Non-dominated Sorting Gravitational Search Algorithm

Hadi Nobahari[1], Mahdi Nikusokhan[1], Patrick Siarry[2]

[1] Department of Aerospace Engineering,
Sharif University of Technology, P.O. Box: 11365-11155, Tehran, Iran.
nobahari@sharif.edu, nikusokhan@ae.sharif.edu

[2] Université Paris-Est Créteil (UPEC)
61 av. du Général de Gaulle, 94010 CRETEIL Cedex, France
siarry@u-pec.fr

Abstract

This paper proposes an extension of the Gravitational Search Algorithm (GSA) to multi-objective optimization problems. The new algorithm, called Non-dominated Sorting GSA (NSGSA), utilizes the non-dominated sorting concept to update the gravitational acceleration of the particles. An external archive is also used to store the Pareto optimal solutions and to provide some elitism. It also guides the search toward the non-crowding and the extreme regions of the Pareto front. A new criterion is proposed to update the external archive and two new mutation (turbulence) operators, called sign and reordering mutations, are also proposed to promote the diversity within the swarm. Numerical results show that NSGSA can obtain comparable and even better performances as compared to the previous multi-objective variant of GSA and some other multi-objective optimization algorithms.

Key words

Gravitational Search Algorithm, Multi-objective Optimization, Non-dominated Sorting, Sign Mutation, Reordering Mutation.

1 Introduction

Over the last decades, several meta-heuristics have been developed to solve complex single and multi-objective optimization problems. In multi-objective optimization problems, there is no single optimal solution, but rather a set of non-dominated solutions, also called Pareto-optimal solutions. The use of meta-heuristics to solve multi-objective optimization problems are growing fast, because they can handle problems with concave and disconnected Pareto fronts.

There are many meta-heuristics such as Simulated Annealing (SA) [9], Tabu Search (TS) [4-5], Evolutionary Algorithms (EAs) [10], Ant Colony Optimization (ACO) [3], Particle Swarm Optimization (PSO) [8], Gravitational Search Algorithm (GSA) [11] and so on. These algorithms are able to solve different optimization problems. However, there is no specific algorithm able to find the best solutions of all problems in finite iterations and some algorithms show better performances for particular problems than the others. Although both single and multi-agent metaheuristics have their own multi-objective variants, the multi-agent meta-heuristics such as EAs, ACO and PSO have shown better potential to solve multi-objective optimization problems, because in a multi-objective problem, a population of solutions is going to be generated, in a single run.

GSA is a new multi-agent optimization algorithm, inspired from the general gravitational law [11]. The algorithm is based on the movement of some particles under the effect of the gravitational forces, applied by the others. In Ref. [7] the first multi-objective variant of GSA, called Multi-Objective GSA (MOGSA), was proposed. MOGSA uses an external archive to store the non-dominated solutions and

uses the same idea as in Simple Multi-Objective PSO (SMOPSO), proposed in Ref. [1], to update the external archive. For this purpose, the interval of each objective function is divided into equal divisions and consequently the entire performance space is divided into hyper-rectangles. When the number of archived members exceeds the maximum archive length, one member of the most crowded hyper-rectangle is randomly chosen and removed.

The main problem to propose the multi-objective variant of GSA is to update the mass of particles based on the value of the multiple objectives. In MOGSA, the mass of all moving particles is set to one and the mass of archived particles is updated based on the distance from the nearest neighbor, within the objective space, although no equation has been presented that relates the mass value to the distance value. This technique distributes the archived elements uniformly, similar to the niching technique. After calculating the mass of the moving particles, they move to the new positions by the forces applied from the archived members. In MOGSA, only the archived particles apply gravitational forces to the moving ones and after each movement, the well-known uniform mutation is applied to the new positions.

In this paper, the authors have proposed a new multi-objective variant of GSA, called NSGSA. In single-objective GSA, proposed in [11], the mass of each particle is taken to be proportional to its fitness. As mentioned, the main problem to propose a multi-objective GSA is to establish a relationship between the mass of each particle and its multiple objectives. In this paper, the non-dominated sorting concept, proposed in Non-dominated Sorting GA (NSGA) [12], is used to divide the particles to several layers within the performance space. In this way, the mass of each particle will depend to the rank of the layer it belongs to. NSGSA utilizes a limited length external archive to store the last found non-dominated solutions. To provide some elitism, specific members of the external archive are added to the list of moving particles and contribute in applying gravitational forces to the others. A new criterion is introduced to update the external archive, based on a new defined spread indicator. The mutation (turbulence) phenomenon is also modeled in NSGSA, not considered in the original GSA. In this regard, in addition to the uniform mutation, two new mutation operators, called sign and reordering mutations, are also proposed.

The organization of the paper is as follows: In section 2 GSA is described in details. Thereafter, in section 3, the new multi-objective variant of GSA, called NSGSA, is introduced. Section 4 presents numerical results. The results of NSGSA are compared with those of MOGSA, SMOPSO and NSGA-II [2] for several multi-objective benchmarks. Finally, the conclusions are outlined in section 0.

2 Gravitational Search Algorithm

GSA has been inspired from the mass interactions based on the general gravitational law. In the proposed algorithm, the search agents are a collection of masses, interact with each other based on the Newtonian laws on gravity and motion. Also, it uses some stochastic operators to increase diversity and the probability of finding the global optimum [11]. In the following, the formulation of GSA is presented in short. The mass of each particle is calculated according to its fitness value, as follows:

$$\begin{aligned} m_i(t) &= \frac{\mathrm{fit}_i(t) - \mathrm{worst}(t)}{\mathrm{best}(t) - \mathrm{worst}(t)} \quad i = 1, 2, ..., \mathrm{n}_{\mathrm{mass}} \\ M_i(t) &= \frac{m_i(t)}{\sum_{j=1}^{\mathrm{n}_{\mathrm{mass}}} m_j(t)} \qquad 0 \le M_i(t) < 1 \end{aligned} \tag{1}$$

where t is time (iteration), fit_i presents fitness value of the i-th particle, M_i is the normalized mass, $\mathrm{n}_{\mathrm{mass}}$ is the number of particles, and worst(t) and best(t) are defined for a minimization problem as follows:

$$\text{worst}(t) = \max_{i=1}^{n_{mass}}\{\text{fit}_i(t)\} \quad , \quad \text{best}(t) = \min_{i=1}^{n_{mass}}\{\text{fit}_i(t)\} \tag{2}$$

The vector of force, applied by mass *j* to mass *i*, $\mathbf{f}_{ij}(t)$, is calculated as follows:

$$\mathbf{f}_{ij}(t) = G(t)\frac{M_i(t)M_j(t)}{R_{ij}(t)+\varepsilon}(\mathbf{x}_j - \mathbf{x}_i) \tag{3}$$

where $\mathbf{x}_i$ is the position vector of the *i*-th agent, ε is a small threshold, and $G(t)$ is the gravitational constant, initialized at the beginning of the algorithm and is reduced with time to control the search accuracy. $R_{ij}(t)$ is the Euclidian distance between two agents *i* and *j*. Using the 2nd Newton law of motion, the total gravitational acceleration of the *i*-th agent, can be calculated as follows:

$$\mathbf{a}_i(t) = G(t)\sum_{j=1}^{n_{mass}} rand_j \frac{M_j(t)}{R_{ij}(t)+\varepsilon}(\mathbf{x}_j - \mathbf{x}_i) \tag{4}$$

where $rand_j$ is a uniform random number within the interval [0,1], considered to add some stochastic behavior to the acceleration. Now, the velocity and position of the agents are updated as follows [11]:

$$\begin{aligned} \mathbf{v}_i(t+1) &= rand'_i\,\mathbf{v}_i(t) + \mathbf{a}_i(t) \\ \mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \end{aligned} \tag{5}$$

where $rand'_i$ is another uniform random number in the interval [0,1]. To prevent the particles go out of the search space, the positions are bounded to the limits of each variable. To perform a good compromise between exploration and exploitation, Ref. [11] has proposed to reduce the number of attractive agents in Eq. (4) with lapse of time. Therefore, in GSA only the $k_{best}(t)$ of the moving agents will attract the others. $k_{best}(t)$ is a decreasing function of time, with the initial value $k_0=n_{mass}$. At the end there will be just one agent applying force to the others.

3 Non-dominated Sorting Gravitational Search Algorithm

In this section the multi-objective variant of GSA, called NSGSA, is introduced. The new algorithm utilizes the concept of Pareto optimality condition and non-dominance. A high level description of NSGSA is presented in section 3.1, details of which are provided in later sections.

3.1 General setting out of the algorithm

Fig. 1 shows the general iterative steps of NSGSA in the form of a pseudo code. The main loop starts after initializing the particle positions and velocities. A limited length external archive of the Pareto optimal solutions is generated in the first iteration. In later iterations, the updated particles are compared with the members of the external archive and the archive is updated based on the Pareto optimality condition. When the external archive is completed and a new member is going to be added, a spread indicator is utilized to decide which solution to be replaced. The moving particles are then ranked in layers using the non-dominated sorting algorithm [12]. To add some elitism to the algorithm and to promote the uniform distribution of the solutions, some specific members of the external archive are added to the list of moving particles and instead the same numbers of the moving particles are discarded from the worst layer(s). In next step the non-dominated sorting algorithm is utilized to assign each particle a rank, regarding the layer it belongs to. Then, rank of each particle is considered as its fitness and the mass is calculated using Eq. (1).

The accelerations, exerted to the particles, are calculated using Eq. (4). Then the velocity and consequently the position of particles are updated using Eq. (5). The position of particles are also mutated to increase the diversity of the algorithm and to prevent premature convergences. The mutation strength depends on the velocity of particles. As the gravitational constant and consequently the velocity of particles are decreased by lapse of time, the mutation strength is decreased, too. Therefore, exploration is decreased through the iterations while at the same time exploitation increases and helps to find the final solutions accurately. In the following subsections, the steps, bolded in Fig. 1 are discussed in details.

```
Initialization()
t=1
While t<tmax
    Evaluate Fitness of each particle
    Update external archive()
    Non-dominated sorting()
    Update the list of moving particles()
    Update the mass of moving particles()
    Update particles acceleration()
    Update particles velocity()
    Update and Mutate particles position()
    t=t+1
EndWhile
Report Pareto optimal solutions stored in archive
```

Fig. 1. Pseudo-code of NSGSA

3.2 Initialization

The new algorithm has 8 control parameters, defined in the next subsections, which must be set before the execution of the algorithm. An empty external archive is generated. Initial position of particles is set randomly within the search intervals and the initial velocities are set to zero for all particles.

3.3 Updating the External Archive

The external archive is updated based on the Pareto dominance criterion. If a member of the swarm is dominated by any member of the external archive, it will not be inserted to the archive. In contrast, if it is non-dominated with respect to all members of the archive or it dominates some members of the archive, it will be inserted to the archive and the dominated member(s) will be removed. If the number of archived members exceeds the maximum archive length, the most crowded area must be determined to remove a member from that area. A spread indicator is introduced in this paper to control the length of external archive. This indicator is based on the spread of the points within the Pareto front. Here, we are interested in getting a set of solutions that spans the entire Pareto optimal region as uniformly as possible. For this purpose, a crowding distance measure, $d_{c,i}$, similar to that proposed in Ref. [2], is defined as follows:

$$d_{c,i} = \sqrt{\sum_{j=1}^{m} \left(d_{c,i}^{j}\right)^2} \qquad (i = 1,..,n_{archive}) \tag{6}$$

Where $d_{c,i}^{j} = fit_{i+1}^{j} - fit_{i-1}^{j}$. fit_{i+1}^{j} is the j-th fitness of the point i+1 when the members of the archive are sorted versus the j-th objective function. To have nearly equal $d_{c,i}$, the deviation of these distances

from their average, d_c, should be minimized. For this purpose a spread indicator, δ, is defined as follows:

$$\delta = \frac{\sum_{\substack{i=1 \\ i \notin E}}^{n_{archive}} \left| d_{c,i} - d_c \right|}{[n_{archive} - m] d_c} \tag{7}$$

where $n_{archive}$ is the archive length and E is the set of extreme points, the size of which is taken equal to m. Ref. [2] uses a similar equation for measuring the performance of the algorithms, called as spread metric. To obtain a uniform spread of Pareto archive, δ must be decreased. In order to decrease δ, the difference between $\max\{d_{c,i}\}$ and $\min\{d_{c,i}\}$ must be decreased (i=1,…, $n_{archive}$). In other words, the maximum $d_{c,i}$ must be decreased or the minimum $d_{c,i}$ must be increased. Inserting a new member to the archive decreases the maximum $d_{c,i}$ or at least does not change it, but increasing the minimum value is only possible through removing a suitable member. For this purpose, the two nearest neighbors in the objective space, are determined and named as members 1 and 2. These two members are removed alternatively to calculate δ_1 and δ_2 corresponding to the removal of members 1 and 2, respectively. The member, the presence of which causes bigger δ is then removed, unless it extends the Pareto front, the case of which the other member should be removed.

3.4 Updating the list of Moving Particles

During any iteration, some members of the external archive should be inserted to the list of moving particles for several reasons: First, insertion of such members adds some elitism to the algorithm. Next, if the extreme points of the external archive are also inserted, then the algorithm will hopefully be able to extend the Pareto front. Finally, insertion of the members, located in the least crowded area, can repair the search gaps within the Pareto front.

The list of moving particles for the next flight is updated as follows: m extreme points of the Pareto front (m single-objective optimal solutions of the external archive) are inserted to the list. Then, m points, located in the least crowded area, are inserted. These points are selected using the crowding distance assignment algorithm, proposed in [2]. A percent of the archive members, $P_{elitism}$, are also selected randomly and inserted to the list of moving particles. Finally, the list length is limited to n_{mass} by removing some of the worst particles, starting from the worst layer of the non-dominated sorted particles. It should be noted that the initial velocity of the members, inserted from the external archive to the list of moving particles, must be set to zero.

3.5 Updating the Mass of Moving Particles

NSGSA utilizes the ranks of the layers, generated using the non-dominated sorting algorithm to determine the fitness of each particle. Then by using Eq. (1), the mass of each particle is calculated. The rank of moving particles is calculated as follows: the m members, imported from the extreme regions of the external archive and the m members, imported from the least crowded area of this archive are ranked as 1, as depicted in Fig. 2. The $P_{elitism}$ percent of the archive members, chosen randomly and inserted to the list of moving particles are ranked as 2. The members, remained from the last iteration, get ranks 3 and more, regarding the rank of the layers they belong to. The rank of each particle is then considered as its fitness and its mass is updated using Eq. (1).

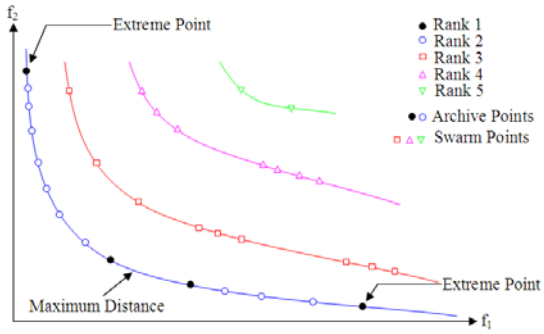

Fig. 2. Non-dominated sorting and ranking

3.6 Updating the acceleration of particles

NSGSA uses the same equation as GSA to update the acceleration of particles. The most important parameter of this equation is the gravitational constant, $G(t)$. In Ref. [11], $G(t)$ is suggested as $G_0 \exp(-\alpha t/t_{max})$. Where α and G_0 are the function parameters. It is difficult to find a suitable constant G_0 for various test functions. In this paper, G_0 is proposed as follows:

$$G_0 = \beta \max_{d\in\{1,..,n\}} (|x_u^d - x_l^d|) \tag{8}$$

where β is a parameter of NSGSA. The quick decrease of $G(t)$, proposed in Ref. [11], causes a fast decay of the exploration. To improve this problem NSGSA utilizes the linear function $G(t)= G_0 (1-t/t_{max})$.

3.7 The Use of Mutation Operator

To the authors' experience, the original GSA suffers from the premature convergence in the case of complex multimodal problems. To solve this problem, a mutation operator is added to the original GSA to decrease the possibility of trapping in local optima. In reality, the close moving particles may impact to each other and consequently the direction of their movement may change. Therefore, to simulate the irregularities, exist in the movement of the real particles, two new mutation operators have been proposed, called sign and reordering mutation. NSGSA utilizes a combination of these two mutation operators and the well-known uniform mutation as its mutation (turbulence) operator. The new mutation operators, proposed by the authors, are defined in the following.

Sign Mutation. In sign mutation, to update the position of each particle, the sign of velocity vector changes temporally, with a predefined probability, P_s, as follows:

$$\begin{aligned} v_i'^d(t+1) &= s_i^d\, v_i^d(t+1) \qquad (d=1,..,n,\ i=1,..,n_{mass}) \\ s_i^d &= \begin{cases} -1 & rand < P_s \\ 1 & \text{otherwise} \end{cases} \\ \mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \mathbf{v}_i'(t+1) \end{aligned} \tag{9}$$

where $v_i'^d(t+1)$ is the mutated velocity by the sign mutation operator and *rand* is a uniform random number, generated in the interval [0,1]. In Ref. [6] a similar mutation operator is proposed. But, there are differences between these two mutations. The sign mutation, proposed in Ref. [6], changes the sign of $v_i^d(t)$; whereas, the sign mutation, proposed here, mutate the position of the particle, by temporally changing the sign of $v_i^d(t+1)$ to update the position. Then, in the later iteration, the non mutated velocity, $v_i^d(t+1)$, is used to calculate $v_i^d(t+2)$.

Reordering Mutation. In this mutation some particles are chosen randomly to be mutated according to reordering mutation probability (P_r). Then, elements of the velocity vector are rearranged randomly. In Fig. 3 different possible mutations of a velocity vector are shown for a two dimension space. As can be seen, when the sign and the reordering mutations are applied together, the exploration space is increased with respect to the case when only one of them is applied.

3.8 Update and mutate the position of particles

NSGSA utilizes a combination of sign, reordering and the well-known uniform mutations to update the position of particles. The mutation operator of NSGSA works as follows: the velocity of particle is first mutated by the sign and then by the reordering mutations. After updating the positions by the mutated velocity vectors, the uniform mutation is applied on the updated position, by a predefined probability, P_u. Therefore, in NSGSA the position of particles is updated as follows:

$$\begin{aligned}
&\mathbf{v}_i(t+1) = w(t) \times \mathbf{v}_i(t) + \mathbf{a}_i(t) \\
&\mathbf{v}'_i(t+1) = \textbf{Sign_Mutate}(\mathbf{v}_i(t+1)) \\
&\mathbf{v}''_i(t+1) = \textbf{Reordering_Mutate}(\mathbf{v}'_i(t+1)) \\
&\mathbf{x}'_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}''_i(t+1) \\
&\mathbf{x}_i(t+1) = \textbf{Uniform_Mutate}(\mathbf{x}'_i(t+1))
\end{aligned} \tag{10}$$

where $w(t)$ is the time varying inertia coefficient. In original GSA, this coefficient is taken a random number between 0 and 1. As in PSO, it is better to use a decreasing inertia weight than a random one to have proper exploration and exploitation in the first and the last iterations, respectively. NSGSA utilizes a time varying weighting coefficient $w(t)=w_0-(w_0- w_1)t/t_{max}$. It should be noted that the mutation operator of NSGSA is applied to the position vector not the velocity vector.

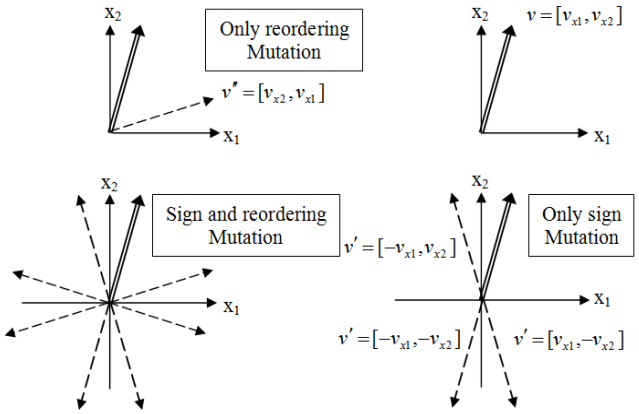

Fig. 3. Sign and reordering mutation operators and their combination

4 Numerical Results

To evaluate the performance of NSGSA, the optimization results, obtained for a set of standard benchmarks, are compared with the well-known NSGA-II, MOGSA and SMOPSO. Ref. [2] have used nine benchmark problems, known as SCH, FON, POL, KUR, ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, to evaluate the performance of NSGA-II. To compare the results of NSGSA with those of NSGA-II, the same benchmarks are utilized here. Ref. [1] and [7] has used three benchmarks, known as Viennet3 (MOP5), Deb (MOP6) and Binh2 (MOPC1), to evaluate the performance of MOGSA and

SMOPSO, respectively. These benchmarks are also utilized here, to compare the performance of NSGSA with that of MOGSA and SMOPSO.

Some metrics are used to measure the performance of multi-objective algorithms, the examples of which can be found in Ref. [1] and [2]. In this study, the two metrics, defined in Ref. [2], are used to compare the performance of NSGSA with NSGA-II. The first metric (γ), measures the convergence of Pareto front, obtained by the algorithm to a known set of true Pareto optimal solutions. The second metric (Δ), called the diversity metric, measures the spread, achieved among the solutions and the extension of the Pareto front. In the same way, to compare the performance of NSGSA with that of MOGSA and SMOPSO, the two metrics, defined in Ref. [7] are utilized. These two metrics, named as Generational Distance (GD) and Spacing (S) are used to measure the convergence and the spread of Pareto front, respectively.

4.1 Parameter Setting

NSGSA has eight control parameters. These parameters and their tuned values are listed in Table 1. To make the results comparable with those of NSGA-II, the number of function evaluations is considered to be 25000, as in Ref. [2]. Therefore, the maximum number of iterations, t_{max}, is 250. Also, the length of external archive ($n_{archive}$) is set to 100. Similarly, to compare results with MOGSA and SMOPSO, the number of function evaluations is considered to be 210000 for MOP5, 60000 for MOP6 and 40000 for MOPC1, as in Ref. [1] and [7] and the length of external archive is set to 800.

Description	Parameters	value
Swarm size	n_{mass}	100
Reordering mutation probability	P_r	0.4
Sign mutation probability	P_s	0.9
Uniform mutation probability	P_u	0.01
Percent of elitism	$P_{elitism}$	0.5
Initial value of inertial coefficient	w_0	0.9
Final value of inertial coefficient	w_1	0.5
Coefficient of search interval	β	2.5

Table 1. The parameters of NSGSA and their tuned values

4.2 Results and Discussion

The results, presented in Ref. [2] are for real-coded and binary coded variants of NSGA-II, none of which outperforms the other in all benchmarks. NSGSA is compared with the both variants of NSGA-II. Table 2 compares the mean and variance of the convergence metric, γ, obtained using NSGA-II (real-coded), NSGA-II (binary-coded) and NSGSA. As can be seen, regarding the mean value of γ, NSGSA has better converge in FON, POL, KUR, ZDT1, ZDT3 and ZDT6 problems than the two other algorithms, while in ZDT4, the real-coded NSGA-II, and in ZDT2, the binary-coded variant outperform the others, and in SCH all algorithms have the same convergence rate.

Table 2 shows the mean and variance of the diversity metric, Δ, obtained using the three algorithms. The Pareto front, obtained using NSGSA has better diversity than the two other algorithms in all benchmarks except in ZDT4, regarding the mean value of the diversity metric. In ZDT4, the binary-coded NSGA-II has obtained a little better spread with respect to the others.

Table 4 and Table 5 compare the performance of NSGSA with that of MOGSA and SMOPSO in the sense of the mean generational distance and spacing metrics, respectively. As can be seen, NSGSA has superior convergence in MOP5 and MOPC1 problems than the two other algorithms, while in MOP6, the MOGSA has a little better convergence. Same as the results in Table 2, the Pareto front,

obtained using NSGSA has better diversity than the two other algorithms in all benchmarks, as compared in Table 4. There are two important reasons that NSGSA has provided better spread in Pareto front. First, an enhanced method to prune the external archive with the limited length is used. Second, some specific members of the archive with higher rank are inserted to the list of moving particles to search the extreme and the least crowded spaces of the Pareto front.

Algorithm		SCH	FON	POL	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGSA	$\bar{\gamma}$	0.003	0.001	0.013	0.017	0.001	0.002	0.005	6.709	0.012
	σ_γ	0	0	0.000	0	0.000	0.003	0.000	3.104	0.002
NSGA-II (real-coed)	$\bar{\gamma}$	0.003	0.002	0.015	0.029	0.033	0.072	0.114	0.513	0.296
	σ_γ	0	0	0.000	0.000	0.005	0.032	0.008	0.118	0.013
NSGA-II (binary-coded)	$\bar{\gamma}$	0.003	0.002	0.017	0.029	0.001	0.001	0.043	3.226	7.807
	σ_γ	0	0.000	0.000	0.000	0	0	0.000	7.307	0.002

Table 2. Mean and variance of the convergence metric γ, obtained for NSGSA and NSGA-II

Algorithm		SCH	FON	POL	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGSA	$\bar{\Delta}$	0.004	0.005	0.010	0.013	0.014	0.050	0.248	0.734	0.489
	σ_Δ	0.003	0.003	0.025	0.005	0.037	0.080	0.001	0.115	0.006
NSGA-II (real-coed)	$\bar{\Delta}$	0.478	0.378	0.452	0.411	0.390	0.431	0.738	0.703	0.668
	σ_Δ	0.003	0.000	0.003	0.001	0.002	0.005	0.020	0.065	0.010
NSGA-II (binary-coded)	$\bar{\Delta}$	0.449	0.395	0.503	0.442	0.463	0.435	0.575	0.479	0.644
	σ_Δ	0.002	0.001	0.004	0.001	0.041	0.025	0.005	0.009	0.035

Table 3. Mean and variance of the diversity metric Δ, obtained for NSGSA and NSGA-II

Algorithm	MOP5	MOP6	MOPC1
NSGSA	1e-4	3e-5	0.001
SMOPSO	0.011	3e-4	0.003
MOGSA	0.001	2e-5	0.004

Table 4. Mean of generational distance metric, obtained for NSGSA, SMOPSO and MOGSA

Algorithm	MOP5	MOP6	MOPC1
NSGSA	0.034	5e-4	0.057
SMOPSO	0.396	0.003	0.116
MOGSA	0.181	0.001	0.180

Table 5. Mean of spacing metric, obtained for NSGSA, SMOPSO and MOGSA

5 Conclusion

In this paper, an extension of the single objective GSA to multi-objective optimization problems was presented. The proposed algorithm, called NSGSA, utilizes the non-dominated sorting approach to update the gravitational accelerations. An external archive of the Pareto optimal solutions was also used to provide some elitism. The new found non-dominated solutions are continually added to the limited length external archive. When the length of the external archive is violated, one member is selected to be removed. For this purpose, a spread indicator, defined as the deviation of the members' crowding distances from the average, was suggested to select a member to be removed. To promote and preserve diversity within the moving particles, two novel mutation operators, called sign and reordering mutations were proposed.

The performance of NSGSA was tested over a set of benchmark problems, to compare its results with those of the real-coded and binary-coded variants of NSGA-II, MOGSA and SMOPSO. The results show that the new multi-objective variant of GSA, proposed as NSGSA, can obtain comparable and even better performance in the sense of convergence rate and spread of solutions.

Acknowledge

The authors would like to thank Dr. Hossein Nezamabadi-pour for providing the code of single-objective GSA that helped the authors develop the multi-objective variant.

References

[1] Cagnina, L., Esquivel, S., and Coello Coello, C.A., *A Particle Swarm Optimizer for Multi-Objective Optimization*, JCS&T, 4 (5), pp. 204-210 (2005).

[2] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., *A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, 6 (2) pp. 182-197 (2002).

[3] Dorigo, M., Maniezzo, V., Colorni, A., *The Ant System: Optimization by a Colony of Cooperating Agents*, IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1), pp. 29–41 (1996).

[4] Glover, F., *Tabu Search: Part I*, ORSA Journal on Computing 1 (3) pp. 190–206 (1989).

[5] Glover, F., *Tabu Search: Part* II, ORSA Journal on Computing 2 (1) pp. 4–32 (1990).

[6] Ho, S.L., Shiyou, Y., Guangzheng, N., Lo, E.W.C., Wong, H.C., *A Particle Swarm Optimization-Based Method for Multi-Objective Design Optimizations*, IEEE Transactions on Magnetics, 41 (5) pp. 1756–1759 (2005).

[7] Hassanzadeh, H. R., Rohani, M., *A Multi-Objective Gravitational Search Algorithm*, Second International Conference on Computational Intelligence, Communication Systems and Networks, CICSyN, pp. 7-12 (2010).

[8] Kennedy, J., Eberhart, R.C., *Particle Swarm Optimization*, In: Proceedings of IEEE International Conference on Neural Networks, 4 pp. 1942–1948 (1995).

[9] Kirkpatrick, S., Gelatto, C.D., Vecchi, M.P., *Optimization by Simulated Annealing*, Science 220, pp. 671–680 (1983).

[10] Tang, K.S., Man, K.F., Kwong, S., He, Q., *Genetic Algorithms and Their Applications*, IEEE Signal Processing Magazine 13 (6), pp. 22–37 (1996).

[11] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., GSA: a Gravitational Search Algorithm, Information Sciences, 179 (13) pp. 2232-2243 (2009).

[12] Srinivas, N., Deb, K., *Multi-objective Optimization using Non-dominated Sorting in Genetic Algorithms*, Evolutionary Computation, 2 (3) pp. 221-248 (1994).