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

Constraint Handling in Agent-Based Optimization by Independent Sub-Swarms

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*—Agent-based optimization algorithms are an effective means of solving global optimization problems with design spaces containing multiple local minima, however, modifications have to be made to such algorithms to be able to solve constrained optimization problems. The gravitational search algorithm (GSA) is an efficient and effective agent-based method, however, the idea of global transfer of data that is key to the algorithm's success prohibits coupling of many state-of-the-art methods for handling constraints. Hence, a robust method, called separation-sub-swarm (3S) has been developed specifically for use with GSA by exploiting but also accommodating the global transfer of data that occurs in GSA, however it can also act as an entirely black-box module so is generally applicable. This newly developed 3S method has been shown to be efficient and effective at optimizing a suite of constrained analytical test functions using GSA.**

I. Introduction

OPTIMIZATION is the process of improving on a current solution. With continually increasing computer power available to the designers of engineering systems, automated algorithms are now a common approach to the problem of optimization. Aerodynamic shape optimization (ASO) is a rapidly expanding field which has historically been restricted by the expensive cost of objective function evaluations, which are often flow field solutions by computational fluid dynamics (CFD). Furthermore, the design space that describes many ASO problems often contains local optima and has constraints on solutions such as lift and volume, so presents a difficult problem for optimization algorithms to solve. State-of-the-art automated optimization algorithms have recently produced notable results for high fidelity two-dimensional [1], [2] and three-dimensional [3], [4], [5] ASO problems.

A common approach to solving an optimization problem, which is common within the field of ASO, is by the use of a gradient-based method where the local gradient is used as a basis about which to construct a search process. The most basic forms of this are the steepest descent and conjugate gradient methods, however these are designed for unconstrained optimization so require modification to allow the effective handling of constraints. More efficient approaches, which are typical in aerodynamic shape optimization, involve solving the Karush-Kuhn-Tucker (KKT) conditions. The most widely adopted approach is sequential-quadratic-programming [6], [7], 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 two-dimensional [8] and three-dimensional [9], [5] aerodynamic optimization problems.

Solving an ASO problem requires an algorithm that can enforce constraints in a strict manner, and minimizes the total number of objective function evaluations, which efficient gradient methods do. All gradient-based optimization algorithms, however, do have the same issues which are that a smooth and continuous design space is required, and the termination of the algorithm in local optima. The local optimization is a major issue, and can mean that for multimodal functions (functions with multiple 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 gradient-based approaches by avoiding the computation and use of the gradient, instead employing the position of the search in the search 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) [10], cooling in simulated annealing (SA) [11], ant colony food searching in the ant colony optimization (ACO) [12], swarm behaviour in particle swarm (PSO) [13], bee colony food searching in the artificial bee colony system (ABS) [14], Newtonian gravitational laws in gravitational search algorithm (GSA) [15] and hearding of krill in the krill heard framework (KH) [16]. The GA, SA and ACO algorithms are primarily designed for discrete optimization problems, 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

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

to a constrained problem, and this is an ongoing research area. This, and the higher cost associated with using global search algorithms, are the primary reasons for the slow uptake in fields with expensive objective function evaluations and the necessity for strict constraint enforcement, such as aerodynamic shape optimization. The objective of this paper is to introduce a new constrained optimization algorithm solution approach that is an entirely black-box method so applicable to any agent-based search algorithm, though was designed to exploit and handle the the gravitational search algorithm framework. The paper will outline approaches currently used to modify the search algorithms to allow feasible solutions to be found, followed by the novel approach developed here, and applications to analytical optimization algorithms.

II. Agent-based Search Algorithms

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: $|\mathbf{h}(\mathbf{x})| - \boldsymbol{\epsilon} \leq \mathbf{0}$.

There are several methods that exploit that use of a set of agents to search a design space. The original agent-based search system is particle swarm optimization, presented by Kennedy and Eberhart[13]. They pioneered the idea of using a set of particles to search a design space in pursuit of the global optimum. In its most basic form, a particle in a population of particles moves to a new position in time defined by some function of the objective function:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)\Delta t \tag{2}$$

Where the perturbation of a particle's position (which comes from its velocity, $\mathbf{v}_i$) is some function of the past and current state of the search.

A. Particle Swarm Optimization (PSO)

PSO uses knowledge of the cognitive (individual) and social (swarm) history of the search to construct a search procedure:

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_{1_i} \frac{p_i^d - x_i^d(t)}{\Delta t} + c_2 r_{2_i} \frac{s^d - x_i^d(t)}{\Delta t} \tag{3}$$

Where the superscript, d, is the design variable in the d-th dimension, and the subscript, i, is the variable for the i-th particle in a swarm of N particles. The random numbers, r_{1_i} and r_{2_i} add a stochastic nature to the algorithm and are randomly distributed between 0 and 1. The constants, c_1 and c_2, are the cognitive and social parameters respectively which give the local and global search extent of the scheme. To obtain a good balance between exploration and exploitation, the two factors required for an effective optimization, both values are often taken to be 2.0 such that the average of $c_1 r_{1_i}$ and $c_2 r_{2_i}$ is unity. An inertia weight, w, was added later by Shi and Eberhart [17] to improve the performance of PSO.

The simplicity of the basic PSO algorithm meant that researchers had to find innovative methods to improve its performance, however, this has resulted in many approaches where each is considered superior to the others (a review is presented by Poli *et al.* [18]). In fact, each approach for an improved particle swarm algorithm produces a new algorithm that is suited to certain classes of problems (and also individual problems) so it is difficult for a suitable algorithm to be selected for a specific real-world optimization problem without investing significant time into testing many flavours of PSO. The following section introduces a relatively new addition to the global search algorithm family called the gravitational search algorithm (GSA). Due to its infancy there are few extra additions to the basic algorithm so this avoids costly development and implementation time of the algorithm for a given optimization problem. The basic algorithm is also highly efficient and effective for many optimization problems, producing superior results to particle swarm algorithms.

B. Gravitational Search Algorithm (GSA)

The gravitational search algorithm (GSA) is an agent-based global search algorithm presented by Rashedi *et al.* [15] 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 acceleration of a particle, from Newton's second law of motion, is dependent on its mass and force, so the velocity is therefore:

$$v_i^d(t+1) = \text{rand}_i v_i^d(t) + a_i^d(t)\Delta t \quad (4)$$

A small increase in memory is required for GSA compared to PSO, however this is a small trade off to allow superior optimization ability: Rashedi *et al.* [15] reported that GSA outperformed PSO in over 80% of the analytical cases tested in their paper, and also for a real world optimization problem[19]; Chatterjee *et al.* [20] and Mallick *et al.*[21] 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 devoid 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 [22], 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' [23] 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 [24], though more efficient penalties are dynamic [25], so change depending on the current objective function [26].

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 [27]), 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[24]. Other approaches modify the velocity of a particle to point back towards the feasible space [28], 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 [29] 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 [30], [31] 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 [32] and ϵ-constrained [33] 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 [34].

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. Proposed Constrained Optimization Framework

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 [35] 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 [36], or by the repair method which initialises an infeasible particle near to the closest feasible particle [37]. 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.

A. Development of Constraint Handling Framework for GSA

From the consideration of the main approaches that have been presented by researchers to handle constraints using agent-based search systems, five requirements have been identified that need to be fulfilled to obtain an effective constraint handling technique for agent-based search algorithms:

1) Infeasible initialisation of particles permitted;
2) Optimization of true objective function in feasible region;
3) Global transfer of data in feasible space;
4) Unidirectional transfer of data from feasible to infeasible space;
5) Facilitation of particles 'pushed' back to feasible region.

The lack of contribution to the area of a constrained GSA algorithm maybe partly due to the characteristics that the algorithm has that lead to high efficiency, namely the global transfer of data throughout the swarm, though this addresses the third requirement immediately. To simultaneously satisfy requirements two, three, four and five a sub-swarm approach has been developed, where the primary swarm is split into two independent secondary sub-swarms; an infeasible swarm and a feasible swarm. The feasible swarm optimizes the objective function and moves by the GSA mechanism. The infeasible swarm minimises the constraint violation, which has a known minimum at the feasible region, such that there is some mechanism to drive the particles back to the feasible space. Using this separation of objective and constraints approach with the independent sub-swarms perfectly satisfies requirement two and also provides a mechanism to aid in the satisfaction of requirement five. An independent sub-swarm approach allows feasible transfer of data (requirement three), though at this point there is no mechanism for the handling of requirement four. Furthermore the infeasible swarm cannot move by the GSA mechanism as this would require a mass in the feasible region to propagate information from the feasible region outwards and that would violate requirement one.

The approach developed here is to move the infeasible swarm by a particle swarm mechanism, which is independent of the GSA mechanism of the feasible particles. The particle swarm approach requires storage of a particle's best position found so far and the swarm's best position found so far. To allow transfer of data from the feasible region to the infeasible particles (fourth requirement), a feasibility rules approach has been adopted such that the local and global best positions are always feasible if possible. If the swarm has never been feasible then these positions are the minimum constraint violation positions found so far. The implication of having feasibility rules is that if the feasible region has been found then information is always propagated into the infeasible region for the infeasible particles to work on, therefore satisfying requirement four. The transfer of information between sub-swarms and within the feasible swarm allows dispersion of information about the nature of the constraints, and the value of the objective function near to and far away from constraints facilitating fast convergence and competitive optimization behaviour. Finally, the only requirement not to have been considered so far is that initialization does not necessarily need to result in a single feasible particle, which is satisfied due to the use of feasibility rules. All five requirements for an effective constraint handling method have been addressed and met using this approach.

A final note regarding the search mechanism of the feasible swarm is worth making. The use of a particle swarm-based approach for solving the infeasible problem requires extra storage of the best position of every particle and the swarm's best position, as well as the objective function values associated with that. To ensure that the best ever positions are feasible, a feasibility rules-based approach is adopted and this ensures that if the swarm has ever been feasible that the best position is also feasible. As this additional storage is required anyway it is more efficient to introduce the particle swarm mechanism into the feasible search as well to produce a hybrid GSA-PSO mechanism for the feasible search which is termed modified-GSA (MGSA), which has shown to be an efficient search mechanism [38], [39], [40].

B. Separation-Sub-Swarm Mechanism

The algorithm developed for solving constrained global optimization algorithms using an independent sub-swarm approach, where the secondary sub-swarms solve a separated objective and constraints function, is hereby called the separation sub-swarm algorithm (3S). It is a black-box approach so any swarm can be used to search the feasible space, so the algorithm is presented here independent of the swarming method for the feasible region.

For each agent, the objective function and the values of the constraints need to be calculated to determine feasibility; a particle is termed infeasible if any of the constraints are violated. The objective function associated with the i-th agent is then:

$$f^p(\mathbf{x}_i) = \begin{cases} f(\mathbf{x}_i) & \text{if } \mathbf{x_i} \text{ is feasible} \\ \sum_{j=1}^{G} \max\{0, g_j(\mathbf{x}_i)\} + & \text{else} \\ \sum_{j=1}^{H} |h_j(\mathbf{x}_i)| & \end{cases} \tag{5}$$

The particle's best ever position, $\mathbf{p}_i$, and the swarm's best ever position, $\mathbf{s}$, need to be updated, 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.

The acceleration of the infeasible particles is done by particle swarm:

$$a_i^d(t) = c_1 r_{1_i}(p_i^d - x_i^d(t)) + c_2 r_{2_i}(s^d - x_i^d(t)) \tag{6}$$

where $\mathbf{p}_i$ and $\mathbf{s}$ are the particle's and swarm's best positions ever, which are always feasible if historically a feasible point has been found either cognitively or socially.

C. Coupling 3S with a GSA-based approach

The 3S approach for handling constraints treats the infeasible and feasible swarms independently so is applicable to any agent-based search algorithm. The approach is designed to specifically handle global transfer of data that occurs within the gravitational search algorithm, which is handled by the use of independent swarms. Here, a modified GSA (MGSA) algorithm is described which incorporates the particle swarm movement into the process, and this is used for testing with the 3S framework.

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

$$\mathbf{x_i} = \{x_i^1, x_i^2, \ldots, x_i^d\}^T \tag{7}$$

For the N_f feasible particles only, the mass is calculated based on a particle's feasible fitness:

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \tag{8}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N_f} m_j(y)} \tag{9}$$

where the best and worst fitnesses are from the feasible particles only. The force acting on particle i from particle j, where both particles are feasible is:

$$F_{i,j}^d(t) = G(t)\left(\frac{M_i(t)M_j(t)}{R_{i,j}(t) + \epsilon}\right)(x_j^d(t) - x_i^d(t)) \tag{10}$$

$$G(t) = G_0 \exp(-\alpha t/T) \tag{11}$$

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

$$F_i^d(t) = \sum_{j=1, j\neq i}^{\min\{N_f, Kbest\}} \text{rand}_j F_{i,j}^d(t) \tag{12}$$

where $Kbest$ is a constant that linearly decreases from N at the start of the optimization to 1 at the end, and controls the effect that poorer acting particles have on good particles. The acceleration of the feasible particles is by MGSA (equation 15), though this could also be just an individual GSA or PSO acceleration as well. If only one particle is feasible then the acceleration due to GSA to that particle becomes zero.

$$a_i^d(t)_{gsa} = F_i^d(t)/M_i(t) \tag{13}$$

$$a_i^d(t)_{pso} = c_1 r_{1_i}(p_i^d - x_i^d(t)) + c_2 r_{2_i}(s^d - x_i^d(t)) \tag{14}$$

$$a_i^d(t) = (a_i^d(t)_{gsa} + a_i^d(t)_{pso})/2 \tag{15}$$

where $\mathbf{p}_i$ and $\mathbf{s}$ are the particle's and swarm's best positions ever, which are always feasible in this part of the optimization process. The cognitive, c_1, and social, c_2, parameters do add additional parameters to the problem but it is expected that these have the same value as the cognitive and social parameters is the infeasible search (equation 6).

The updating procedure for all of the particles is:

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

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{17}$$

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

V. Constrained Analytical Optimization

The performance and efficiency of the 3S algorithm is analysed here. An analytical function suite, as outlined by Michalewicz and Schoenauer [41], is commonly used to test constrained global optimization algorithms. 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: $|h_j(\mathbf{x})| - \epsilon \leq 0$.

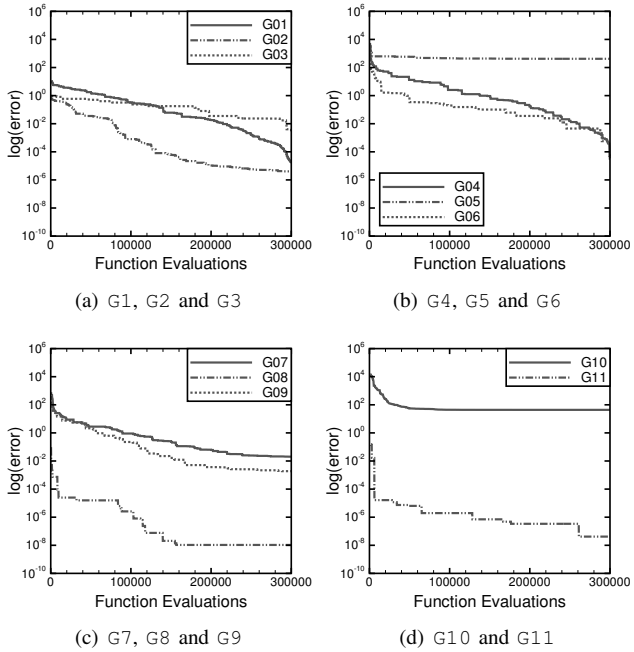
25 independent runs of each of the test functions were performed. 200 particles were used; maximum number of timesteps at 1500; $G_0 = 30$; $\alpha = 10$; $c_1 = c_2 = 2$. The degree of violation for the equality constraints was $\epsilon = 0.0001$, so as a result optimum solutions better than the theoretical optimum were possible. The objective function evaluations

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

have been parallelised using the message passing interface (MPI) to allow more efficient algorithm performance.

From the 25 independent runs using the 3S algorithm developed in this paper combined with the MGSA swarm, the best, median and worst results are presented in table II, as well as the mean of all the feasible results and the standard deviation. If an infeasible solution results from any of the runs then the worst value is presented as INF, meaning infeasible, though the remaining statistical values are all from feasible solutions if possible. The convergence of the best results are shown in figure 1.



(a) G1, G2 and G3 (b) G4, G5 and G6

(c) G7, G8 and G9 (d) G10 and G11

Fig. 1. Convergence plots of function error for 11 test cases using 3S-MGSA. FES represents number of objective function evaluations.

The results from table II show that the 3S-MGSA algorithm has successfully located the global optimum in all the eleven test cases. In cases G1, G2, G4, G6, G8, G9 and G11 the algorithm has located the global optimum exactly, emphasising the efficiency that the algorithm has in terms of optimum exploration and exploitation. In the remaining cases, the global optimum position has been located though the exact exploitation of the absolute optima would require more specific tuning of the parameters, or use of a gradient-based optimization system to accurately exploit the global optimum. The efficiency of the algorithm is reflected by the small spread of results, with the average standard deviation normalised by the mean being 3%. These problems are characterised by highly multimodal search spaces with multiple non-linear and linear constraints, so the small spread of data for all the cases is encouraging. Further tuning of parameters for each problem would result in a lower spread of data but it was decided to have parameters that were constant for all the test cases as this more realistically depicts the difficulty that maybe presented for optimization of an expensive objective function, where parameter tuning would be a time consuming process. This has, however, increased the difficulty of the problem though the 3S-GSA algorithm has been extremely successful.

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 of G5 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 overall feasibility rate was 99% showing that a feasible solution is almost always found.

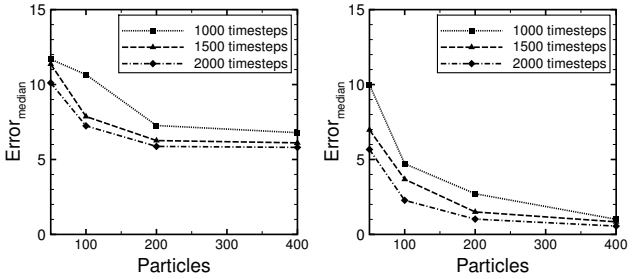
VI. PARAMETER SENSITIVITY

The separation-sub-swarm constraint handling method has been shown to be an effective method for solving constrained optimization problems using a GSA type mechanism. The 3S approach was designed specifically for GSA, though the method can also be considered a black-box approach and as such can be wrapped around any agent-based search algorithm. In the results presented above, a hybrid GSA/PSO swarm was used as the global transfer of data that is present in GSA and the memory qualities present in PSO are both exploited, though pure GSA can also be used with the 3S algorithm, and this is explored here. Furthermore, the population size and total number of iterations are often considered important parameters as they have a direct correlation on the total cost of running the algorithm and can often influence greatly its performance. The 3S algorithm applied to the two search methods has been run on the analytical test suite 25 times each for various numbers of particles and various timesteps. Figure 2 shows the effect of varying particle number for constant number of iterations using the three feasible search mechanisms. The vertical axis is the measure

TABLE II
RESULTS OF 11 ANALYTICAL FUNCTION TEST SUITE OPTIMIZED USING THE 3S-MGSA ALGORITHM.

Function	Optimal	Best	Median	Worst	Mean	St. Dev.	SD/Opt
G1	-15.000	-15.000	-15.000	-12.453	-14.432	1.13	7.55%
G2	-0.80362	-0.80362	-0.77744	-0.68978	-0.77239	2.98×10^{-2}	3.71%
G3	-1.0005	-0.99124	-0.94100	-0.65084	-0.89597	9.61×10^{-2}	9.61%
G4	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	1.09×10^{-4}	3.59×10^{-9}%
G5	5126.50	5126.51	5271.40	INF	5355.2886	214.24	4.18%
G6	-6961.81	-6961.81	-6961.81	-6961.80	-6961.81	4.50×10^{-3}	6.46×10^{-5}%
G7	24.306	24.313	24.396	24.712	24.441	0.11	4.47×10^{-1}%
G8	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	5.07×10^{-15}	5.21×10^{-12}%
G9	680.630	680.630	680.632	680.648	680.633	3.53×10^{-3}	5.19×10^{-4}%
G10	7049.248	7077.759	7489.336	8465.689	7518.675	280.12	3.97%
G11	0.750	0.750	0.750	0.824	0.756	1.61×10^{-2}	2.16%

of the average error of the median run for all functions and all runs.



(a) 3S-GSA (b) 3S-MGSA

Fig. 2. Effect of varying particle number and evolutions on mean percentage error of median solutions from 25 independent test suite runs using GSA and MGSA.

The results indicate, almost exclusively, that larger numbers of particles and larger numbers of timesteps lead to better optimization results. This is generally expected as a larger population can search the space in considerably more detail and therefore provide superior exploration abilities. The exact exploitation of the globally optimal solution is best provided by the MGSA feasible swarm which makes use of the global data transfer of GSA and the memory qualities of PSO to provide a highly efficient feasible search mechanism. Furthermore, by using the MGSA algorithm, a blending of the velocity exists between the feasible and infeasible region such that the PSO component of velocity exists in both swarm mechanisms to aid in smooth transfer between the two swarms.

The use of global search algorithms within optimization frameworks containing expensive objective function evaluations can pose runtime and cost problems particularly if the total number of evolutions is large. The effect of particle number on cost can be somewhat mitigated by the spatial parallelisation where each processor is assigned a specific particle, hence the major factor affecting wall-time is the number of evolutions. To minimise the total wall time, the number of evolutions should be kept to a minimum though this restricts the amount of exploration and the degree of exploitation that can occur, as is shown in figure 2, though in general the use of 1500 evolutions provides much superior results to using 1000 evolutions. The use of 2000 evolutions improves the results further but the ratio of improvement to cost is lower. A compromise can therefore be made by using 1500 evolutions. Furthermore, the use of more particles can improve the optimization capability, though again there exists a point which represents the optimum ratio of improvement to cost and is around 200. The combination of 1500 evolutions using 200 particles therefore appears to be the best compromise between cost and optimization ability.

VII. CONCLUSIONS

This paper has presented a novel constraint handling technique to allow the effective optimization of constrained problems by the 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 constrained based systems for GSA. GSA has been shown to be more effective at global optimization than PSO so the work here introduces a novel approach that is generic for any continuous optimization problem.

The approach taken 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 entire population to search for a feasible solution. This also means that non-connected feasible regions can be handled. 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, instead moving by a particle swarm 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 gravitational search algorithm (3S-GSA) was tested on a suite of analytical test cases

and found to be efficient at finding the global optimum in all tests. The exact exploitation of the global optimum was found in many of the cases and overall the performance was good.

REFERENCES

[1] H. P. Buckley, B. Y. Zhou, and D. W. Zingg, "Airfoil optimization using practical aerodynamic design requirements," *Journal of Aircraft*, vol. 47, no. 5, pp. 1707–1719, 2010.

[2] X. Han and D. W. Zingg, "An adaptive geometry parameterization for aerodynamic shape optimization," *Optimization and Engineering*, 2013, published online 31st January.

[3] H. S. Chung and J. J. Alonso, "Multiobjective optimization using approximation model-based genetic algorithms," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, 2004, AIAA Paper 2004–4325.

[4] W. S. Wong, A. Le Moigne, and N. Qin, "Parallel adjoint–based optimisation of a blended wing body aircraft with shock control bumps," *The Aeronautical Journal*, vol. 111, no. 1117, pp. 165–174, 2006.

[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] 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.

[7] J. L. Zhou and A. L. Tits, "Nonmonotone line search for minimax problems," *Journal of Optimization Theory and Applications*, vol. 76, no. 3, pp. 455–476, 1993.

[8] 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.

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

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

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

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

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

[14] 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.

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

[16] 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.

[17] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computing*, Anchorage, Alaska, 1998.

[18] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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