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

Analysis of Constraint Handling Methods for the Gravitational Search Algorithm

Daniel J. Poole
Department of Aerospace Engineering
University of Bristol
Bristol, BS8 1TR, U.K.
Email: dp8470@bristol.ac.uk

Christian B. Allen
Department of Aerospace Engineering
University of Bristol
Bristol, BS8 1TR, U.K.
Email: c.b.allen@bristol.ac.uk

Thomas C. S. Rendall
Department of Aerospace Engineering
University of Bristol
Bristol, BS8 1TR, U.K.
Email: thomas.rendall@bristol.ac.uk

***Abstract*—The gravitational search algorithm (GSA) is a recent addition to the family of global optimization algorithms based on phenomena found in nature, specifically the gravitational attractive force between two bodies of mass. However, like almost all global search algorithms of this type, GSA has no direct method of handling a constrained optimization problem. There has been much attention to constraint handling using other agent based systems, though the mechanics of GSA make the application of many of these difficult. This paper has therefore analysed constraint handling methods for use with GSA and compared the performance of simple to implement methods (penalties and feasible directions) with a novel separation-sub-swarm (3S) approach, and found that feasible direction methods ideally need at least one initially feasible particle, and that the novel 3S approach is highly effective for solving constrained optimization problems using GSA outperforming the other approaches tested.**

I. Introduction

OPTIMIZATION is the process of improving on a current solution. In real world problems, historically optimization has often been performed manually where designers use intuition to produce solutions to problems so that the solution performs better than the initial starting point. However, it has now become commonplace to use automated optimization algorithms to allow a more streamlined and strict approach to this process and with the advent of increased computer power available to engineers, expensive optimization problems are being solved within an entirely automated process, such as the drag minimization of aircraft wings using computational fluid dynamics [1].

The solution to many real world optimization problems require a solution that is feasible; constraints appear on the total cost or other physical barriers to the solution. Mathematically, a single objective constrained optimization problem can be described as:

$$\begin{aligned} \underset{\mathbf{x}\in\Re^n}{\text{minimise}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the solution vector $[x_1, x_2, \ldots, x_n]^T$ where each element of the vector is a design variable, $f(\mathbf{x})$ is the value of the objective function for the given solution vector, $\mathbf{g}(\mathbf{x})$ represents inequality constraints, and $\mathbf{h}(\mathbf{x})$ represents equality constraints. The solution vector is bounded by an upper, U_k, and lower, L_k, bound such that for each x_k where $k = 1, 2, \ldots, n$, the solution must be $L_k \leq x_k \leq U_k$. In certain classes of optimization algorithms, it is typical to transform equality constraints into inequality constraints within some small tolerance: $|\mathbf{h}(\mathbf{x})| - \boldsymbol{\epsilon} \leq \mathbf{0}$.

Solving optimization problems commonly uses one of two approaches: gradient-based or global search algorithms. Gradient-based algorithms use the local gradient as a basis along which to search. The most basic forms of this are the steepest descent and conjugate gradient approaches, however these are designed for unconstrained optimization so require modification to allow the effective handling of constraints; feasible directions and barriers allow constraint handling to be implemented within an unconstrained gradient-based framework. More efficient approaches, which are typical in gradient-based optimization algorithms, involve solving the Karush-Kuhn-Tucker (KKT) conditions. The most widely adopted approach is sequential-quadratic-programming [2], which allows the strict enforcement of constraints without either approximating them or altering the underlying search space of the problem. Work performed at the University of Bristol has shown that a feasible-SQP (FSQP) algorithm is effective for the optimization of 2D [3] and 3D [4], [5] aerodynamic optimization problems.

All gradient-based optimization algorithms do have the same issues which are the computation of the gradient, and the termination in local minima. Computing the gradient can be an expensive process, especially if done using finite difference where the number of objectiveevaluations is proportional to the number of design variables. Furthermore, the gradient evaluation requires a smooth and continuous design space which may not be the case for all problems, or when the solution is close the the upper and lower limits of the design space. All purely gradient-based algorithms will also terminate where the gradient becomes zero i.e. an optimum. For highly multimodal functions (functions with large numbers of local optima) the problem of termination in local optima may not allow the global optimum solution to be found so other approaches are required.

Global search algorithms avoid the issues associated with

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

gradient-based approaches by avoiding the computation and use of the gradient, instead employing the position of the search in the space as a method to build an algorithm. These meta-heuristic approaches often mimic natural processes or behaviour such as evolution in genetic algorithms (GA) [6], cooling in simulated annealing (SA) [7], ant colony food searching in the ant colony optimization (ACO) [8], swarm behaviour in particle swarm (PSO) [9], bee colony food searching in the artificial bee colony system (ABS) [10], Newtonian gravitational laws in gravitational search algorithm (GSA) [11] and hearding of krill in the krill heard framework (KH) [12]. The GA, SA and ACO algorithms are primarily designed for discrete optimization, whereas ABS, PSO, GSA and KH are agent based search algorithms designed specifically for continuous optimization problems.

Agent based search algorithms are effective at solving general unconstrained optimization problems that are large and small scale, unimodal and multimodal. However the framework around which all agent based optimization systems are built does not directly take into account the solution to a constrained problem, and this is an ongoing research area. Furthermore, constraint handling methods for agent based search systems are often designed for use with particle swarm and as such suitable methods for handling constraints using the highly efficient gravitational search algorithm have had little attention. The objective of this paper, therefore, is to review the various approaches used for constraint handling in agent based search algorithms and analyse suitable methods for use with GSA.

II. Gravitational Search Algorithm

The original agent based search system is particle swarm optimization, presented by Kennedy and Eberhart[9]. They introduced the idea of using a set of particles to search a design space in pursuit of the global optimum. The Gravitational Search Algorithm (GSA) is a further addition to this field, presented by Rashedi *et al.* [11] for unconstrained global optimization where the principles of basic Newtonian mechanics act as the basis on which the algorithm is constructed. The algorithm was shown to perform well in many analytical test function due to its use of an agent's position, and therefore fitness, in the search algorithm, where GSA uses this directly to act as a measure for all other particles to see. This notion is manifested in the algorithm by the mass of an agent, where a high mass is equal to a good fitness and *vice versa* for an agent that has a poor fitness. This is further propagated through the population by a gravitational force where every particle is attracted to every other particle by a force that is proportional to the product of masses between particles. Overall, these two effects provide a global transfer of data through the entire swarm, and this is the mechanism that is often attributed to the high performance of GSA.

The algorithm requires a system of N agents to initialised within the bounds of the design variables, where the position of the i-th agent is denoted by:

$$\mathbf{x_i} = \{x_i^1, x_i^2, \ldots, x_i^d\}^T \tag{2}$$

The fictitious mass is calculated based on a particle's fitness:

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \tag{3}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(y)} \tag{4}$$

The force acting on particle i from particle j is:

$$F_{i,j}^d(t) = G(t)\left(\frac{M_i(t)M_j(t)}{R_{i,j}(t)+\epsilon}\right)(x_j^d(t) - x_i^d(t)) \tag{5}$$

$$G(t) = G_0 \exp(-\alpha t/T) \tag{6}$$

The total force acting on the i-th particle is:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{Kbest} \text{rand}_j F_{i,j}^d(t) \tag{7}$$

where $Kbest$ reduces linearly through the optimization from N at the start to 1 at the final iteration, which allows a balance between exploration and exploitation. The acceleration is then:

$$a_i^d(t) = F_i^d(t)/M_i(t) \tag{8}$$

Here, a modified GSA (MGSA) is used as the base swarm which is a hybrid of the GSA and particle swarm systems, and has been shown to be an effective approach [13], [14], [15]. The updating procedure for the particles is:

$$\begin{aligned} v_i^d(t+1) = \text{rand}_i v_i^d(t) + \frac{1}{2}a_i^d(t) \\ + \frac{1}{2}\left(c_1 r_{1_i}(p_i^d - x_i^d(t)) + c_2 r_{2_i}(s^d - x_i^d(t))\right) \end{aligned} \tag{9}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{10}$$

where $\mathbf{p}_i$ is the best position found by the i-th particle so far and $\mathbf{s}$ is the best position found by the swarm; $\mathbf{r}_1$ and $\mathbf{r}_2$ are random vectors; c_1 and c_2 are the cognitive and social parameters respectively. Rashedi *et al.* [11] reported that GSA outperformed PSO in over 80% of the analytical cases tested in their paper, and also for a real optimization problems[16]; Chatterjee *et al.* [17] and Mallick *et al.*[18] demonstrated the superior efficiency and performance of GSA against PSO for multi-modal engineering optimization problems.

Overall, the simplicity of the gravitational search algorithm, and the efficiency and effectiveness of its performance makes it a suitable candidate to act as the basis for the development of a global optimizer for solving multi-modal constrained cases. The following section will survey approaches developed for solving constrained optimization cases using agent-based systems.

III. Constraint Handling in Agent-based Search Algorithms

Agent-based global search algorithms are void of direct constraint handling in their formulation and therefore require modification. A short review is presented here, and a comprehensive review of constraint handling of nature inspired numerical optimization algorithms has been performed by Mezura-Montes and Coello Coello [19], which the reader is guided towards for a deeper review than is covered here.

A. Penalty Functions

The principle of penalty functions is that a constrained optimization problem can be transformed into an unconstrained one by incorporating the constraints into the objective function. The most simple type of penalty is called a 'death-penalty' [20] which eliminates the position of a particle violating a constraint from the search. This usually occurs by either reinitialising the particle at a new position if it violates a constraint, or assigns it a large penalty constant. This blinds the overall swarm from any knowledge of the search space outside the feasible region and is therefore inefficient. Other common approaches use a static penalty function approach which adds the constraint violations to the objective function with some static scaling [21], though more efficient penalties are dynamic [22], so change depending on the current objective function [23].

B. Mutation Operators

Mutation operators work on the principle that a feasible solution is better than an infeasible one, and as such manipulate the search algorithm to either force or tend the solution to the feasible space. The first methods of this nature were based on the idea of feasible directions (from Vanderplaats [24]), which is a direction that reduces the value of the objective function while pointing towards the feasible space, and shown to produce feasible results when applied to real-world optimization problems[21]. Other approaches modify the velocity of a particle to point back towards the feasible space [25], though both of these approaches require initialisation of particles to all be feasible. This prerequisite is often expensive, especially for highly constrained problems where multiple initialisations would be required to obtain feasibility. Moreover, mutation operators mutate the swarm so tend to alter the natural self-organising swarm dynamics that cause the algorithm to be effective at optimizing search spaces.

C. Separation of Objective Function and Constraints

The idea of a penalty function is to combine the value of the true objective function and the constraint violation values into a single augmented objective function, however, there exists a school of thought which is the antithesis to this, i.e. keeping the objective and constraint values separate. These methods use various techniques for optimizing the value of the objective and constraint values, or use various techniques for the treatment of swarm characteristics for separate groups of particles, though in general they seek to optimize either the true objective function or the constraint violation.

Liang and Suganthan [26] employed a sub-swarm approach where sub swarms optimized either a constraint or the objective function, where transfer of data about the feasible space is provided by random reinitialisations of the swarms. A more popular approach is by the use of hard feasibility rules which selects leader particles based on feasibility or constraint violation [27], [28] though it was recognised that random operators had to be added to the hard feasibility rules to facilitate good convergence properties. This effectively represents a diffusion of the hard feasibility rules, which leads to a similar approach called soft feasibility rules where the difference represents a relaxation of the rules. The α-constrained [29] and ϵ-constrained [30] feasibility rules are just an effective relaxation of the hard feasibility rules, where a satisfaction or tolerance represents the relaxation allowed, which also represents the handicap of such methods being the fine tuning of the α or ϵ parameters. Furthermore, the bi-objective problem can also be treated using multi-objective techniques, which find the pareto front of non-dominated solutions [31].

In conclusion, hard feasibility rules are popular for solving constrained optimization algorithms using agent-based systems, though on their own result in premature convergence and therefore require a mutation operator to alleviate this. Soft feasibility rules can allow a greater flexibility by relaxing the binary operator, therefore not requiring a mutation, though at the expense of adding another user defined parameter which requires fine-tuning.

IV. Suitable Constraint Handling Methods for use with GSA

The previous section introduced the methods developed for handling constraints using agent-based global search algorithms, however, these were all developed for a particle swarm system – or modified to suit particle swarm. It has been shown from the review of the literature that the recently developed gravitational search algorithm (GSA) is a more efficient and effective global search algorithm than PSO, outperforming PSO on many analytical and real world optimization cases. There has been little contribution to the discussion of constrained optimization using GSA. A simple implementation is by the use of a penalty function, as Amoozegar and Nezamabadi-Pour [32] did. In general, however, penalty functions are not ideal as they distort the underlying search space and also introduce the problem of selecting an appropriate penalty term, which is highly case dependent. The other approach seen in the literature is by reinitialising infeasible particles either randomly [33], or by the repair method which initialises an infeasible particle near to the closest feasible particle [34]. Reinitialisation encounters problems when the feasible space is small, or disconnected, so overall these approaches are not ideal. It is obvious that an effective constraint handling approach is required for GSA as there is little dealing with it in the literature – researchers instead dealing with PSO – but the

underlying search mechanism has been shown to be effective for unconstrained problems.

The exact mechanisms that allow GSA to be an effective optimization algorithm also act somewhat as a hindrance for coupling with general constraint handling methods that have been developed with particle swarm in mind. Separation approaches, where particles either optimize the constraints or the objective function are difficult in GSA as the global transfer of data among all particles swamps the true optimization with unwanted data about the feasible region. Complications also arise with the evaluation of a particle's mass such that an infeasible particle could have a very large mass compared to a feasible particle owing to the infeasible one being very close to the feasible region so having a high fitness value. The direct use of feasibility rules also has little to no use in GSA as there is no selection of leader particles, unlike in PSO; in GSA the particles are ranked based on their fitness.

Penalty functions can be used in the GSA environment as these just manipulate the objective function so are simple to directly insert into the algorithm. The slight implication in the use of penalties is that an infeasible particle must be forced to have a much worse fitness than a feasible particle, but this can be handled by setting a large penalty weighting. The feasible direction approach can also be used as this manipulates a particle's velocity to point in towards the feasible space. Finally, the authors have developed an effective constraint handling mechanism specifically designed for use with GSA called separation-sub-swarm (3S), which uses a separation approach that is handled by using two independent swarms which are constructed based on their feasibility. The infeasible swarm minimises the constraint violation, whereas the feasible swarm optimizes the true objective function, where the independence of the two swarms is important. Infeasible particles are transferred information about the feasible space by use of feasibility rules where the local and global best positions are decided based on the feasibility history of a particle and the swarm.

A. Penalties

Penalty functions are analysed as they allow direct insertion into any agent-based algorithm and are therefore simple to implement in GSA. Different fidelity penalty functions have been considered. The first is a death penalty which randomly reinitialises an infeasible particle. The second is a static penalty where the augmented objective function is:

$$f^p(\mathbf{x}_i) = f(\mathbf{x}) + 10 \sum_{j=1}^{G} q(\mathbf{x}_i)_j \tag{11}$$

where q_j is the j-th constraint violation of G constraints. Finally a more complicated dynamic penalty [23] has been implemented, where the augmented objective function in this cases is:

$$f^p(\mathbf{x}_i) = f(\mathbf{x}) + \kappa \sum_{j=1}^{G} \theta q(\mathbf{x}_i)_j^{\gamma} \tag{12}$$

where κ is the dynamic penalty which is given by $t\sqrt{t}$ at the t-th evolution; $\theta = 10$ if $q_j < 0.001$, else $\theta = 20$ if $q_j < 0.1$, else $\theta = 100$ if $q_j < 1.0$, otherwise $\theta = 300$; $\gamma = 1.0$ if $q_j < 1$, otherwise $\gamma = 2$.

B. Feasible Directions

The feasible direction approach used here is based on an approach developed for particle swarm by Venter and Sobieszczanski-Sobieski [21] where a manipulation of the velocity vector is made if a particle is infeasible to attempt to force it towards the feasible region:

$$v_i^d(t+1) = \text{rand}_i(s^d(t) - x_i^d(t)) \tag{13}$$

The difference between the feasible direction and the pure GSA velocity is the setting of the previous velocity to be zero and the introduction of the vector that points in the direction of the best position the swarm has found so far, which will be feasible if it is possible. Selecting the best particle to be feasible if it is possible will make sure the velocity points in the direction of feasibility, and also avoid any issues with requiring points to be initialised as being feasible. A further change required is that the mass of any feasible particle must be set as being extremely small to avoid it interfering with the feasible search too much.

C. Separation Sub-Swarm (3S)

An efficient constraint handling method for use with GSA has been developed by the authors that allows independent optimization of the constraints or objective function by a separation-sub-swarm (3S) approach which employs two independent swarms decided based on a candidate's feasibility. The independence of the swarms is important such that global data transfer among all particles does not occur, but between necessary particles. Furthermore, the optimization of the constraint violation, which has a solution at the feasible space, occurs by particle swarm to avoid the issues of setting the correct mass values in GSA. This approach allows transfer of data between feasible particles, and from the feasible region to the infeasible particles by employing feasibility rules and moving infeasible particles by a particle swarm approach. The algorithm is a general framework for use with any agent-based algorithm, though it is designed with GSA in mind. The objective function, f^p associated with the i-th agent is transformed depending on a particle's feasibility:

$$f^p(\mathbf{x}_i) = \begin{cases} f(\mathbf{x}_i) & \text{if } \mathbf{x_i} \text{ is feasible} \\ \sum_{j=1}^{G} \max\{0, g_j(\mathbf{x}_i)\} + & \text{else} \\ \sum_{j=1}^{H} |h_j(\mathbf{x}_i)| & \end{cases} \tag{14}$$

The particle's best ever position, p, and the swarm's best ever position, g, are needed for use by the infeasible particles, which is done by comparing the current position with the best positions by the following rules:

1) If current and best positions are feasible, the one with best fitness wins;

2) If either the current or best positions are feasible and the other infeasible, the feasible position wins;
3) If current and best positions are infeasible, the one with the minimum constraint violation wins.

For the N_f feasible particles only, the acceleration is as per gravitational search. The acceleration of the infeasible particles is done by particle swarm:

$$a_i^d(t) = c_1 r_{1_i}(p_i^d - x_i^d(t)) + c_2 r_{2_i}(g^d - x_i^d(t)) \quad (15)$$

The velocity of the particles is:

$$v_i^d(t+1) = \text{rand}_i v_i^d(t) + a_i^d(t) \quad (16)$$

which represents a 'fuzzy inertia' particle swarm system for the infeasible particles, and is the standard GSA model for the feasible particles. The update procedure is therefore:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (17)$$

If a particle exceeds the boundary of the search space then it is not infeasible but is a particle without a solution so is reinitialised in its last position with a zero velocity.

V. Constrained Analytical Optimization

The performance of the GSA based swarm with the various constraint handling methods introduced above is analysed here. An analytical function suite, as outlined by Michalewicz and Schoenauer [35], which is commonly used to test constrained global optimization algorithms, is employed for the testing in this work. The suite contains 11 test cases that are all minimisation problems and contain various numbers of linear and non-linear inequality and equality constraints, various sizes of feasible search space, and various types of objective function. The nature of the cases is outlined in table I and as can be seen by considering the size of the feasible search space, represent a difficult optimization problem. Furthermore, the presence of equality constraints poses a difficult problem for the optimizer, so for the purpose of this work, as is typical when dealing with equality constraints, they are transformed into inequality constraints to within a small tolerance: $|h_j(\mathbf{x})| - \epsilon \leq 0$.

25 independent runs of each of the test functions were performed. The exact swarm mechanism used is the MGSA as this outperforms the individual mechanisms for many optimization cases [13], [14], [15]. 200 particles were used; maximum number of timesteps at 1500; $G_0 = 30$; $\alpha = 10$; $c_1 = c_2 = 2$. The degree of violation for the equality constraints was $\epsilon = 0.0001$, so as a result optimum solutions better than the theoretical optimum were possible. The objective function evaluations have been parallelised using the message passing interface (MPI) to allow more efficient algorithm performance. The best solutions found from the 25 independent runs for each function using each constraint handling method and the median results are given in table II, and the feasibility rate is given in table III. The convergence of the best result for each function using each method is shown in figure 1.

TABLE I
Summary of eleven analytical test cases, where d is the number of design variables, ρ is the ratio of the feasible search space to the whole search space, and LI, NE, NI represent the number of linear inequalities, non-linear equalities and non-linear inequalities respectively.

Function	d	Type of f	ρ	LI	NE	NI
G1	13	quadratic	0.0111%	9	0	0
G2	20	non-linear	99.8474%	0	0	2
G3	10	polynomial	0.0000%	0	1	0
G4	5	quadratic	52.1230%	0	0	6
G5	4	cubic	0.0000%	9	3	0
G6	2	cubic	0.0066%	9	0	2
G7	10	quadratic	0.0003%	3	0	5
G8	2	non-linear	0.8560%	0	0	2
G9	7	polynomial	0.5121%	0	0	4
G10	8	linear	0.0010%	3	0	3
G11	2	quadratic	0.0000%	0	1	0

TABLE III
Feasibility rate of 11 analytical function test suite optimized using various constraint handling methods with MGSA swarm.

Function	3S	Death	Static	Dyn	FD
G1	100%	36%	100%	68%	0%
G2	100%	100%	100%	100%	100%
G3	100%	60%	0%	96%	0%
G4	100%	100%	0%	28%	24%
G5	88%	0%	100%	36%	0%
G6	100%	100%	0%	100%	80%
G7	100%	12%	100%	96%	0%
G8	100%	100%	0%	100%	88%
G9	100%	100%	100%	100%	16%
G10	100%	12%	0%	0%	0%
G11	100%	100%	100%	100%	4%
Overall	99%	65%	55%	73%	28%

The comparison of various penalties of differing fidelity and complication further demonstrates the effectiveness and efficiency of the separation-sub-swarm algorithm. The efficiency of finding a feasible solution, whether it is the globally optimal solution or not, is almost perfect using the 3S algorithm, and is much less good using penalty approaches. The best performing penalty is the more complicated dynamic penalty followed by the death and finally the static. The death penalty is inefficient as it blinds the swarm from any knowledge of the history of a particle when it becomes infeasible and just ignores that information instead of transferring it back to the feasible swarm. The static penalty also performs poorly compared to the dynamic penalty though its performance could improve with fine-tuning of the penalty parameter, though this is an expensive process. The dynamic penalty performs the best of the three penalty functions, which is to be expected due to the algorithm being able to interactively alter the penalty applied based on the constraint violation. The 3S algorithm outperforms all the penalties on

TABLE II
RESULTS OF 11 ANALYTICAL FUNCTION TEST SUITE OPTIMIZED USING VARIOUS CONSTRAINT HANDLING METHODS WITH MGSA SWARM. RESULT CLOSEST TO TRUE OPTIMAL VALUE (BEST) AND MEDIAN OF ALL RUNS FOR EACH FUNCTION ARE PRESENTED FOR EACH FUNCTION FROM 25 INDEPENDENT RUNS

	Function	Optimal	3S	Death	Static	Dyn	FD
Best	G1	-15.000	-15.000	-3.554	-15.000	-15.000	INF
	G2	-0.8036	-0.8036	-0.795	-0.8036	-0.8036	-0.7761
	G3	-1.0005	-0.9974	-0.1462	INF	-0.9996	INF
	G4	-30665.54	-30665.54	-30665.54	INF	-30665.54	-30248.29
	G5	5126.497	5127.780	INF	5131.045	5133.212	INF
	G6	-6961.814	-6961.813	-6864.455	INF	-6961.814	-6961.808
	G7	24.306	24.326	212.368	25.348	24.340	INF
	G8	-0.0958	-0.0958	-0.0958	INF	-0.0958	-0.0958
	G9	680.630	680.631	684.574	680.630	680.6308	684.356
	G10	7049.248	7092.875	11418.715	INF	INF	INF
	G11	0.7499	0.7499	0.7500	0.499	0.7499	0.9281
Median	G1	-15.000	-15.000	INF	-15.000	-15.000	INF
	G2	-0.8036	-0.7774	-0.7670	-0.7525	-0.7862	-0.7072
	G3	-1.0005	-0.6410	-0.0007	INF	-0.9949	INF
	G4	-30665.54	-30665.54	-30665.54	INF	INF	INF
	G5	5126.497	5271.40	INF	INF	INF	INF
	G6	-6961.814	-6961.813	-6164.350	INF	-6961.813	-6961.096
	G7	24.306	24.396	INF	24.635	24.692	INF
	G8	-0.0958	-0.0958	-0.0958	INF	-0.0958	-0.0958
	G9	680.630	680.632	699.053	680.631	680.636	INF
	G10	7049.248	7489.336	18590.46	INF	INF	INF
	G11	0.7499	0.7499	0.7502	0.7499	0.7499	INF

feasibility rate and optimization effectiveness, emphasising that the five requirements used to develop the algorithm have lead to a high performance optimization capability.

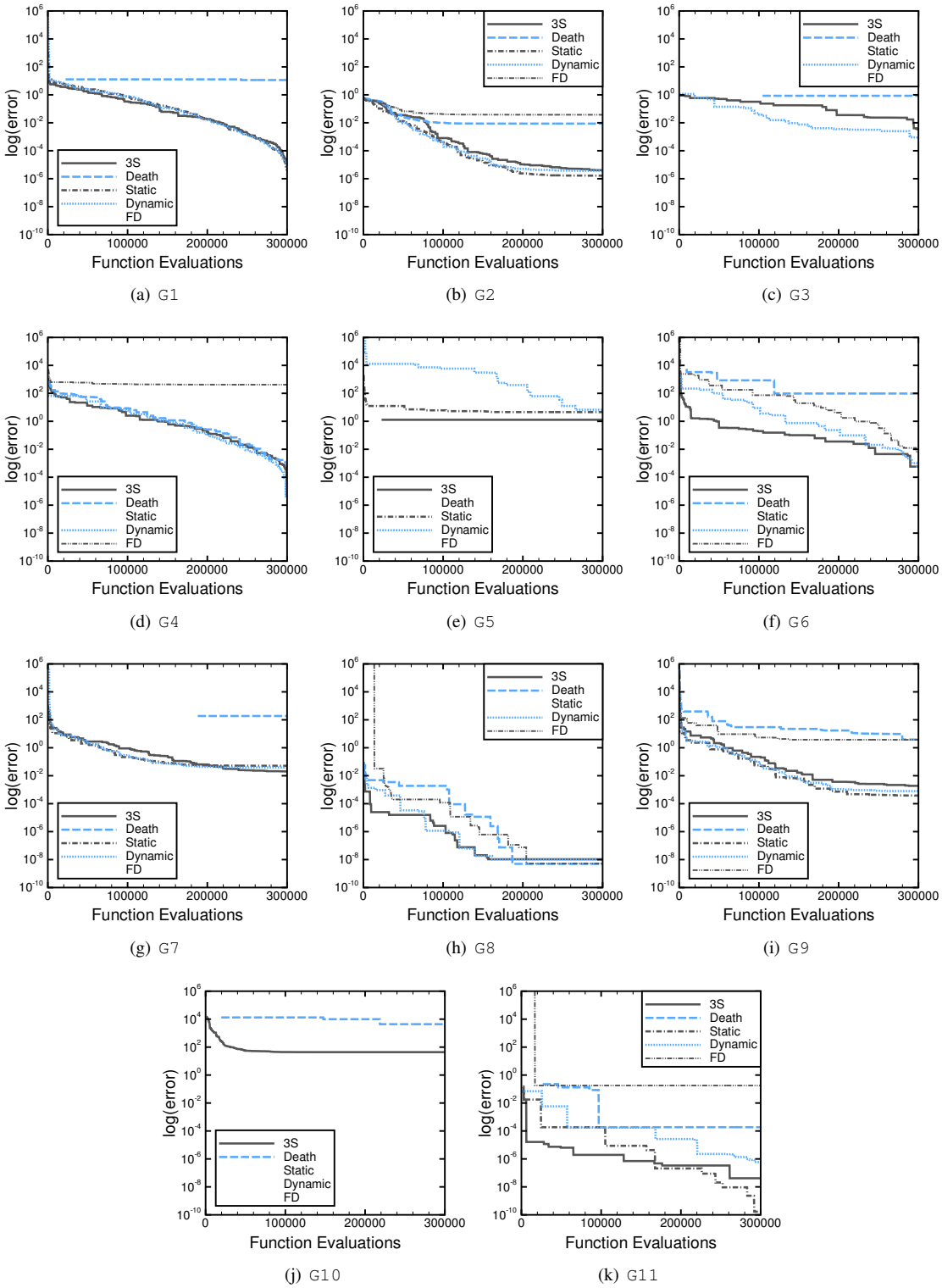
The method of feasible directions performed poorly, possible indicating that this method requires initialisation to result in at least one feasible solution such that the algorithm can point towards the feasible space. If this is not the case then the feasible particles will just point to where the least infeasible particle is found, which may not point it back towards the space. Problems which had equality constraints posed a particularly problem for the feasible direction approach, and problems that have small design spaces (which includes those with equality constraints) were not handled well as the algorithm had no knowledge of the feasible direction. The results could be improved by forcing the algorithm to reinitialise particles until at least one particle is feasible, though in problems with expensive objective function evaluations this is not a feasible approach.

In terms of finding a feasible solution, the 3S-MGSA algorithm is also highly dependable. In all but one of the test cases all of the 25 independent runs resulted in a feasible solution. The only case where a feasible solution was not found 100% of the time is G5, which is a difficult function to optimize due to having three equality constraints. The feasibility rate was 88% so the algorithm still performed well, even in this difficult problem, though further tuning of the parameters may result in improved results. This just demonstrates the problem that all agent-based search algorithms have which is that the performance and optimum parameter settings are highly case dependent.

The variability associated with using a penalty functions is reflected in the overall spread of data. The success rate of using the penalty function is also not as high as the 3S-MGSA algorithm with more infeasible solutions resulting from the 25 independent runs. The problem with the penalty approach is that the swarm has no knowledge of the quality of the infeasible search space compared to the feasible search space, instead the underlying search space is fundamentally altered by the use of the penalty. As a result, infeasible solutions are allowed under the penalty approach as the overall objective function can be considerable smaller in the infeasible region. This is, however, not possible in the 3S-MGSA approach as the infeasible and feasible regions are searched by swarms that have independent characteristics, though the infeasible swarm has knowledge of the feasible space by the use of the global and local best positions, which must be feasible if possible.

VI. CONCLUSIONS

This paper has considered various constraint handling techniques to allow the effective optimization of constrained problems by an efficient gravitational search algorithm (GSA). Much work has been performed on developing constraint handling systems for use with particle swarm optimization (PSO) such as penalty, mutation and separation approaches, though little work has developed constraint based systems for GSA, perhaps because the global data transfer that occurs within GSA poses some issues with coupling to traditional particle swarm constraint handling methods. GSA has been shown to be more effective at global



(a) G1 (b) G2 (c) G3

(d) G4 (e) G5 (f) G6

(g) G7 (h) G8 (i) G9

(j) G10 (k) G11

Fig. 1. Convergence of various constraint handling techniques applied to MGSA feasible swarm. Best solution from 25 independent runs is shown. If not feasible solution has been found then line is not shown.

optimization than PSO so the work here has compared standard constraint handling approaches with a novel separation-sub-swarm method.

The approach taken in the novel method here is to separate the infeasible and feasible particles into separate swarms to allow the independent optimization of the true objective function, while still allowing the population to search for a feasible solution. This also means that non-connected feasible regions can be handled too. The sub-swarming approach therefore leads to the separate swarms solving either the true

objective function or the constraints, though the infeasible sub-swarm cannot move under the rules of GSA as this produces a global transfer of data from the feasible to the infeasible region, instead moving by a PSO approach using feasibility rules to ensure the local and global best positions are feasible if possible. This also means that if initialisation produces infeasible particles, they can be handled within the existing framework without having to be reinitialised.

The separation-sub-swarmed (3S) constraint handling method for gravitational search algorithm was compared to different fidelity penalty functions and a feasible direction approach. Other separation, binary tournament and soft feasibility rule approaches cannot be implemented in GSA as GSA does not select leader particles so there is no need for feasibility rules, or would need manual manipulation of the masses of particles such as in a pure separation approach. The 3S method was shown to be highly efficient and effective at analytical constrained optimization compared to all the penalty approaches. A dynamic penalty outperforms other penalty approaches as the higher fidelity provides a more accurate penalty function to apply.

REFERENCES

[1] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders, "Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation," in *34th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 1996, AIAA Paper 1996–94.

[2] E. Panier and A. L. Tits, "On combining feasibility, descent and superlinear convergence in inequality constrained optimization," *Mathematical Programming*, vol. 59, pp. 261–276, 1993.

[3] A. M. Morris, C. B. Allen, and T. C. S. Rendall, "CFD–based optimization of aerofoils using radial basis functions for domain element parameterization and mesh deformation," *International Journal for Numerical Methods in Fluids*, vol. 58, no. 8, pp. 827–860, 2008.

[4] ——, "Domain–element method for aerodynamic shape optimization applied to a modern transport wing," *AIAA Journal*, vol. 47, no. 7, pp. 1647–1659, 2009.

[5] C. B. Allen and T. C. S. Rendall, "Computational-fluid-dynamics-based optimisation of hovering rotors using radial basis functions for shape parameterisation and mesh deformation," *Optimization and Engineering*, vol. 14, pp. 97–118, 2013.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[8] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *European Conference on Artificial Life*, Paris, France, 1991.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *1995 IEEE International Conference on Neural Networks*, Perth, Australia, 1995.

[10] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, pp. 459–471, 2007.

[11] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, pp. 2232–2248, 2009.

[12] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio–inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, pp. 4831–4845, 2012.

[13] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *2010 International Conference on Computer and Information Application (ICCIA)*, Tianjin, China, 2010.

[14] C. Li and J. Zhou, "Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm," *Energy Conversion and Management*, vol. 52, no. 1, pp. 374–381, 2011.

[15] H. C. Tsai, Y. Y. Tyan, Y. W. Wu, and Y. H. Lin, "Gravitational particle swarm," *Applied Mathematics and Computation*, vol. 219, no. 17, pp. 9106–9117, 2013.

[16] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Filter modelling using gravitational search algorithm," *Engineering Applications of Artifical Intelligence*, vol. 24, pp. 117–122, 2011.

[17] A. Chatterjee, G. K. Mahanti, and N. Pathak, "Comparative performance of gravitational search algorithm and modified particle swarm optimization algorithm for synthesis of thinned scanned concentric ring array antenna," *Progress in Electromagnetics Research B*, vol. 25, pp. 331–348, 2010.

[18] S. Mallick, S. P. Ghoshal, P. Acharjee, and S. S. Thakur, "Optimal static state estimation using improved particle swarm optimization and gravitational search algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 52, pp. 254–265, 2013.

[19] E. Mezura-Montes and C. A. Coello Coello, "Constraint handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, pp. 173–194, 2011.

[20] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Orlando, Florida, 2002.

[21] G. Venter and J. Sobieszczanski-Sobieski, "Particle swarm optimization," *AIAA Journal*, vol. 41, no. 8, pp. 1583–1589, 2003.

[22] G. Coath and S. K. Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *2003 IEEE Congress on Evolutionary Computation*, Canberra, Australia, 2003.

[23] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Euro-International Symposium on Computational Intelligence 2002*, 2002, pp. 214–220.

[24] G. N. Vanderplaats, *Numerical Optimization Techniques for Engineering Design*, 3rd ed. Vanderplaats Research and Development Inc., 1999.

[25] C. L. Sun, J. C. Zeng, and J. S. Pan, "An improved vector particle swarm optimization for constrained optimization problems," *Information Sciences*, vol. 181, pp. 1153–1163, 2011.

[26] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *2006 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.

[27] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, 2000.

[28] G. Toscano Pulido and C. A. Coello Coello, "A constraint handling mechanism for particle swarm optimization," in *2004 IEEE Congress on Evolutionary Computation*, Portland, Oregon, 2004.

[29] T. Takahama and S. Sakai, "Constrained optimization by the alpha constrained particle swarm optimizer," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 9, no. 3, pp. 282–289, 2005.

[30] ——, "Constrained optimization by the epsilon constrained particle swarm optimizer with epsilon-level control," *Advances in Soft Computing*, vol. 29, pp. 1019–1029, 2005.

[31] G. Venter and R. T. Haftka, "Constrained particle swarm optimization using a bi-objective formulation," *Structural Multidisciplinary Optimization*, vol. 40, pp. 65–76, 2010.

[32] M. Amoozegar and H. Nezamabadi-Pour, "Software performance optimization based on constrained GSA," in *16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*, Shiraz, Iran, 2012.

[33] S. Mondal, A. Bhattacharya, and S. Halder, "Solution of cost constrained emission dispatch problems considering wind power generation using gravitational search algorithm," in *2012 International Conference on Advances in Engineering, Science and Management (ICAESM)*, Nagapattinam, India, 2012.

[34] K. Pal, C. Saha, S. Das, and C. A. Coello Coello, "Dynamic constrained optimization with offspring repair based gravitational search algorithm," in *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 2013.

[35] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, pp. 1–32, 1996.