search space without limiting the velocity. In this case, the vectors $\vec{v}_{\max}$ and $\vec{x}_{\max}$ are not used (Bratton and Kennedy, 2007).

Another variation for the PSO update equation is introduced in Clerc and Kennedy (2002) where constriction is applied to further control the local and global search capabilities of the PSO. In this equation, a new parameter χ is introduced and is calculated as a function of Φ_1 and Φ_2 as shown by Equation (7):

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \varphi = \Phi_1 + \Phi_2 \tag{7}$$

Using a constant value for $\varphi = 4.1$, which results in $\chi = 0.72984$ and $\Phi_1 = \Phi_2 = 2.05$, is standard to ensure convergence with the updated PSO Equation (8) (Bratton and Kennedy, 2007):

$$\vec{v}_{m+1} = \chi(\vec{v}_m + \Phi_1 r_1(P_{\text{ib}} - \vec{x}_m) + \Phi_2 r_1(P_{\text{gb}} - \vec{x}_m)) \tag{8}$$

In our work a *lbest* topology is adopted where each particle is only linked with four particles of the swarm. The neighborhood-best vector is updated to the best solution in these four particles depending on the Pareto optimality criteria as will be explained in Section 4.3.

3 Related Work

Particle swarm optimization and its multi-objective variant have attracted much research interest for several applications. Most of the research in this area is inspired by the research carried out using evolutionary algorithms. Some effort has been geared toward the hybrid EA-PSO technique, but not multi-objective optimization. In the following subsections, related work in multi-objective PSO and hybrid EA-PSO is presented.

3.1 Multi-Objective Particle Swarm Optimization

A sigma method was introduced in Mostaghim and Teich (2003) to select the individual-best vector, P_{ib}, for each particle to improve the convergence and diversity of the final Pareto front in multi-objective PSO. In this method, a value σ is assigned to each point with coordinates $(f_{1,i}, f_{2,j})$ so that all the points which are on the line $f_2 = a \times f_1$ have the same value of σ. The value of σ for each particle is then used to rank the particles to select the individual-best. However, the use of the sigma approach results in a premature convergence which affects the quality of the final solution.

Coello and Lechuga proposed another approach for multi-objective PSO in Coello et al. (2004) where the space is divided to a finite number of hypercubes. A fitness value is assigned to each hypercube, depending on the number of nondominant particles that lie in it. The hypercube approach is used to update the contents of an external archive that stores the nondominant solutions. A mutation operator is also employed to prevent premature convergence. The authors compared their results with three MOEA algorithms and the performance of their approach appeared to outperform those MOEA implementations. Although this approach showed good performance for some benchmarks, it suffers from a number of drawbacks. First, the diversity of solutions in the Pareto front depends heavily on the number of hypercubes used. This requires tuning itself, and therefore can result in inadequate distribution of solutions in the Pareto front. Second, the algorithm was not able to avoid local fronts in some benchmarks.

In Ho et al. (2005) a PSO based algorithm for finding the Pareto solutions of multi-objective design problems is proposed. A novel formula for updating the particles' velocity and position to enhance the global searching ability of PSO is given. The authors introduce a fitness assignment schema to rank different individuals based on the concept of Pareto optimality, combined with the use of two external archives to handle the Pareto front and the selection of both global and local bests. However, the ranking schema used in Ho et al. (2005) is not efficient to ensure the diversity of solutions in the Pareto front.

The nondominant sorting approach used in NSGA (Srinivas and Deb, 1994; Deb et al., 2002a) is used to build a multi-objective PSO in the work proposed in Li (2003). The authors compared their approach to the original NSGA-II algorithm and their results show that multi-objective PSO outperforms NSGA-II in many cases. The nondominant sorting approach is employed again in Mahfouf et al. (2004) to build a multi-objective PSO. In this work the velocity update in Equation (5) is modified such that the effect of the second and third terms of the equation is reduced in later iterations to force a local

search close to the end of the optimization process. A weighted aggregation approach is used to construct an evaluation function F, which is used for the selection of both P_{ib} and P_{gb}. This approach is compared to SPEA2 and NSGA-II evolutionary algorithms and showed some competitive performance, although the diversity of the final Pareto front is relatively low and is affected by the selection of weighted objective values of function F.

Multi-objective optimization is handled in Gill et al. (2006) by using Pareto ranking and a weighted objective function (WOF). In this algorithm, all the particles are ranked according to their Pareto optimality. All solutions ranked "1" are considered the Pareto front. The WOF is used to select the median of the Pareto front which is used as the global-best for the complete swarm. However, the weights for the WOF are problem dependant, difficult to estimate, and affect the diversity of solutions in the resulting Pareto front.

Hypercubes are used again in Kitamura et al. (2006) to support multi-objective optimization in PSO. The objective space is divided into a specific number of hypercubes and the global-best vector is selected using this approach. One hypercube is selected with a probability inversely proportional to the number of nondominant solutions in the hypercube. The higher the number of nondominant solutions, the lower the probability of selection of that hypercube. Within the selected hypercube, P_{gb} is selected randomly. This approach is applied to energy management systems and is not tested for numerical problems. Therefore, its performance cannot be verified. The number of hypercubes used remains a problem that affects the diversity of solutions in the Pareto front.

The research in multi-objective PSO incorporates techniques used in multi-objective evolutionary algorithms. The use of an external archive to store nondominant solutions found during the search process is a common approach to maintain the Pareto front. Different approaches are used to update the contents of the archive. Examples are the hypercube approach, nondominant ranking, and the nondominant sorting approach. The selection of both P_{gb} and P_{ib} is usually made by random selection with the probability depending on the distribution of solutions in the Pareto front. The diversity of solutions in the Pareto front and the ability to avoid local fronts are two of the main problems facing these techniques. Adopting a different approach for handling multiple objectives in PSO that increases the diversity of solutions in the Pareto front is one of the main motivations behind our work.

3.2 Hybrid EA-PSO

Using a hybrid form of evolutionary algorithms and particle swarm optimization is gaining some attention. However, it has not been fully used for multi-objective optimization. PSO has good search properties to optimize continuous functions and has good local search capabilities. EA, on the other hand, has been shown to be useful in the optimization of discontinuous problems and is good in exploring the solution space (Robinson et al., 2002; Boeringer and Werner, 2004). Combining EA and PSO should result in a hybrid algorithm with the advantages of both types of heuristic algorithms. Some work on hybrid algorithms is presented in this section.

In Robinson et al. (2002) a hybrid EA-PSO approach is introduced for the optimization of a profiled corrugated horn antenna. This work introduces two forms of hybrid algorithms: GA-PSO and PSO-GA. In the first form, a GA is first run until it starts to converge. Then, PSO is allowed to run on the final population found by the GA. In the second form, PSO is allowed to run first, followed by GA. The application targeted by

Robinson et al. (2002) has only a single objective and the results of the two hybrid forms outperform GA or PSO. They also show that GA followed by PSO gives better results than the other proposed algorithms.

In Shi et al. (2003) a variable population size GA (VPGA) algorithm is introduced and a hybrid form of this algorithm with PSO is implemented. In this hybrid form, both GA and PSO run in two separate populations. After each iteration, both systems exchange the best individuals. Six single objective benchmarks are used to test the three algorithms: PSO alone, VPGA alone, and their hybrid form (PGBEHE). The results show that the hybrid form is more efficient than either PSO or GA running alone. The PGBEHE is only tested with single-objective problems and its use with multi-objective optimization requires further investigation.

A hybrid GA-PSO algorithm for recurrent network design is introduced in Juang (2004). In this algorithm, individuals in the new generation are created by traditional GA operators (crossover and mutation) and then updated using the PSO update operation. This allows PSO to enhance the properties of the generation before selecting different individuals for breeding using the GA operators.

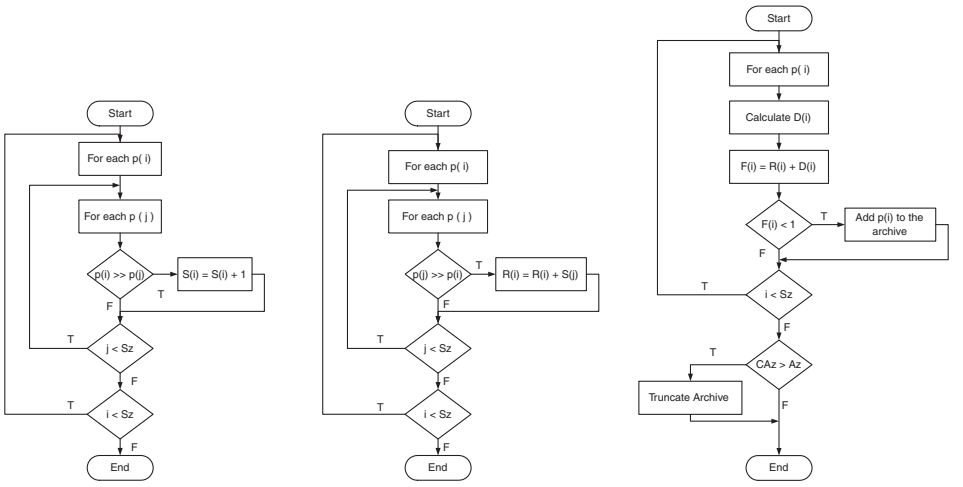
The breeding swarms introduced in Settles and Soule (2005) is yet another form of hybrid EA-PSO algorithm. In this work, the velocity update of the PSO and the crossover and mutation operators of GA are combined to breed new solutions from the previous generation. Parameters for both GA and PSO are set so that GA performs a global search and PSO performs a local search. The authors presented several tests for a variety of single objective problems and the results show that their developed hybrid outperforms both GA and PSO.

A novel hybrid evolutionary algorithm based on evolutionary programming (EP) and PSO, named EPPSO, is presented in Zhang et al. (2007) for multi-objective optimization. The characteristics of the solution of each particle are modified to become more suitable to the objective function before producing offspring. Both EP and PSO are used to update all individuals before breeding. The mutation operator in EP effectively restrains the premature convergence of PSO caused by the fast convergence, while the memory and collaboration characteristics of PSO help EP to fine-tune the search. Furthermore, the adaptive grid algorithm is applied to the EPPSO to maintain the diversity of the Pareto front. The effectiveness of the proposed algorithm was demonstrated with numerical test functions. The results show that the hybrid form demonstrates the best Pareto front, the lowest computation complexity, and most improvement in the convergence efficiency compared to EA, or PSO running alone. However, this algorithm is tested only on single objective problems and further testing and investigation is required for multi-criteria optimization.

Although all forms of hybrid GA-PSO algorithms outperform the original algorithms (GA or PSO alone), as found in Robinson et al. (2002); Shi et al. (2003); Settles and Soule (2005); Juang (2004) and others, their work did not focus on multi-objective optimization, so the use of PSO for local search with GA has never been investigated to the best of our knowledge.

4 Proposed Approach

In this paper, the fitness assignment technique proposed in the SPEA2 (Zitzler et al., 2001, 2002) is used to build a multi-objective based PSO. An external set (archive $\overline{P_t}$) is used to keep a record of the nondominated solutions in each iteration. A strength value is assigned to each individual (particle) which is equal to the number of solutions it



(a) Strength Calculation (b) Raw Fitness Calculation (c) Fitness Assignment and Archive Update

Figure 2: Strength Pareto algorithms (Sz = Swarm Size, Az = Archive Size, CAz = Current Archive Size).

dominates in the current population using Equation (9):

$$S(i) = |\{j | j \in P_t + \overline{P_t} \wedge i \succ j\}| \tag{9}$$

where $S(i)$ is the strength of individual i, P_t is the current population, and $\overline{P_t}$ is the external archive. The strength calculation algorithm is shown in Figure 2(a).

A raw fitness value is then assigned to each individual (particle) which is equal to the sum of the strength of individuals dominating it using Equation (10):

$$R(i) = \sum_{j \in P_t + \overline{P_t}, j \succ i} S(j) \tag{10}$$

where $R(i)$ is the raw fitness value for individual i. A raw fitness of zero indicates that this individual is not dominated by any other solution. The raw fitness calculation algorithm is shown in Figure 2(b).

To ensure the diversity between solutions in the Pareto front, the kth nearest neighbor method is used to assign another amount to each individual fitness, $D(i)$. The final strength Pareto fitness of each individual is therefore set to $F(i) = R(i) + D(i)$. The strength Pareto fitness assignment algorithm has complexity of $O(M^2 \cdot \log M)$ where M is the sum of the population and the archive sizes.

All the individuals are sorted using the calculated strength Pareto fitness and the external archive is updated with all the individuals of fitness less than one. If the number of selected individuals is less than the archive size, dominated solutions with the lower strength Pareto fitness are added to the archive. When the number of selected individuals is greater than the archive size, an archive truncation mechanism is used to iteratively reduce the number of individuals. The individuals with the minimum

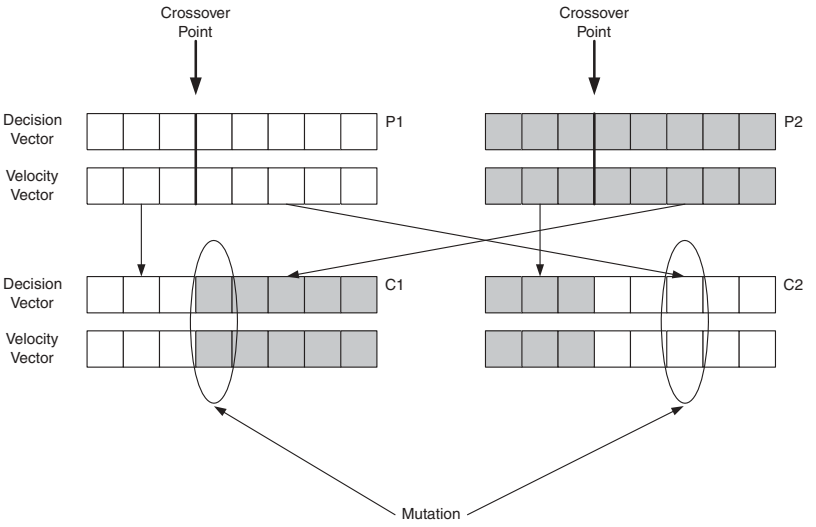


Figure 3: Particle data structure and modified crossover and mutation operators.

distances to each other are removed one at a time to ensure diversity in solutions. A simplified flowchart of SPEA2 fitness assignment and archive update algorithm is shown in Figure 2(c).

The proposed approach is used to maintain the archive used for the selection of the neighborhood-best vector and for breeding in the EA. This allows both EA and PSO to operate simultaneously with the same fitness assignment scheme.

4.1 Particle/Individual Data Structure

The data structure of the particle/individual in the swarm/population is designed to support both PSO and EA operations as shown in Figure 3. Each particle contains a decision vector or chromosome that holds several genes representing a solution that is common for both the EA and PSO algorithms. For each particle, an individual-best vector is also stored. It holds the best solution ever found by the particle. For PSO, a velocity vector is also required to hold the velocity for each variable in the decision vector. Both the EA and PSO operators manipulate the three vectors. Each particle contains a pointer to its own neighborhood-best in the population, that is, its own guide. The pointer is usually linked to a particle in the external archive selected at each iteration. There are other variables associated with each individual representing the maximum velocity and maximum value for each variable, that are fixed throughout the population.

4.2 Modified EA Operators

The crossover and mutation operators are modified to preserve the information required by PSO. The modification is set to affect both the decision and velocity vectors as shown in Figure 3. Hence, each variable in the decision vector (chromosome) is transferred from

one generation to the next with its velocity information transferred along with it. This enables both PSO and EA to operate simultaneously on the same population without further changes. For the individual best vector, each child inherits one of the two vectors of the parents without modification.

4.3 Global-Best Selection

For each particle in the swarm, the global-best is selected from the external archive using tournament selection with a size of four. The particle with lowest strength Pareto fitness is selected as the neighborhood-best.

4.4 Individual-Best Selection and the Modified Pareto Selection

Each particle keeps a record of its own best solution ever found (individual-best). Following the update of the particle position, the current solution is compared to the individual-best solution according to the different objectives. A modified Pareto dominant approach is used to update the individual-best vector as follows: (i) if the new solution dominates the individual-best using the normal Pareto dominant criteria, the individual-best is updated with the new solution; (ii) if the new solution and the individual-best do not dominate each other, the one with a better value for the majority of objective functions is selected; (iii) when the two vectors are equal in this criterion, one of them is selected randomly. For two objective problems this approach will tend to perform similar to normal Pareto dominant testing. However, for three or more objectives problems, this will help to direct the search toward solutions with better improvements in all dimensions.

4.5 Hybrid Algorithms

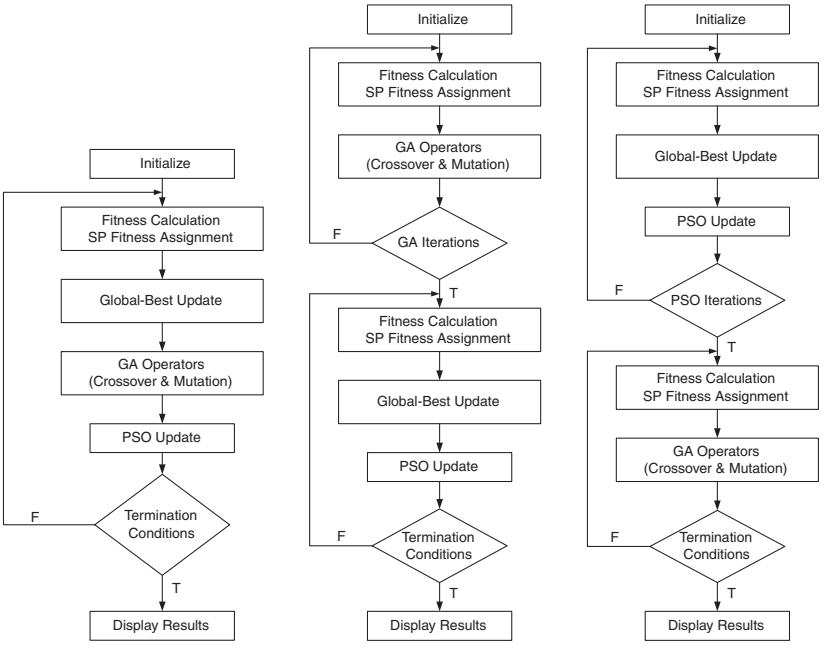
Using the strength Pareto approach described earlier, different hybrid forms of EA and PSO are designed and compared. Each algorithm depends on a different organization of EA and PSO operators. In this section, three hybrid forms are introduced. In all of these algorithms, strength Pareto fitness assignment is used to maintain an external archive. Particles in the archive are used for EA breeding and for PSO global-best selection for each particle.

4.5.1 EAPS1

In this hybrid algorithm, the PSO update operator is inserted into the EA main loop. As discussed earlier, the crossover and mutation operators preserve the information required by PSO to perform its update. This allows each particle to explore its boundaries with faster steps in the hope of improving the performance of the EA local search as shown in Figure 4(a).

4.5.2 EAPS2

In this hybrid algorithm, the EA evolves solutions for a finite number of iterations or until a specific criterion is reached. PSO is then used on the final population of the EA as its initial population. PSO is allowed to run for several iterations, as shown in Figure 4(b). In all the tests conducted in this paper, both EA and PSO are allowed to run for the same number of iterations. EA is generally used to perform global search at the



(a) EAPS1 (b) EAPS2 (c) EAPS3

Figure 4: Hybrid evolutionary and particle swarm optimization algorithms.

beginning of the search process while PSO is used to perform local search at the end of the search process.

4.5.3 EAPS3

In the third hybrid algorithm, the order of PSO and EA is reversed. PSO operates normally for a finite number of iterations, or until a terminating criterion is reached. EA then starts operating on the final population of the PSO as its initial population and is allowed to run for another finite number of iterations, as shown in Figure 4(c).

5 Experimental Setup

5.1 Benchmarks

Seven benchmarks from the literature are used to evaluate the functionality of the proposed algorithms. Each benchmark has two or more objective functions to be optimized. These benchmarks are chosen because they possess different features and characteristics that are important to test the effectiveness of the developed algorithms to exploit the Pareto front. In all of these benchmarks, the objective functions are to be minimized.

5.1.1 Benchmark 1

This benchmark is proposed in Kursawe (1991). The global front of the benchmark is not continuous and is described by Equations (11) and (12). The benchmark is selected because it has a nonconvex Pareto optimal front that allows testing different algorithms

against such types of multi-objective landscapes.

$$f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(-10 \exp \left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \tag{11}$$

$$f_2(\vec{x}) = \sum_{i=1}^{n} (|x_i|^{0.8} + 5 \sin(x_i)^3) \tag{12}$$

where $-5 \leq x_1, x_2, x_3 \leq 5$

5.1.2 Benchmark 2

The second benchmark was proposed in Deb (1999) and is described by Equations (13) and (14). This benchmark has a convex Pareto front with one global front at $x_2 = 0$. The benchmark is used to test the ability of the algorithm to find a well distributed front for convex shaped Pareto problems.

$$f_1(x_1, x_2) = x_1 \tag{13}$$

$$f_2(x_1, x_2) = g(x_1, x_2) \cdot h(x_1, x_2) \tag{14}$$

where

$$g(x_1, x_2) = 11 + x_2^2 - 10 \cdot \cos(2\pi x_2),$$

$$h(x_1, x_2) = \begin{cases} 1 - \sqrt{\dfrac{f_1(x_1, x_2)}{g(x_1, x_2)}}, & \text{if } f_1(x_1, x_2) \leq g(x_1, x_2) \\ 0, & \text{otherwise} \end{cases}$$

and $0 \leq x_1 \leq 1$, $-30 \leq x_2 \leq 30$

5.1.3 Benchmark 3

This benchmark was also proposed in Deb (1999) and is described by Equations (15) and (16). The benchmark has one local front at $x_2 = 0.6$ and one global front at $x_2 = 0.2$, which makes it a good benchmark to test the ability of different algorithms to avoid local fronts.

$$f_1(x_1, x_2) = x_1 \tag{15}$$

$$f_2(x_1, x_2) = \frac{g(x_2)}{x_2} \tag{16}$$

where

$$g(x_2) = 2.0 - \exp \left\{ - \left(\frac{x_2 - 0.2}{0.004} \right)^2 \right\} - \exp \left\{ -0.8 \cdot \left(\frac{x_2 - 0.6}{0.4} \right)^2 \right\}$$

and $0.1 \leq x_1 \leq 1.0$, $0.1 \leq x_2 \leq 1.0$

5.1.4 Benchmark 4

This benchmark is a modification of benchmark 3 that adds another local front at $x_2 = 0.8$. The $g(x_2)$ component in Equation (16) is modified as shown in Equation (17). Increasing the number of local fronts in this benchmark increases the challenge for the algorithm to find the global front. This allows more testing of the different proposed algorithms against premature convergence.

$$g(x_2) = 2.0 - \exp\left\{-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right\} - \exp\left\{-0.8 \cdot \left(\frac{x_2 - 0.6}{0.4}\right)^2\right\} - \exp\left\{-0.3 \cdot \left(\frac{x_2 - 0.8}{0.002}\right)^2\right\} \tag{17}$$

and $0.1 \leq x_1 \leq 1.0, 0.1 \leq x_2 \leq 1.0$

5.1.5 Benchmark 5

The SYMPPART problem was proposed recently in Rudolph et al. (2007) for testing multi-objective algorithms and was used in Huang et al. (2007) for the evaluation of different algorithms. This test problem is a scalable, two objective problem. The problem is represented by Equation (18).

$$\begin{aligned}
f_1(x) &= (x_1' + a - t_1 c_2)^2 + (x_2' - t_2 b)^2 + \ldots \\
&\quad + (x_{D-1}' + a - t_1 c_2)^2 + (x_D' - t_2 b)^2 \\
f_2(x) &= (x_1' - a - t_1 c_2)^2 + (x_2' - t_2 b)^2 + \ldots \\
&\quad + (x_{D-1}' - a - t_1 c_2)^2 + (x_D' - t_2 b)^2 \\
X' &= \begin{pmatrix} \cos\omega & -\sin\omega & 0 & 0 & \ldots \\ \sin\omega & \cos\omega & 0 & 0 & \ldots \\ 0 & 0 & \cos\omega & -\sin\omega & \ldots \\ 0 & 0 & \sin\omega & \cos\omega & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix} X \\
t_1 &= sgn(x_1') \times \left\lceil \frac{|x_1'| - \frac{c_2}{2}|}{c_2} \right\rceil \\
t_2 &= sgn(x_2') \times \left\lceil \frac{|x_1'| - \frac{b}{2}|}{b} \right\rceil \\
x &\in [-20, 20]^D
\end{aligned} \tag{18}$$

In these equations, the constant values are $a = 1, b = 10, c = 8, c_2 = c + 2a = 10$. The constant D represents the dimension, or the number of variables, and it was set to 30 variables in our tests.

5.1.6 Benchmarks 6 and 7

The DLTZ2 benchmark introduces a more complex problem with a scalable number of objectives and variables. This benchmark was introduced in Deb et al. (2002b) and

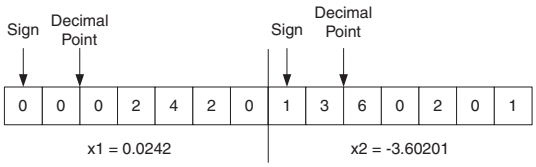


Figure 5: Example of the numerical representation used in different benchmarks variables.

modified in Huang et al. (2007) to introduce more scalability to the problems. The extended, shifted version of the problem (S_DTLZ2) was tested in this work with two versions where the number of objectives was set to three objectives (benchmark 6) in the first version and five objectives (benchmark 7) in the second one (S_DTLZ2_3 and S_DTLZ2_5). The scaling parameters introduced in Huang et al. (2007) were used to scale the two versions with 30 different variables.

5.2 Problem Representation

Problem representation plays an important role in evolutionary algorithms (Hiroyasu et al., 2003). Good representation of the problem results in a better performance. In this experiment, each variable in the particle decision vector represents one digit in the benchmark problem variable or the sign of the variable as shown in Figure 5. A precision of five digits is used in all the tests and the sign digit is used when the problem variable has a negative bound. This representation may not be efficient for PSO, but it enables the integration of PSO and EA. The chromosome length depends on the number of variables in the problem and the bounds of each variable.

5.3 Performance Metrics

To evaluate the performance of the developed algorithms with each benchmark, several performance metrics are evaluated. These metrics measure different aspects in multi-objective optimizations, including the following.

1. The **distance** between the computed Pareto front and the actual Pareto front of the benchmark. This distance should be minimized and can be expressed by the generational distance (GD) shown in Equation (19).

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{19}$$

   where n is the number of solutions in the nondominated solution set and d_i is the Euclidean distance in the objective space between each solution and the nearest point in the actual front (Coello et al., 2004).

2. The **spacing** and distribution of the solutions in the Pareto front. The spacing (SP) measure introduced by Equation (20) is used to give a measure of the distribution

Table 1: Parameters for PSO.

Parameter	Tuning range	Value
ω Momentum (inertia) factor	0.05–0.5	0.1
Φ_1 Self-knowledge factor	0.4–3	1.6
Φ_2 Social knowledge factor	0.4–3	1.6
$v_{\max}$ Maximum allowed velocity	0.5–5	3
T_{pso} Tournament size for P_{gb} selection	—	4

spread in the nondominated front.

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{20}$$

where $d_i = min_j |f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|$, $i, j = 1, \ldots, n$, $\bar{d}$ is the mean of all d_i, and n is the number of solutions in the nondominant set. A smaller value of this metric indicates a good distribution of solution in the nondominated set as the solutions will be equally spaced (Coello et al., 2004).

3. The ratio of actual nondominant solutions in the final Pareto front which forms the **error** metric (ER) as shown by Equation (21).

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{21}$$

where e_i equals 0 if solution i is a member of the actual front, and equals 1 otherwise. A value of 0 indicates a perfect match with the actual front (Coello et al., 2004).

4. Computational time, which is defined as the actual runtime required by the algorithm to complete a fixed number of iterations. In this work the number of iterations is set to 200.

5.4 Parameter Tuning

Running any meta-heuristic requires the setting of a number of parameters. However, finding settings that work well on a specific problem is not a trivial task. Poor settings lead to inferior results; good settings require time-consuming trials to find. Some of the methods used to tune parameters include: (1) using values recommended in the literature by other researchers working on similar problems; (2) using two different sets of test problems, one for parameter tuning and one for reporting the final results; (3) adaptively changing the parameters in response to some statistical measures that represent the search status. Initial tuning is performed to find effective values for most of the parameters listed in Tables 1 and 2.

Each parameter under investigation is varied for the range of valid values while all the other parameters are kept constant. The GD, SP, and computation time measures are used to select appropriate values for each parameter. In the following sections, parameters used for PSO and EA are presented, followed by a description of the method used to tune these parameters.

Table 2: Parameters for EA.

Parameter	Tuning range	Value
P_{co} Crossover rate	0.5–1	0.95
P_{mu} Mutation rate	0–0.1	0.05
P_{sl} Selection probability	—	1
T_{ea} Tournament size for evolutionary operation	2–8	2
C_p Crossover points	1–4	2

Table 3: Common parameters used for EA and PSO.

Parameter	Value
P_t population size	100
$\overline{P}_t$ archive size	100
I Number of iterations	200

5.4.1 PSO Parameters

The list of parameters for PSO operations is shown in Table 1. The momentum (inertia) factor ω, the self-knowledge factor Φ_1, and the social-knowledge factor Φ_2 are the PSO update parameters used in Equation (5). It is suggested in the literature that these two parameters be assigned the same value (Kennedy and Eberhart, 1995; Bratton and Kennedy, 2007). The maximum allowed velocity v_{max}, controls the maximum velocity in which the particle flies. This value controls the local/global search trade-off capabilities of the PSO algorithm. In the selection of the global-best for each particle, tournament selection is used with size T_{pso}. For each parameter, the range of values used to tune this parameter and the value used after tuning are shown in Table 1.

5.4.2 EA Parameters

The list of parameters and corresponding values for EA are shown in Table 2. Crossover rate, P_{co}, mutation rate, P_{mu}, and selection probability, P_{sl}, are the evolutionary operation parameters. Tournament selection is also used for the selection operation in the EA with size T_{ea}. The crossover points, C_p, define the number of points at which crossover is performed.

5.4.3 Common Parameters

In all tests performed, the population size P_t and the archive size $\overline{P}_t$ are kept fixed at a value of 100 each. The archive size defines the number of points in the final Pareto front. The number of iterations I is also kept constant at 200. The values of P_t, $\overline{P}_t$, and I are chosen to keep the number of fitness calculations to a constant value throughout the test. However, increasing the archive size in general increases the number of points in the Pareto front. This results in a trade-off between the solution quality and computation time. A list of these parameters is shown in Table 3.

5.4.4 Static Parameter Tuning

Initial testing for different values of each parameter is performed. This initial testing specified the range of values that can be used for best performance for different algorithms. This range of values is later tuned statistically as shown in the following two examples.

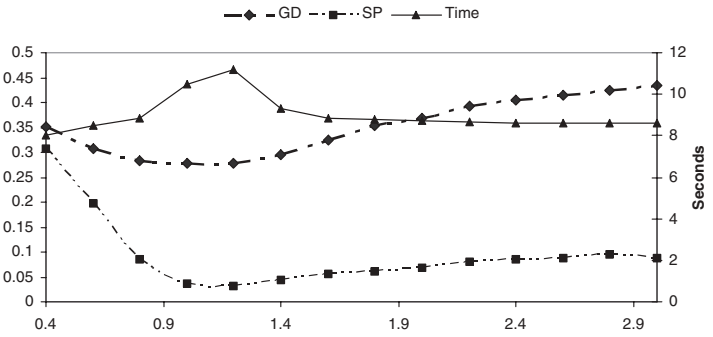


Figure 6: Tuning Φ_1 and Φ_2.

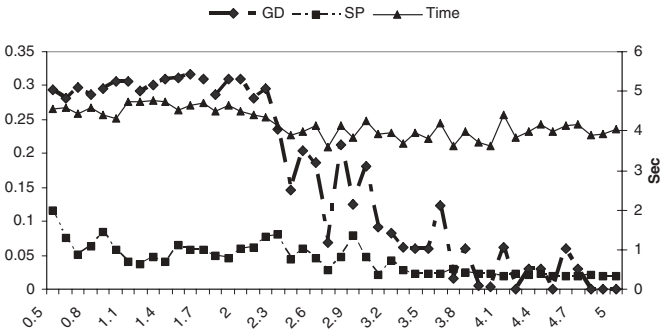


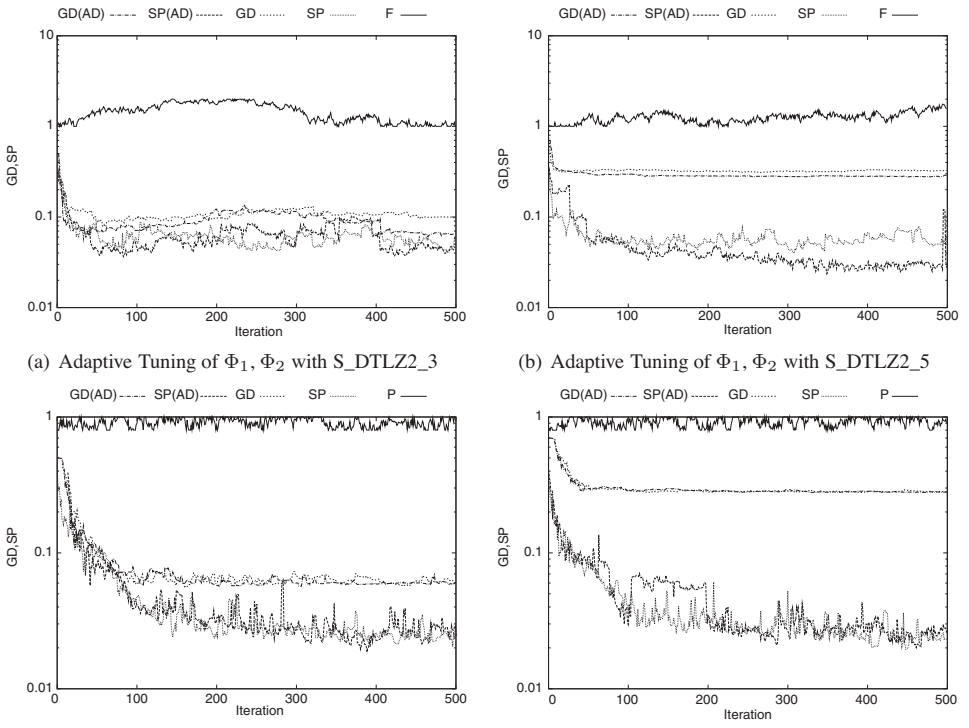
Figure 7: Tuning $V_{\max}$.

An example for parameter tuning is shown in Figure 6, which demonstrates the effect of Φ_1 and Φ_2 using the benchmark S_DTLZ2_3. Each point in the graph represents the average of 10 independent runs for each value. Values between 1 and 1.6 result in the best performance using the three different measures.

Another example is shown in Figure 7, which shows the effect of $V_{\max}$ on the second benchmark. Each point represents an average of 10 independent runs. As shown, any value greater than 3 is considered to give the best performance. The values selected by manual tuning depends on the problem being solved and may vary depending on the complexity of the problem.

5.4.5 Adaptive Parameter Tuning

The goal of using adaptive parameters (Younes et al., 2007) is twofold. First, adaptation controls population diversity and hence balances exploration and exploitation. Second, adaptation reduces the effort spent on parameter tuning and makes it more systematic and meaningful. Tuning the parameter values during the search process allows the algorithm to adapt to the given problem, and this may result in better quality solutions. In this section, we present a simple methodology for adaptive parameter tuning and introduce some initial results for the effect of adaptive tuning on the quality of the solution.



(a) Adaptive Tuning of Φ_1, Φ_2 with S_DTLZ2_3

(b) Adaptive Tuning of Φ_1, Φ_2 with S_DTLZ2_5

(c) Adaptive Tuning of Crossover Rate with S_DTLZ2_3

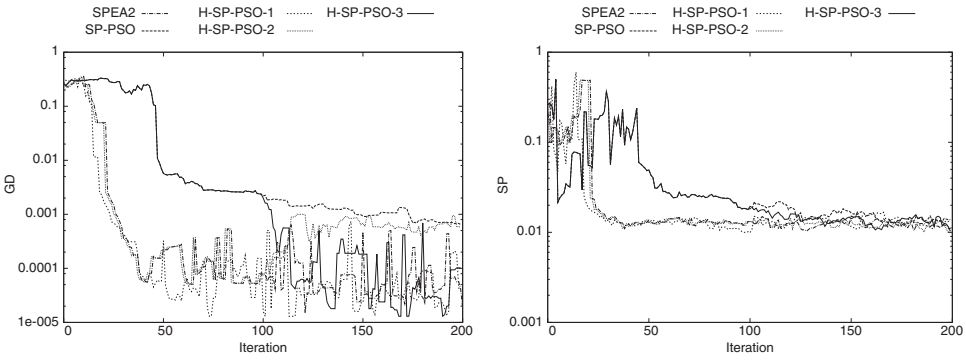
(d) Adaptive Tuning of Crossover Rate with S_DTLZ2_5

Figure 8: Adaptive tuning, logarithmic scale.

In the proposed approach, each parameter subject to tuning is updated with a randomly small amount within a specific range as shown in Equation (22). The tuned parameter, p, can have the range of values between p_+ and p_-, r is a randomly selected number between α_+ and α_-, and t is the current iteration number.

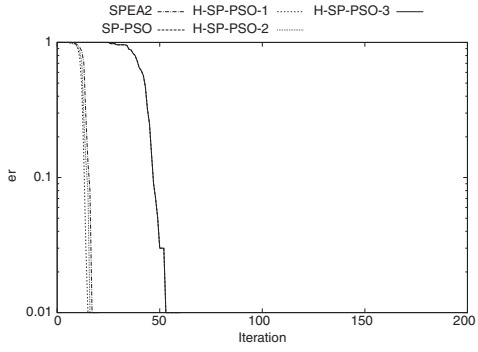
$$\begin{aligned} p_t &= p_{t-1} + r \\ r &\in [\alpha_-, \alpha_+] \\ p_t &\in [p_-, p_+] \end{aligned} \tag{22}$$

The modified value of the tuned parameter is accepted only if it results in an improvement in the population properties measured by the spacing measure (SP) introduced in Section 5.3. Figure 8 presents initial results of the proposed adaptive approach. Two benchmarks are used to test the adaptive approach: S_DTLZ2_3 and S_DTLZ2_5. For the SP-PSO we vary the values of Φ_1, Φ_2 while fixing all the other parameters and setting $p_- = 1$ and $p_+ = 2$. The results are shown in Figures 8(a) and 8(b). For each test, the SP-PSO algorithm is allowed to run for 500 iterations while the change in the GD and SP metrics is observed. Each test is repeated while the adaptive tuning is turned on and off. As can be seen, adaptive tuning results in an improvement in both the GD and SP metrics for both problems. The same set of tests was repeated for the SPEA2



(a) Generation Distance (GD) - Logarithmic Scale

(b) Spacing (SP) - Logarithmic Scale



(c) Error (ER) - Logarithmic Scale

Figure 9: Convergence of the different metrics for benchmark 3.

algorithm while tuning the crossover rate. These results are shown in Figures 8(c) and 8(d) with $p_- = 0.8$, $p_+ = 1$. As can be seen, the effect of the adaptive parameter tuning on SPEA2 is not significant.

The results presented in the section show that adaptive tuning has an impact on the SP-PSO algorithm. This effect is very problem dependent. Parameter tuning on the SPEA2 algorithm was less effective as it appears that the method is robust to the choice of parameters.

5.5 Statistical Analysis

For each test, 30 independent runs were performed. The average and the standard deviation of the 30 runs is reported and used to perform the statistical analysis. Two types of statistical methods are used to report the results of all tests.

1. First, confidence intervals were calculated using a confidence factor of 95%. This test was performed to provide the range where the actual mean may reside.

2. Second, the Student's t-test was used to check the significance of the comparison between averages.

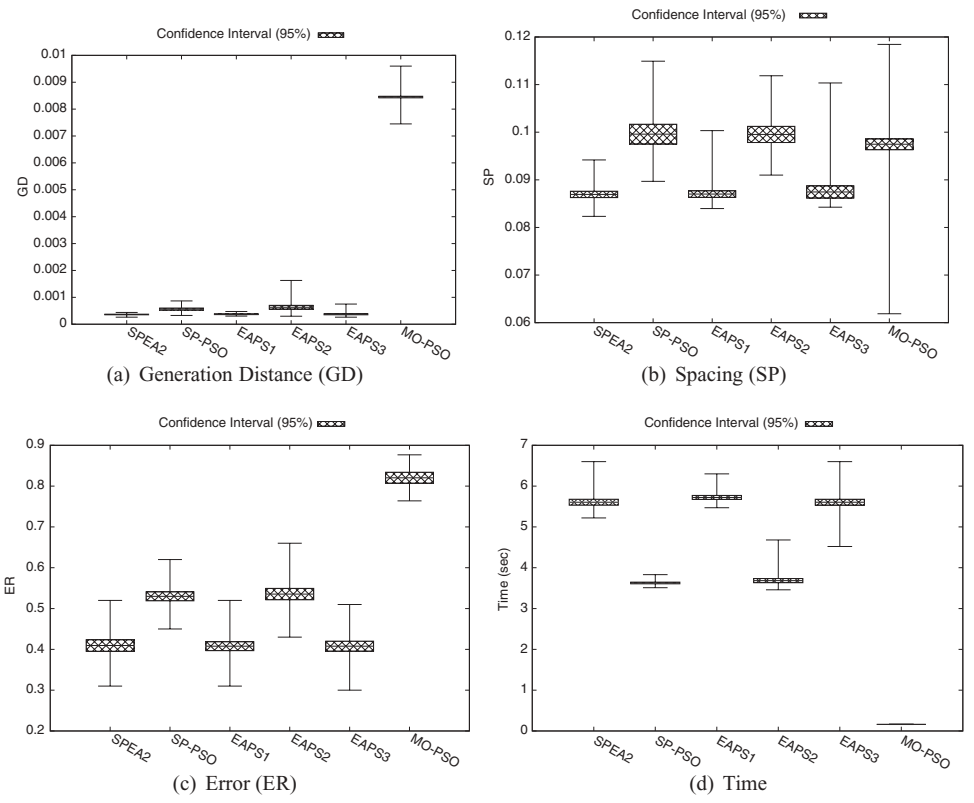


Figure 10: Comparison between different algorithms for benchmark 1.

6 Experimental Results

Four algorithms are proposed in this paper: SP-PSO, EAPS1, EAPS2, and EAPS3. These algorithms are tested using the benchmarks presented in Section 5.1 and compared to SPEA2 and the MO-PSO algorithm introduced in Coello et al. (2004).

The comparison includes the convergence rate and the statistical results of each algorithm over 30 independent runs. The statistical analysis is performed for each of the four metrics presented in Section 5.3.

The statistical analysis includes the confidence intervals around the average of the 30 runs with confidence factor of 95% presented using box and whisker plots. The plot includes the maximum and minimum values encountered for each metric over the 30 runs. The Student's t-test analysis for GD and SP metrics is also presented for each algorithm.

6.1 Convergence

The convergence of all the five algorithms is comparable, with SP-PSO being the slowest algorithm to converge (especially for the ER metric). In benchmarks 3 and 4, in which there are one or more local fronts, several peaks appear as shown in Figure 9(b) for benchmark 3. This shows that all the proposed algorithms managed to escape local fronts.

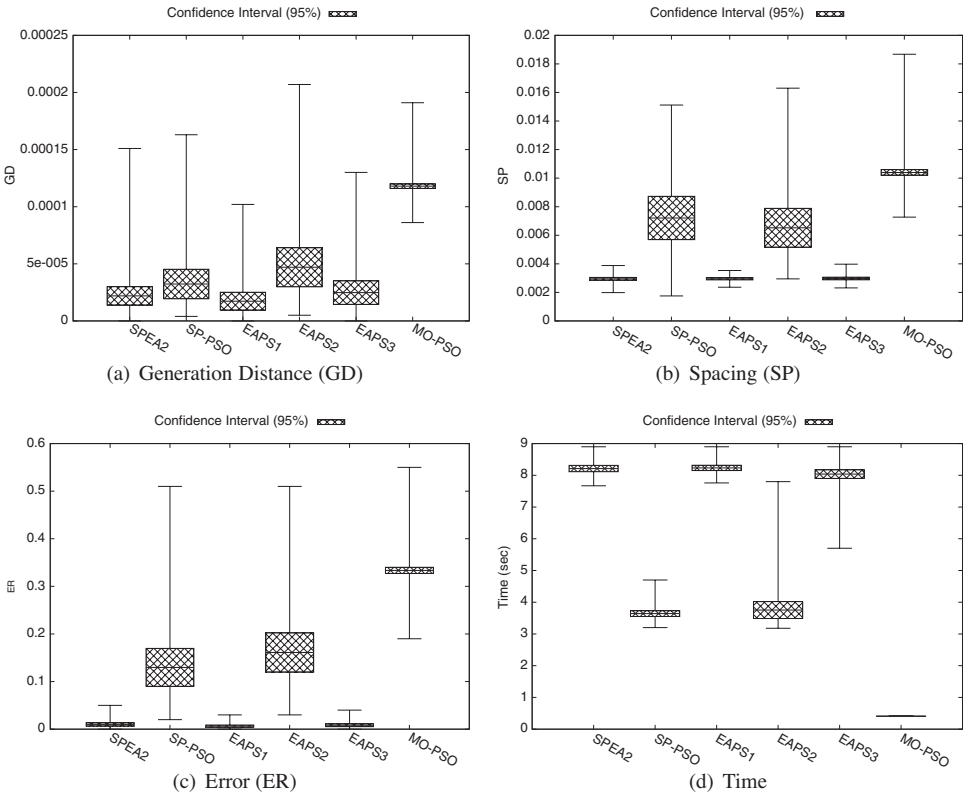


Figure 11: Comparison between different algorithms for benchmark 2.

6.2 Performance Metrics

As discussed in Section 5.3, four metrics are used to evaluate the performance of the algorithms proposed in this paper. The results are presented in Figures 10, 11, 12, 13, 14, 15, and 16 and are discussed in detail below.

6.2.1 GD Metric

For benchmark 1, the Student's t-test results show that SPEA2, EAPS1, and EAPS3 have an equivalent mean for the GD metric with probability of 58%, 59%, and 90%, respectively. On the other hand, SP-PSO and EAPS2 show an equivalent mean with a probability of 20%. Both algorithms produced an average GD value less than, but comparable to that of SPEA2 and its equivalent algorithms. These results can be compared to benchmark 2, where the hybrid algorithms EAPS1 and EAPS3 show an equivalent performance to SPEA2. The performance of SP-PSO is comparable to SPEA2 with probability of 23%. For benchmark 2, EAPS1 shows the best performance for the GD metric.

For benchmark 3, SPEA2, EAPS1, and EAPS3 show an equivalent performance in the Student's t-test, which is equivalent to the results obtained from benchmarks 1 and 2 with the exception that the EAPS3 is the best performing algorithm for this metric. On the other hand, SP-PSO and EAPS2 show an equivalent performance with probability

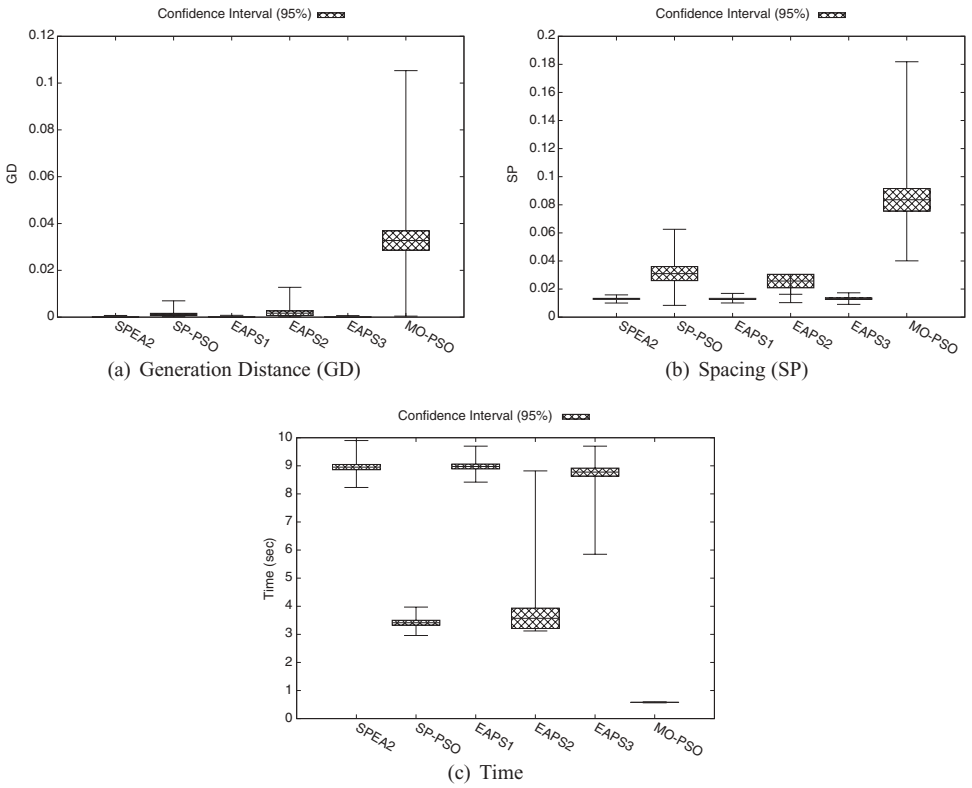


Figure 12: Comparison between different algorithms for benchmark 3.

of 33% for the GD metric. The same results can be seen for benchmark 4 where EAPS3 is the best performing algorithm. In this case, the performance of SP-PSO is comparable to that of SPEA2, while EAPS2 is the worst performing algorithm in terms of the GD metric.

All the proposed algorithms show an equivalent performance for the GD metric when tested using benchmark 5. The Student's t-test shows that all the averages are equivalent, although EAPS1 shows the lowest average. For benchmarks 6 and 7, the SPEA2 algorithm and the two hybrid algorithms EAPS1 and EAPS3 show an equivalent performance which is better than SP-PSO and EAPS2, although the difference in average between the two groups is not significant. Comparing the results of the proposed algorithms to MO-PSO (Coello et al., 2004), it is clear that all the proposed algorithms outperform MO-PSO in the GD metric.

6.2.2 SP Metric

Comparing the SP metric for benchmarks 1, 2, 3, and 4, it is clear that SPEA2, EAPS1, and EAPS3 show equivalent performance that is generally better than SP-PSO and EAPS2. The performance of MO-PSO is comparable to SPEA2 and SP-PSO for the SP metric. For this metric, EAPS3 is the best performing algorithm. For benchmark 5, EAPS1 shows the best performance for the SP metric, although the results are close for all five algorithms.

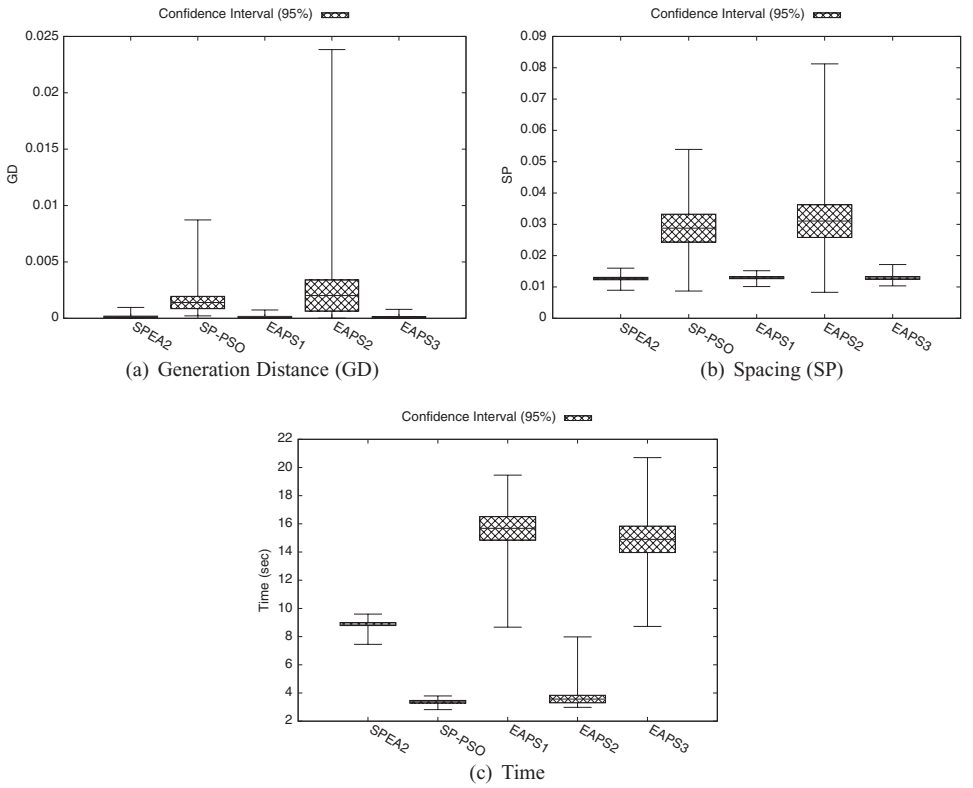


Figure 13: Comparison between different algorithms for benchmark 4.

For benchmarks 6 and 7, the SPEA2 algorithm and the two hybrid algorithms EAPS1 and EAPS3 show an equivalent performance that is better than SP-PSO and EAPS2. The EAPS1 shows the best performance for these two benchmarks.

6.2.3 ER Metric

For benchmark 1, the ER metric shows a similar trend for all the evaluated algorithms as that for GD and SP metrics. The average error resulting from benchmark 2 for SPEA2, EAPS1, and EAPS3 is identical, and these methods outperform all the other algorithms. For benchmarks 3 and 4, all algorithms proposed showed a zero value for the ER metric except for MO-PSO, which has an ER value of 0.335. All algorithms also show an equivalent performance with respect to the ER metric for benchmarks 5, 6, and 7.

6.2.4 Computational Time

For the first four benchmarks, SP-PSO outperforms all the proposed algorithms except for MO-PSO. The MO-PSO has the best computational time compared to the results obtained from SP-PSO. However, the results show that the speed of operation is usually at the expense of the quality of solutions achieved. The MO-PSO requires an additional number of iterations to achieve comparable performance. EAPS2 and EAPS3 show

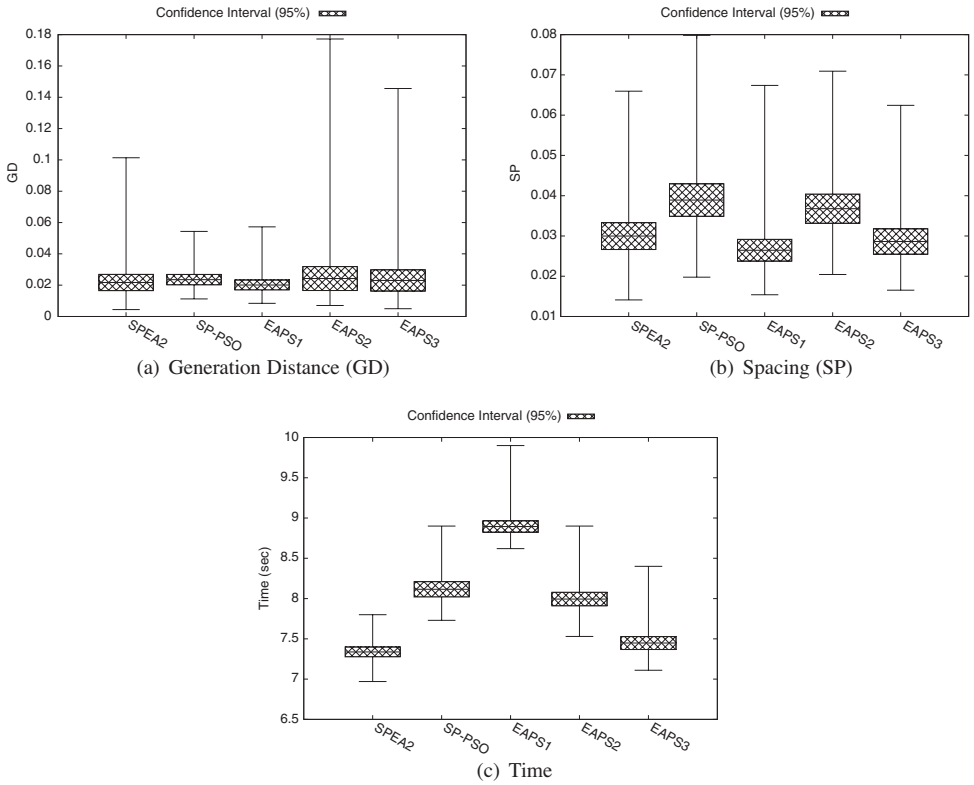


Figure 14: Comparison between different algorithms for benchmark 5.

computational time close to that of SP-PSO with high performance in terms of GD and SP metrics.

For benchmark 5, SPEA2 shows the smallest computational time, although the difference is not significant. The complexity of the problem increases the time required to compute the fitness of each particle and, accordingly, dominates the time required by the algorithm itself. For benchmarks 6 and 7, EAPS3 is the fastest algorithm, although the increase in performance is not significant compared to the other algorithms.

6.3 Discussion

From the above results, the following can be concluded about the proposed algorithms.

1. SP-PSO is a suitable approach for multi-objective optimization, since it is based on a fast and simple model. However, the problem representation should be modified to be more suitable for PSO computational model.

2. The three proposed hybrid forms of EA-PSO are valid and result in an improvement compared to both SPEA2 and SP-PSO algorithms.

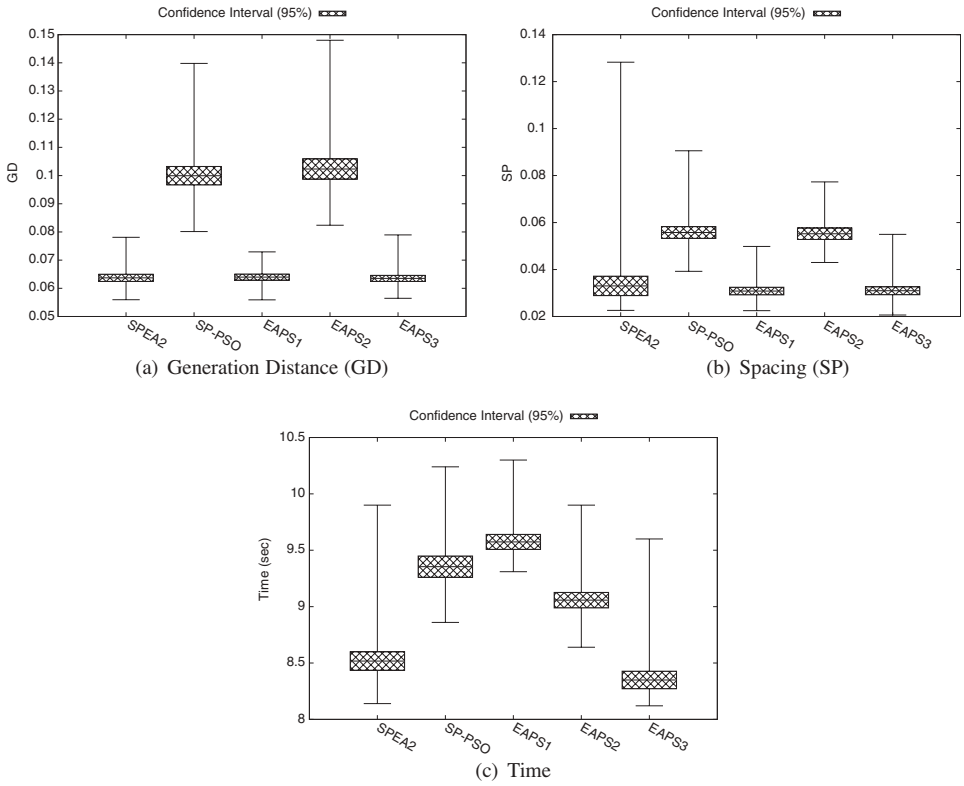


Figure 15: Comparison between different algorithms for benchmark 6.

3. EAPS3 is considered the best performing hybrid algorithm in general for all the benchmarks, although it is not consistently better for a specific metric and benchmark.

4. The hybrid based algorithms show promising results for multi-objective optimization and can be applied for real world applications with more complicated objective functions.

5. The low performance of SP-PSO is mainly due to the variable representation of the problem which is used for all the tests to support both EA and PSO. The current representation, to some extent, is not suitable for PSO. However, due to its simple computational model, SP-PSO operates faster than the other algorithms for all benchmarks. Modifying the problem representation, where problem variables are represented directly with floating point numbers instead of one decision variable for each digit as shown in Section 5.2, can easily improve the performance of SP-PSO.

6. With complex problems, the computational time is almost constant and is not affected by the algorithm itself due to the time consumed in the computation of the fitness function.

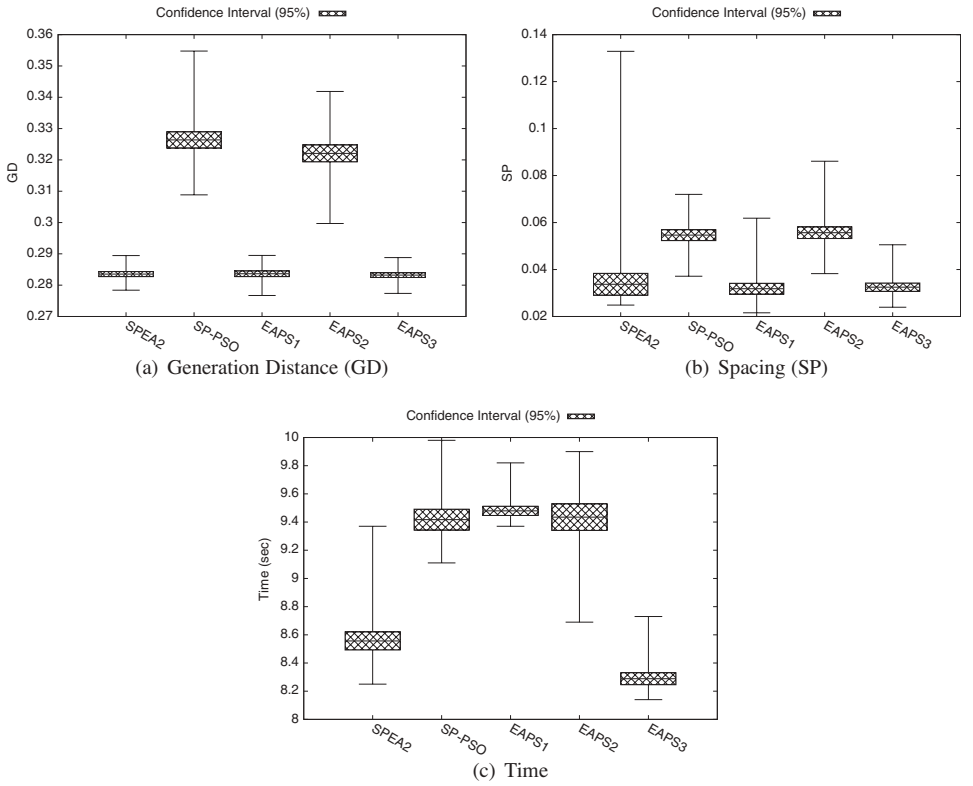


Figure 16: Comparison between different algorithms for benchmark 7.

7. The work presented in this paper outperforms a competitive MO-PSO algorithm. However, this was at the expense of more computational time due to the complexity of the strength Pareto algorithm. More analysis of the algorithm can reduce its complexity and improve the computational requirements.

7 Conclusion and Future Work

In this paper a novel approach for multi-objective particle swarm optimization is proposed. It is based on the strength Pareto (SP) method originally used in evolutionary algorithms (EA) to handle multiple objectives in the well known SPEA2 algorithm (Zitzler et al., 2001, 2002). The new algorithm, SP-PSO, is integrated with SPEA2 to build three hybrid forms of EA-PSO based algorithms. The SPEA2, SP-PSO, and the three hybrid algorithms are tested using several numerical benchmarks and are compared with respect to four metrics. All the algorithms proposed in this paper are also compared to the MO-PSO proposed by Coello et al. (2004).

The results obtained show that the SP-PSO algorithm has a low performance compared to SPEA2 and the three hybrid forms. However, it outperforms the MO-PSO algorithm. The three hybrid forms showed a comparable performance (and sometimes better) to SPEA2 and an improvement in performance compared to SP-PSO. The last two algorithms showed a moderate computation time compared to SPEA2

and EAPS1 due to the use of SP-PSO in half the number of iterations. All the algorithms presented in this paper outperformed the MO-PSO algorithm. The only drawback was in the computation time where MO-PSO proved to have an edge.

Our future work involves modifying the problem representation to improve the performance of the SP-PSO algorithm. We also seek better analysis of the strength Pareto algorithm to reduce its complexity and to reduce the computational time required by the suggested algorithms. Further investigation of advanced adaptive parameter tuning to improve the performance of different algorithms should also be sought. Finally, we intend to apply and use the newly developed hybrid algorithms for VLSI design exploration, which is an important problem in embedded systems design.

References

Boeringer, D., and Werner, D. (2004). Particle swarm optimization versus genetic algorithms for phased array synthesis. *IEEE Transactions on Antennas and Propagation*, 52(3):771–779.

Bratton, D., and Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 120–127.

Clerc, M., and Kennedy, J. (2002). The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6:58–73.

Coello, C., Pulido, G., and Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279.

Deb, K. (1999). *Multi-objective genetic algorithms: Problem difficulties and construction of test problems*, vol. 7, pp. 205–230.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002a). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation*, 6(2):182–197.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2002b). Scalable multi-objective optimization test problems. In *Congress on Evolutionary Computation*, pp. 825–830.

Eberhart, R., and Kennedy, J. (1995). New optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43.

Elhossini, A., Areibi, S., and Dony, R. (2008). An architecture exploration framework for DSP application. In *Proceedings of the Twenty-First Canadian Conference on Electrical and Computer Engineering*, pp. 149–154.

Gill, M. K., Kaheil, Y. H., Khalil, A., McKee, M., and Bastidas, L. (2006). Multiobjective particle swarm optimization for parameter estimation in hydrology. *Water Resources Research*, 42(7):W07417.

Givargis, T., Vahid, F., and Henkel, J. (2002). System-level exploration for Pareto-optimal configurations in parameterized system-on-a-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(4):416–422.

Hiroyasu, T., Miki, M., and Fukunaga, T. (2003). A discussion on real number vector representation, generation alternation models and effect by division of population of genetic algorithm. *Science and Engineering Review of Doshisha University*, 44:25–35.

Ho, S., Yang, S., Ni, G., Lo, E. W. C., and Wong, H. (2005). A particle swarm optimization-based method for multiobjective design optimizations. *IEEE Transactions on Magnetics*, 41:1756–1759.

Huang, V. L., Qin, A. K., Deb, K. E. Z., Suganthan, P. N., Liang, J. J., Preuss, M., and Huband, S. (2007). *Problem definitions for performance assessment on multi-objective optimization algorithms*. Technical Report, Nanyang Technological University, Singapore.

Juang, C.-F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics*, 34:997–1006.

Kennedy, J., and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE Conference on Neural Networks*, pp. 1942–1948.

Kitamura, S., Mori, K., Shindo, S., and Izui, Y. (2006). Modified multiobjective particle swarm optimization method and its application to energy management system for factories. *Electrical Engineering in Japan*, 156:33–42.

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. *LNCS*, vol. 496, pp. 193–197.

Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. *LNCS*, vol. 2723, pp. 37–48.

Lieverse, P., van der Wolf, P., Vissers, K., and Deprettere, E. (2001). A methodology for architecture exploration of heterogeneous signal processing systems. In *Proceedings of the 1999 IEEE Workshop on Signal Processing Systems. SiPS 99. Design and Implementation*, vol. 29, pp. 197–207.

Mahfouf, M., Chen, M.-Y., and Linkens, D. (2004). Adaptive weighted particle swarm optimisation for multi-objective optimal design of alloy steels. *Parallel Problem Solving from Nature 2004*, pp. 762–771.

Mostaghim, S., and Teich, J. (2003). *Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)*, pp. 26–33.

Pimentel, A. D., Erbas, C., and Polstra, S. (2006). A systematic approach to exploring embedded system architectures at multiple abstraction levels. *IEEE Transactions on Computers*, 55(2):99–112.

Reyes-Sierra, M., and Coello, C. (2005). Fitness inheritance in multi-objective particle swarm optimization. In *Proceedings of the Swarm Intelligence Symposium, SIS 2005*, pp. 116–123.

Robinson, J., Sinton, S., and Rahmat-Samii, Y. (2002). PINBOOK swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna. In *IEEE Antennas and Propagation Society International Symposium*, vol. 1, pp. 314–317.

Rudolph, G., Naujoks, B., and Preuss, M. (2007). Capabilities of EMOA to detect and preserve equivalent Pareto subsets. *EMO 2006*, pp. 36–50.

Settles, M., and Soule, T. (2005). Breeding swarms: A GA/PSO hybrid. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pp. 161–168.

Shi, X., Wan, L., Lee, H., Yang, X., Wang, L., and Liang, Y. (2003). An improved genetic algorithm with variable population-size and a PSO-GA based hybrid evolutionary algorithm. *Computer Science and Technology*, 3:1735–1740.

Srinivas, N., and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Yin, P.-Y., Yu, S.-S., Wang, P.-P., and Wang, Y.-T. (2007). Multi-objective task allocation in distributed computing systems by hybrid particle swarm optimization. *Applied Mathematics and Computation*, 184(2):407–420.

Younes, A., Basir, O., and Calamai, P. (2007). Adaptive control of genetic parameters for dynamic combinatorial problems. In *Metaheuristics* (pp. 205–223), Vol. 39 of *Operations Research/Computer Science Interfaces*. Berlin: Springer.

Zhang, L., Ye, B., Jiang, Q., and Cao, Y. (2007). Novel hybrid evolutionary algorithm for multi-objective optimization problem. In *Proceedings of the Parallel Problem Solving from Nature, PPSN*, vol. 14, pp. 110–114.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report No. 1034, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pp. 95–100.

Zitzler, E., and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.