A Particle Swarm Approach to Constrained Optimization Informed by "Global Worst"

David L. Cushman[©]
Penn State Great Valley
School of Graduate Professional Studies
30 East Swedesford Road, Malvern, PA 19355, USA
e-mail: dlc229@psu.edu

Abstract

Particle Swarm Optimization has been used effectively for many types of optimization problems. Many challenges arise when the algorithm is applied to heavily constrained problems where feasible regions may be sparse or disconnected. This paper examines the use of a particle swarm approach to the optimization of thirteen constrained benchmark functions. A multi-objective approach is taken to the problems, and weak dominance is used to determine fitness. Variations on the basic particle swarm algorithm are explored, including the use of perturbation operators and global best versus neighborhood best approach. Results from both random and uniform particle initialization are compared. The possibility of exploiting both the best and the worst solutions the swarm has seen is explored by means of a "global worst" term that is added to the velocity equation. Initial results are promising, but no single approach proved to be the best for all thirteen problems.

Keywords

Particle Swarm Optimization, Constrained Optimization, Swarm Intelligence, Weak Pareto Dominance

1 Introduction

Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995; Kennedy & Eberhart, 2001; Engelbrecht, 2005; Clerc, 2006) is a stochastic, population-based algorithm inspired by the flocking of birds, schooling of fish, and the swarming of certain kinds of insects in search of food and avoidance of danger. Each particle in the population represents a potential solution to the problem at hand. The problems in this study are all concerned with continuous variables. However, PSO has been adapted for use with discrete variables, as well (Engelbrecht, 2005). While PSO is sometimes classified as an evolutionary algorithm (EA) (Coath & Hulgamuge, 2003; Pulido & Coello, 2004), in its basic form PSO differs from other EAs in some important respects. Most notably PSO does not use the concept of replacement of individuals in the population found in other EA approaches. In Genetic Algorithms, for example, the less fit members of the population are eliminated over many iterations to raise the overall quality of the population. PSO instead uses a cooperative approach between the particles where information about good solutions is shared between members of the population (Engelbrecht, 2005; Sutton et al., 2006). A particle that has low fitness in a given iteration may become a better solution later by moving towards the best solution that its neighbors have seen.

The basic PSO implementation uses only two equations: an equation for updating velocity (Kennedy, 2005) and an equation for updating position.

The position of particle $\vec{x}_i$ at iteration t+1 is given by Equation (1), where *d* represents the dimension:

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \tag{1}$$

Velocity for particle $\vec{x}_i$ in dimension *d* at iteration t+1 is given by Equation (2) below.

$$v_{i,d}(t+1) = wv_{i,d}(t) + \underbrace{c_1 r_{1d}(t)\left[pbest_{i,d}(t) - x_{i,d}(t)\right]}_{cognitive} + \underbrace{c_2 r_{2d}(t)\left[lbest_{i,d}(t) - x_{i,d}(t)\right]}_{social} \tag{2}$$

The right hand side of Equation (2) includes three terms. The first term is the current velocity times an inertia factor *w*. A value for *w* greater than 1 encourages exploration by the particles, but it may lead to velocity explosion (Clerc, 2006; Engelbrecht, 2005) where the particles fly out of the search space. A very small value will encourage rapid convergence of the swarm, but the solution found may not be optimal. When this formula is used, a value close to 0.7 has been found empirically to give good results generally. The second term represents what the particle itself has learned, and it is sometimes referred to as the "cognitive" term. The difference between this "particle's best" (for dimension d) and the current value of the particle is multiplied by a cognitive coefficient c_1 and a randomly selected value r_{1d} in the range [0,1]. The cognitive component in the velocity equation represents the best performance of a particle up to iteration t. In the third term, sometimes referred to as the "social term," the difference between the group best (for dimension *d*) and the current value of the particle is multiplied by a social coefficient c_2 and a random number r_{2d} in the range [0,1]. The social component represents the group best solution seen up to iteration t. The social and cognitive components, c_1 and c_2 (usually held static), are sometimes referred to as acceleration coefficients and can be weighted to give relatively more importance to the social or the cognitive term. It has been determined empirically that values in the range [0.5, 2.5] are generally best (Engelbrecht, 2005). Pulido and Coello (2004) used values in the range [1.5, 2.5] in their study.

A particle swarm algorithm must define a neighborhood for each particle. The neighborhood is the set of particles with which a given particle shares information, and it has both a size and a topology. Many different neighborhood topologies have been defined and are sometimes referred to as the "social network structure" of the swarm. Social network structures for PSO include Star, Ring, Wheel, Pyramid, Four Clusters, and Von Neumann social structures (Engelbrecht, 2005). For the purposes of this study, only the Ring and Star topologies were used. A particle swarm algorithm which uses a neighborhood with a size less than the size of the entire swarm tracks the "local best" (sometimes referred to as "neighborhood best"). In the Ring topology each particle communicates with a given number of immediate neighbors. The neighborhoods overlap, so a local best approach tends to slow down convergence. However, it does not prevent sharing of information about optimal solutions across the entire population. The special case of local best where all other particles are in the neighborhood of a given particle is called "global best." In this case the topology is the Star topology since every particle is informed by every other particle (Mendes et al., 2004). This paper refers to global best as "gbest" and to local best as "lbest." In Equation (2) above "lbest" in the social term may be replaced by "gbest" if the neighborhood is the entire population.

PSO has been used effectively for optimization of many kinds of problems and many real-world applications including the design of aircraft wings, antennas, and cantilevered beams as well as truss optimization, generator maintenance scheduling, and aircraft mission route planning. In the basic PSO implementation there is no built in mechanism for dealing with constraints. Since many real-world applications have constraints, this makes constraint handling in PSO an important area of research (Hu & Eberhart, 2002; Zavala et al., 2005; Mezura-Montes & Coello, 2005).

The general nonlinear optimization problem may be characterized as follows:

Find $\bar{x}$ which optimizes $f(\bar{x})$,
subject to : $g_i(x) \le 0, i = 1, ..., n$ and $h_i(x) = 0, i = 1, ..., p$

Equality constraints may be converted to inequalities as follows:

$$|h_i(\vec{x}) - \varepsilon| \leq 0$$

where ε is some suitably small tolerance value. By making this adjustment all constraints become inequalities.

There are several methods that have been applied to constraint handling by stochastic algorithms in general. These include rejecting infeasible solutions altogether, applying penalty functions, converting the constrained problem to an unconstrained problem and solving the new problem, using feasibility-preserving methods that make already feasible solutions better, Pareto ranking methods, and repair methods that change infeasible solutions to feasible (Engelbrecht, 2005). For an in-depth survey of constraint-handling methods see Coello (2002).

Infeasible solutions may be rejected for the purpose of determining personal best or neighborhood (or global) best solutions. This method has the advantage of drawing infeasible solutions back towards the best values seen, which are guaranteed to be feasible. However, the diversity of the swarm may suffer if the ratio of infeasible to feasible particles becomes too large. This would result in convergence to solutions that are less than optimal.

The penalty function approach, in which a penalty is added to the objective function, is meant to discourage searching in infeasible regions of the search space. Engelbrecht (2005) observes that a difficulty with this approach is that it tends to change the shape of the objective function itself and may introduce false optima into the original search space.

A feasibility preserving method requires that the solutions be initialized in feasible space. This has the advantage of starting with feasible solutions from which better solutions may be found. However, if the ratio of the size of the feasible space to the size of the search space is small, it may be computationally expensive to achieve initialization (Hu & Eberhart, 2002). A related approach would be repair methods which begins with feasible solutions and would use the particle's best to pull an infeasible solution back into the feasible region.

Approaches using Pareto-optimality and dominance have been used effectively in multi-objective optimization (Baumgartner et al., 2004; Coello & Mezura-Montes, 2002). One implementation of this approach uses the following rules: if both particles are feasible, select the non-dominated particle; if one of the particles is feasible, select that particle; if both particles are infeasible, select the particle with the least constraint violations. This study used a variation of this method explained in greater detail below.

Pulido and Coello (2004) noted that there has been remarkably little theoretical work done in dealing with constraints in PSO, yet most real-world applications have constraints as noted earlier. They proposed an approach with two components: a *turbulence operator* and a *decision-making scheme* based on closeness to the feasible region *to choose a "best"* (neighborhood or global best). The turbulence operator encourages exploration, while the decision-making scheme encourages exploitation. To test their approach they used thirteen benchmark functions (see Appendix) which offered a mixture of characteristics and varying degrees of difficulty.

How can the difficulty of a problem be measured? Michalewicz and Schoenauer (1996) identified several of the parameters in this type of problem: the number of linear, nonlinear, and active constraints, the type of objective function (e.g., linear, non-linear, quadratic), or the estimated ratio of feasible points in the search space. This last metric is given by the size of the feasible region *F* divided by the size of the search space *S*. See Equation (3) below. This ratio was derived experimentally by generating 1,000,000 random points within the search space and then determining how many of those points lay within the feasible region. Pulido and Coello (2004) also used this metric. However, their numbers differ somewhat from those of Michalewicz and Schoenauer (1996).

$$\rho = |F|/|S| \tag{3}$$

Table 1 below shows the estimates for ρ of Pulido and Coello (2004) as well as the dimensions, the type of objective function, and the number and type of the constraints for the benchmark functions listed in the Appendix.

This paper describes work that was done to follow up on the direction suggested by Pulido and Coello (2004). Experiments were conducted with two different perturbation operators to determine their effect on preventing premature convergence of the population. In addition a new feature was added: an attempt was made to incorporate information from bad solutions that the population had seen in each iteration. This was defined in terms of constraint violation. In keeping with the inspiration for PSO from nature, it was recognized that a population in nature not only moves towards promising areas but also learns to avoid perilous areas. In an attempt to implement this negative learning, a "global worst" was determined for each iteration and used in the calculation of the new velocity. The global worst was also used in one of the two perturbation operators that were explored. This was an attempt to find the right mix of exploration and exploitation.

Table 1 Values of ρ for the 13 test problems (Pulido & Coello, 2004)
(LI=Linear inequalities, NI=Non-linear inequalities, LE=Linear equalities, NE=Non-linear equalities)

Problem	n	Function	ρ	LI	NI	LE	NE
g01	13	quadratic	0.0003%	9	0	0	0
g02	20	nonlinear	99.9973%	1	1	0	0
g03	10	nonlinear	0.0026%	0	0	0	1
g04	5	quadratic	27.0079%	0	6	0	0
g05	4	nonlinear	0.0000%	2	0	0	3
g06	2	nonlinear	0.0057%	0	2	0	0
g07	10	quadratic	0.0000%	3	5	0	0
g08	2	nonlinear	0.8581%	0	2	0	0
g09	7	nonlinear	0.5199%	0	4	0	0
g10	8	linear	0.0020%	3	3	0	0
g11	2	quadratic	0.0973%	0	0	0	1
g12	3	quadratic	4.7697%	0	729	0	0
g13	5	nonlinear	0.0000%	0	0	1	2

2 Proposed Approach

The set of thirteen problems (g01 through g13) used for this study (see Appendix) contain a mixture of both maximization and minimization problems and represent a mix of linear, quadratic, and nonlinear objective functions. The four maximization problems (g02, g03, g08, g12) among the thirteen were converted to minimization problems by reversing the sign of the objective function. The coordinates for each particle were represented as a vector $\vec{x}$ of real values where the dimensions ranged from two (g06, g08, g11) to twenty (g02).

Each problem was converted to a multi-objective optimization problem in which not only an optimum (minimum) value was sought for the objective function but a (minimum) constraint violation of 0 was sought for each of the constraint functions. Fitness for each solution was therefore represented as a vector of functions to be minimized:

$$f(\vec{x}) = (objective(\vec{x}), g_1(\vec{x}), \ldots, g_C(\vec{x})) \tag{3}$$

where C is the number of constraints for a problem (including boundary value constraints) and $objective(\vec{x})$ is the value of the objective function for the problem.

Fitness was evaluated using weak Pareto dominance (Balke & Guntzer, 2005). For a given particle each member of the fitness vector was compared with the corresponding member of the particle's best fitness vector as well as that for the global best.

$$f(\vec{x}_{pbest}) \preceq f(\vec{x}_p) \tag{4}$$

If each member of the particle's fitness vector was less than or equal to each member of the "particle's best" fitness vector, that is, weakly dominated it as indicated in Equation (4) above, the current particle became the new "particle's best." The same method was applied to determine the gbest or lbest.

Information from the global worst violation of each constraint was stored. Each constraint had a "worst offender" associated with it. In each iteration for each constraint, this global worst was updated. However, in order to represent the least feasible region seen so far, a centroid was created by determining the arithmetic mean of the coordinates of the particles stored as global worst.

Rather than using the classic approach to calculating the velocity of a particle shown above, the present study modified a method developed by Clerc (2006) using a constriction coefficient. The constriction approach was developed as a dynamic way of encouraging convergence.

$$v_{id}(t+1) = \chi\left[v_{id}(t) + \underbrace{\phi_1(pb_{id}(t) - x_{id}(t))}_{Pbest} + \underbrace{\phi_2(gb_d(t) - x_{id}(t)}_{Gbest} - \underbrace{\phi_3(gw_d(t) - x_{id}(t))}_{Gworst}\right] \tag{5}$$

$$\text{where } \chi = \frac{2\kappa}{\left|2 - \phi - \sqrt{\phi(\phi-4)}\right|} \tag{6}$$

with $\phi = \phi_1 + \phi_2;\ \phi_1 = cognitive * r_1;\ \phi_2 = social * r_2;\ \phi_3 = 0.0001\, and\ \phi \geq 4;\ \kappa \in [0,1]$.

After initial experimentation with the cognitive and social coefficients, it was determined empirically that they should be held static at 1.47 and 2.47 respectively. By setting the social coefficient greater than the cognitive, the particles are more attracted to the global best than to the particle's best. Each of these two coefficients was multiplied by a randomly-generated factor in the real interval [0, 1].

The global worst term ("Gworst") was added in Equation (5) to experiment with the possibility of informing the swarm not only by the best that had been seen but also by unpromising solutions in terms of feasibility. Could information about infeasible solutions be exploited in some way? In nature swarming insects would tend not only to be drawn towards areas that promised food but would also seek to avoid danger and obstacles. A similar idea has occurred to a group of researchers in Greece (Leontitsis et al., 2006), who related this idea to natural swarms and introduced into PSO the idea of the "repellor." Since their research was not concerned with constrained optimization, however, their implementation of the idea differs significantly from that reported in this paper. Their "worst" was defined in terms of the value of the objective function whereas this study focused on feasibility. It was determined empirically that a coefficient of 0.0001 for the global worst term yielded the best results. The idea was to provide a small nudge away from the center of the least feasible solutions seen thus far.

Each function was tested using global best (gbest) and neighborhood best (lbest) in turn. The neighborhood best used a neighborhood size of 2 where the neighborhoods were determined on the basis of particle indices in a Ring topology. The neighborhood for a particle with index *i* includes particles p_{i-1}, p_i, and p_{i+1}.

Two different perturbation operators were tested. In the first a particle was selected randomly from the swarm in each iteration and randomly reinitialized, thus sending it off to another part of the search space. The second method was to select one particle in each iteration randomly and change its coordinates to those of the global worst, again sending it off to a different part of the search space, but in a different manner. The algorithm was also tested with no perturbation operator. Each of these three perturbation options was used for gbest and for lbest, so each function was tested in six different ways.

In keeping with the approach by Pulido and Coello (2004), particle swarm size was set at 40, and 8500 iterations were performed. Thirty independent runs were done for each of the six implementations of the algorithm making a total number of 10,200,000 solutions generated for each of the six tests of each benchmark. A starting value of $\varepsilon = 0.00175$ was used which was reduced on each iteration by multiplying it by 0.999 if feasible solutions had been found. If the stop-and-restart feature was used because no feasible solutions had been generated by midway through the test, then ε was reset to 0.00175. In the results reported for functions where equality constraints had been converted to inequalities, only those solutions where $\varepsilon < 0.001$ were accepted as feasible. Of course, for a real-world application, an acceptable value of ε would be problem dependent.

The particle swarm was initialized using random initialization, where particles were randomly distributed throughout the bounded region. However, if no feasible solutions had been generated by halfway through 8500 iterations, the swarm was reinitialized using uniform initialization within the bounded region. The particles were spread evenly throughout the bounded region, and the remaining iterations were run.

3 Results

Tables 2 and 3 below show results for the global worst and the random perturbation operators, respectively. Table 4 shows results for the control where no perturbation operator was employed. In each table overall best results are shown in bold. Functions g03, g05, g11 and g13 involved equality constraints that were converted to inequalities using a tolerance of 0.001 as indicated above. The best result known or reported is shown in the "Optimal" column for the sake of comparison.

Table 2 Results for global worst perturbation operator
*(GB=Global best; LB=Local best; ***=No feasible solutions found)*

		Best Result		Mean Result		Worst Result	
Problem	**Optimal**	**GB**	**LB**	**GB**	**LB**	**GB**	**LB**
g01	-15.000000	**-14.880313**	-6.202094	-14.056749	-4.778408	-12.161777	-4.274730
g02	0.803619	**0.761717**	0.335616	0.530459	0.266451	0.313739	0.196716
g03	1.000000	0.996866	***	0.981247	***	0.939279	***
g04	-30665.539000	**-30665.395361**	-30446.129056	-30634.757682	-30215.315529	-30435.530683	-30070.693308
g05	5126.498000	5144.208250	5346.214331	5175.376352	5503.366354	5206.544449	5585.126648
g06	-6961.814000	-6884.630125	-6932.550153	-5866.541259	-5877.790616	-3513.085239	-3049.096300
g07	24.306000	25.429051	***	28.944208	***	38.178045	***
g08	0.095825	**0.095825**	0.095808	0.095820	0.094385	0.095775	0.086728
g09	680.630000	680.647206	745.440304	681.113625	911.637164	687.155101	1167.090690
g10	7049.330700	7102.690148	14906.945758	8932.346704	18213.006566	19257.649020	21450.256510
g11	0.750000	0.749073	0.749092	0.749223	0.750900	0.749260	0.753549
g12	1.000000	**1.000000**	0.999997	1.000000	0.999950	1.000000	0.999822
g13	0.053950	**0.116590**	***	0.300965	***	0.596544	***

The best results were obtained in this study with the gbest neighborhood. In fact lbest shared in the best results in only two problems (g08 and g12) but never yielded the best results alone. Between the two perturbation operators, global worst performed better in four problems (g01, g02, g04, and g13). The random perturbation operator yielded better results in six problems (g03, g05, g07, g09, g10 and g11). In one problem (g06) the use of no perturbation operator yielded best results. In two problems (g08 and g12) mixed results were experienced where gbest with both perturbation operators and gbest and lbest with no perturbation operator yielded the same results.

The combined methods reported in this paper performed reasonably well in cases where the global optima lie on the boundaries of the feasible regions (g01, g02, g04, g06, g07 and g09). Compared to the results of Pulido and Coello (2004), the methods reported here experienced some difficulty in finding the optima in problems with very small feasible regions (g05 and g13), although with refinement of technique, these results might be improved. The

combination of gbest with global worst perturbation performed reasonably well on a problem with a large search space and small feasible region (g10).

Table 3 Results for random perturbation operator
*(GB=Global best; LB=Local best; ***=No feasible solutions found)*

		Best Result		Mean Result		Worst Result	
Problem	**Optimal**	**GB**	**LB**	**GB**	**LB**	**GB**	**LB**
g01	-15.000000	-14.183767	-10.859	-13.683489	-9.276258167	-12.775817	-7.812016
g02	0.803619	0.532535	0.271412	0.374253	0.2263528	0.247025	0.195425
g03	1.000000	**1.001690**	***	0.996993	***	0.990526	***
g04	-30665.539000	-30665.361381	-30425.34808	-30647.498515	-30229.95976	-30536.469576	-30014.1642
g05	5126.498000	**5141.443449**	5466.62204	5142.775155	5466.62204	5144.106861	5466.62204
g06	-6961.814000	-6961.766529	-6885.324437	-6944.104108	-5139.406149	-6866.559330	-1491.604301
g07	24.306000	**24.540138**	1185.002512	26.082900	1185.002512	28.634068	1185.002512
g08	0.095825	**0.095825**	0.095815	0.095804	0.0942495	0.095489	0.083234
g09	680.630000	**680.640230**	741.079086	681.267479	935.6956561	698.091206	1307.36506
g10	7049.330700	**7096.713992**	18233.4814	7919.061614	18233.4814	12402.687120	18233.4814
g11	0.750000	**0.749032**	0.749401	0.749159	0.750575333	0.749615	0.751621
g12	1.000000	**1.000000**	0.999998	1.000000	0.9999498	1.000000	0.999757
g13	0.053950	0.575132	***	0.711534	***	0.964576	***

Table 4 Results with no perturbation operator
*(GB=Global best; LB=Local best; ***=No feasible solutions found)*

		Best Result		Mean Result		Worst Result	
Problem	**Optimal**	**GB**	**LB**	**GB**	**LB**	**GB**	**LB**
g01	-15.000000	-4.912774	-5.414698	-4.294579	-4.760484	-3.806425	-4.138286
g02	0.803619	0.450395	0.656417	0.314473	0.435320	0.206084	0.298636
g03	1.000000	0.932286	0.964373	0.781854	0.637647	0.678244	0.037750
g04	-30665.539000	-30650.115600	-30557.150751	-30502.539180	-29890.409597	-30103.706397	-28972.497952
g05	5126.498000	5141.628223	***	5148.849324	***	5159.265138	***
g06	-6961.814000	**-6961.808263**	-6961.799446	-6932.743299	-6828.536067	-6870.774622	-5626.025743
g07	24.306000	26.125098	24.574098	37.928894	25.237266	66.610951	25.900434
g08	0.095825	**0.095825**	**0.095825**	0.095764	0.095825	0.095566	0.095823
g09	680.630000	680.659986	680.663911	682.608512	686.575140	690.246445	695.850719
g10	7049.330700	7313.447074	7714.311202	7969.264028	7714.311202	9942.081694	7714.311202
g11	0.750000	0.749048	0.749076	0.749156	0.750640	0.749323	0.757652
g12	1.000000	**1.000000**	**1.000000**	0.999625	0.999812	0.994375	0.994374
g13	0.053950	0.740941	***	0.775872	***	0.841074	***

As reported above, two initialization methods were used: random initialization within the given bounds for the problem or a uniform initialization within the bounds in which the particles were spread evenly throughout the bounded region. Table 5 shows that neither method was the best for every function. A “U” indicates that uniform initialization yielded better results, whereas an “R” means that random initialization produced better results. Those cells containing “***” indicate a case where no feasible solutions were generated.

Table 5 Initialization method by neighborhood and perturbation operator (Gworst=global worst; U=Uniform initialization; R=Random initialization)

	gbest			lbest		
	Gworst	**Random**	**None**	**Gworst**	**Random**	**None**
g01	U	U	R	U	U	U
g02	R	R	R	R	R	R
g03	U	U,R	U	U,R	U,R	U
g04	R	R	R	R	R	R
g05	R	R	R	R	R	***
g06	U,R	R	R	U,R	U,R	R
g07	R	R	R	***	***	U,R
g08	R	R	R	R	R	R
g09	R	R	R	R	R	R
g10	R	R	R	***	***	R
g11	R	R	R	R	R	R
g12	R	R	R	R	R	R
g13	R	R	R	***	***	***

To illustrate the effect of the different initialization methods, Figure 1 below shows the value of function g01 at initialization. Extreme negative values were removed from the chart illustrating uniform initialization. As shown in Table 5 above, uniform initialization yielded better results for g01 in five of the six tests. The chart on the left below shows the results of random initialization and clearly indicates that values for g01 did not start close to the optimum of -15. On the other hand, the values for uniform initialization range from 2.388 to -28386.80325. These values span the target value of -15, as the chart to the right clearly indicates. In the case of g01, uniform initialization started the swarm closer to the target value leading to a better outcome. Similar considerations apply to the other problems, as well. Table 5 above shows that in most cases random initialization yielded better results.

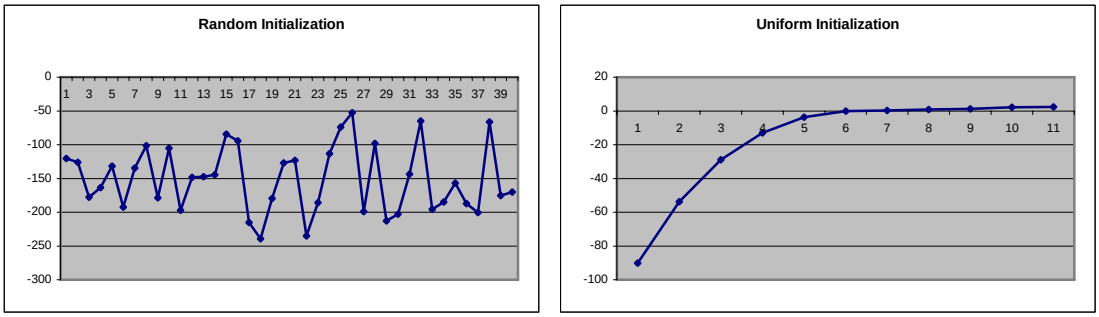

Figure 1 Random versus Uniform Initialization for Problem G01

Table 6 below shows by neighborhood type and perturbation method the percentage of feasible solutions generated for each function. The strategy that produced the best overall result was not necessarily the strategy that produced the highest percentage of feasible solutions in every case. The best value for g02 from this study was produced by gbest using the global worst perturbation operator with an objective value of 0.761717 as shown in Table 2 above. While lbest using the global worst perturbation operator produced the highest percentage of feasible solutions, for g02 the best value produced by this strategy was a mediocre 0.335616.

Table 6 Percentage of feasible solutions generated by function by neighborhood and perturbation operator (Gworst=global worst)

	gbest			lbest		
	Gworst	**Random**	**None**	**Gworst**	**Random**	**None**
g01	3.8636%	3.1487%	0.3295%	2.4827%	2.2676%	7.5730%
g02	28.4573%	57.0870%	36.7702%	79.3556%	76.3959%	36.8438%
g03	0.1139%	0.0243%	0.0861%	0.0026%	0.0759%	0.0759%
g04	5.8915%	7.6614%	4.2257%	27.1282%	33.9971%	52.9435%
g05	0.1687%	0.2074%	0.6301%	0.1219%	0.0544%	0.0000%
g06	0.0111%	0.6509%	0.0666%	0.0133%	0.0071%	22.8561%
g07	3.8760%	14.1904%	1.3418%	0.0000%	0.0000%	1.3411%
g08	68.5223%	64.0144%	59.4543%	0.6687%	1.3582%	72.9681%
g09	16.0889%	33.4545%	1.8209%	2.6409%	0.9706%	55.0974%
g10	0.2559%	0.2512%	0.1416%	0.0000%	0.0000%	1.1103%
g11	0.1400%	0.0000%	0.2692%	0.0536%	0.0383%	0.3619%
g12	81.6932%	78.4878%	99.2963%	7.5960%	6.9978%	97.5919%
g13	0.0154%	0.1141%	0.0057%	0.0000%	0.0000%	0.0000%

Five problems (g01, g02, g05, g10, g13) left room for improvement in the overall best value found for the objective function. To determine the effect of the global worst term in the velocity equation for these problems, another test was performed in which it was removed from the velocity equation. Then thirty independent runs were performed for each of these five problems with 40 particles and 8500 iterations using the neighborhood and perturbation combination in each case that produced the overall best results for them in the earlier test. Table 7 below shows the results of this new test. See Tables 2, 3 and 4 above for comparison with earlier results. Four of the problems (g01, g02, g10, g13) did not perform as well under these new test conditions. The best, mean, and worst results were not as good as in the previous test, which included the global worst term in the velocity equation. One problem (g05) improved on the best result in this new test, but the worst and mean were not as good as those achieved earlier. This test clearly demonstrates that the inclusion of the global worst term in the velocity equation contributed positively to the overall results achieved.

Table 7 Results for five problems re-tested with the global worst term removed from the velocity equation

	Best	**Mean**	**Worst**
Problem			
g01	-14.798564	-12.531152	-6.989869
g02	0.71003	0.526726	0.280714
g05	5131.358701	5377.207878	5623.057055
g10	7358.41727	8870.040475	11942.27445
g13	0.474551	0.523053	0.636602

What was the computational cost of tracking the global worst? Extra overhead was required to store data for the particles that maximally violated each constraint of a problem. As explained earlier, from these data the global worst centroid was computed as the arithmetic mean of the stored coordinates once each iteration. The problems were treated as multi-objective problems, so a fitness vector was stored for each particle, as well. The computation of the global worst centroid added slightly to the total cost per iteration as coordinates for this point have to be calculated (dimensions times constraints). This was the equivalent of one extra fitness function evaluation per iteration.

3 Conclusions and Recommendations

This study sought to find a way to use information from the worst solutions seen by a swarm where "worst" was defined in terms of constraint violation. The study demonstrated that information from the worst could successfully be exploited in order to find feasible solutions in a heavily constrained search space. Thus the emphasis was on feasibility. In future work the concept of the worst solution could be refined to incorporate information about the worst objective function values seen as well.

In ten of the thirteen problems (76.92%), the strategy that yielded the highest percentage of feasible solutions did not produce the best overall results. Problems g07, g08, and g12 were the exceptions, although g08 and g12 each had four strategies yield equivalent results. Clearly with this set of problems the conditions contributing to better solutions are not the same conditions that lead to a large number of mediocre solutions. Exploration of the search space and the fostering of diversity in the particle swarm were encouraged by the use of two perturbation operators (Monson & Seppi, 2006). It is important to note that for seven of the thirteen problems, the strategies that produced the greatest percentage of feasible solutions involved no perturbation operator at all. The perturbation operators proved to be essential to obtaining good results. Random perturbation proved to be slightly more effective (g03, g05, g07, g09, g10, g11) than the global worst operator (g01, g02, g04, g13).

Converting the problem to a multi-objective optimization problem where the constraints were handled by driving down their violations proved to be a successful approach to constrained optimization. The use of a weak dominance approach for the fitness evaluation worked well.

As indicated earlier, PSO is a stochastic algorithm and depends heavily on the generation of pseudo-random numbers. Clerc (2006) has observed that the ANSI C pseudo-random number generator (PRNG) as implemented by most compilers is statistically flawed and may give less than optimal results. No attempt was made in this study to address that concern. A next step would be to replace the PRNG used for this study with one that shows a better statistical pattern such as that recommended by Clerc.

Finally, while this study demonstrates that no single implementation of PSO is likely to work equally well over all types of problems, it shows clearly that it is an algorithm that can be adapted successfully to deal with a wide variety of constraint problems. It would therefore be highly misleading to describe PSO as a "black box" algorithm (Wolpert & McReady, 1997). In each case what is known about the nature of the problem itself should provide guidance for the variation of PSO which is used for that problem.

4 Acknowledgements

I would like to thank Dr. Walter Cedeño, my faculty adviser, for his guidance on this project. I would also like to thank my wife, Rhonda, for her constant encouragement.

5 References

Balke, W-T and Guntzer, Ulrich, (2005), Efficient Skyline Queries under Weak Pareto Dominance, *Proceedings of the IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling (PREFERENCE 2005)*, 2005, Edinburgh, UK.

U. Baumgartner, C. Magele, and W. Renhard, (2004), Pareto optimality and particle swarm optimization, *IEEE Trans. Magn.*, vol. 40, no. 2, pp. 1172–1175, 2004.

Clerc, Maurice, (2006), *Particle Swarm Optimization*, ISTE Ltd, Newport Beach, CA, 2006.

Coath, G. and Halgamuge, S.K., (2003), A Comparison of Constraint-Handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization Problems, *The 2003 Congress on Evolutionary Computation, 2003 CEC '03.*

Coello Coello, C.A., Mezura-Montes, E., (2002), Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments, Parmee, I., and ed.: *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002).* Volume 5., University of Exeter, Devon, UK, Springer-Verlag (2002) 273—284

Carlos A. Coello Coello, (2002), Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art, *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245--1287, January 2002.

Engelbrecht, Andries P., (2005), *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, Ltd, West Sussex, England, 2005.

X. Hu and R. C. Eberhart. (2002), Solving constrained nonlinear optimization problems with particle swarm optimization, *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002.*

Kennedy, J.; Eberhart, R., (1995), Particle swarm optimization, *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol.4, no.pp.1942-1948 vol.4, Nov/Dec 1995

Kennedy, J., Eberhart, R, (2001), *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA, 2001.

Kennedy, J. (2005), Why does it need velocity?, *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, Vol., Iss., 8-10 June 2005, Pages: 38- 44

Leontitsis, A., Kontogiorgos, D., Pagge, J., (2006), Repel the swarm to the optimum!\, *Applied Mathematics and Computation* Volume 173, Issue 1, 1 February 2006, pp. 265-273, 2006.

R. Mendes, J. Kennedy, and J. Neves, (2004), The fully informed particle swarm: Simpler, maybe better, *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, 2004.

Mezura-Montes, E.; Coello Coello, C.A., (2005), Identifying on-line behavior and some sources of difficulty in two competitive approaches for constrained optimization, *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol.2, no.pp. 1477- 1484 Vol. 2, 2-5 Sept. 2005

Monson, C. K. and Seppi, K. D. , (2006), Adaptive diversity in PSO, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (Seattle, Washington, USA, July 08 - 12, 2006). GECCO '06. ACM Press, New York, NY, 59-66.

Z. Michalewicz and M. Schoenauer., (1996), Evolutionary Algorithms for Constrained Parameter Optimization Problems, *Evolutionary Computation*, 4(1):1--32, 1996

Pulido, G.T.; Coello, C.A.C., (2004), A constraint-handling mechanism for particle swarm optimization, *Evolutionary Computation, 2004. CEC2004. Congress on* , vol.2, no.pp. 1396- 1403 Vol.2, 19-23 June 2004

Sutton, A. M., Whitley, D., Lunacek, M., and Howe, A., (2006), PSO and multi-funnel landscapes: how cooperation might limit exploration, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (Seattle, Washington, USA, July 08 - 12, 2006). GECCO '06. ACM Press, New York, NY, 75-82.

D. H. Wolpert and W. G. Macready, (1997), No free lunch theorems for optimization, *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

Muñoz Zavala, A. E., Aguirre, A. H., and Villa Diharce, E. R., (2005), Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation* (Washington DC, USA, June 25 - 29, 2005). H. Beyer, Ed. GECCO '05. ACM Press, New York, NY, 209-216

6 Appendix

$$Minimize: g01(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{13} x_i$$

$$subject\ to\ g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0,\ g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0,\ g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0,$$
$$g_4(\vec{x}) = -8x_1 + x_{10} \le 0,\ g_5(\vec{x}) = -8x_2 + x_{11} \le 0,\ g_6(\vec{x}) = -8x_3 + x_{12} \le 0,\ g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \le 0,$$
$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \le 0,\ g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \le 0\ where\ \ 0 \le x_i \le 1 (i = 1,...,9), 0 \le x_i \le 100\ (i = 10,11,12), 0 \le x_{13} \le 1$$

$$Maximize: g02(\vec{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

$$subject\ to: g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \le 0,\ g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \le 0,\ where\ \ 0 \le x_i \le 10\ (i = 1,...,20)$$

$$Maximize: g03(\vec{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

$$subject\ to: h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0,\ where\ \ 0 \le x_i \le 1 (i = 1,...,10)$$

$$Minimize: g04(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$
$$subject\ to: g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$$
$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$$
$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$$
$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$$
$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$$
$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$
$$where\ \ 78 \le x_1 \le 102, 33 \le x_2 \le 45, 27 \le x_i \le 45\ (i = 3,4,5)$$

$$Minimize: g05(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$
$$subject\ to: g_1(\vec{x}) = -x_4 + x_3 - 0.55 \le 0,\ g_2(\vec{x}) = -x_3 + x_4 - 0.55 \le 0$$
$$h_3(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_4(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_5(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$
$$where\ \ 0 \le x_1 \le 1200,\ 0 \le x_2 \le 1200,\ -0.55 \le x_3 \le 0.55,\ -0.55 \le x_4 \le 0.55$$

$$Minimize: g06(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$
$$subject\ to: g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0,\ g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$
$$where\ \ 13 \le x_1 \le 100,\ \ 0 \le x_2 \le 100$$

$$Minimize: g07(\vec{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$
$$subject\ to: g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0,\ g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0,$$
$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0,\ g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0,$$
$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0,\ g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \le 0,$$
$$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0,\ g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0$$
$$where\ -10 \le x_i \le 10\ (i = 1, ..., 10)$$

$$Maximize: g08(\vec{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$
$$subject\ to: g_1(\vec{x}) = x_1^2 - x_2 + 1 \le 0,\ g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \le 0$$
$$where\ 0 \le x_1 \le 10,\ 0 \le x_2 \le 10$$

$$Minimize: g09(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7$$
$$subject\ to: g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0,\ g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0,$$
$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0,\ g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1 x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0$$
$$where\ -10 \le x_i \le 10\ \ (i = 1, ..., 7)$$

$$Minimize: g10(\vec{x}) = x_1 + x_2 + x_3$$
$$subject\ to: g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \le 0,\ g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \le 0,$$
$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \le 0,\ g_4(\vec{x}) = -x_1 x_6 + 833.33252x_4 + 100x_1 - 833333.333 \le 0,$$
$$g_5(\vec{x}) = -x_2 x_7 + 1250x_5 + x_2 x_4 - 1250x_4 \le 0,\ g_6(\vec{x}) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500x_5 \le 0$$
$$where\ 100 \le x_1 \le 10000,\ 1000 \le x_i \le 10000,\ (\imath = 2, 3)\ 10 \le x_i \le 100,\ (i = 4, ..., 8)$$

$$Minimize: g11(\vec{x}) = x_1^2 + (x_2 - 1)^2$$
$$subject\ to: h(\vec{x}) = x_2 - x_1^2 = 0$$
$$where\ -1 \le x_1 \le 1,\ -1 \le x_2 \le 1$$

$$Maximize: g12(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$$
$$subject\ to: g_1(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$
$$where\ 0 \le x_i \le 10\ (i = 1, 2, 3),\ p, q, r = 1, 2, ..., 9$$

$$Minimize: g13(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$$
$$subject\ to: h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0,\ h_2(\vec{x}) = x_2 x_3 - 5x_4 x_5 = 0,\ h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$
$$where\ -2.3 \le x_i \le 2.3\ (i = 1, 2),\ -3.2 \le x_i \le 3.2\ (i = 3, 4, 5)$$