Search Space Reduction Technique for Constrained Optimization with Tiny Feasible Space

Abu S. S. M. Barkat Ullah
University of New South Wales at the Australian Defence Force Academy
ACT-2600, Australia.
+61 2 6268 8180
barkat@adfa.edu.au

Ruhul Sarker
University of New South Wales at the Australian Defence Force Academy
ACT-2600, Australia.
+61 2 6268 8051
r.sarker@adfa.edu.au

David Cornforth
University of New South Wales at the Australian Defence Force Academy
ACT-2600, Australia.
+61 2 6268 8956
d.cornforth@adfa.edu.au

ABSTRACT
The hurdles in solving Constrained Optimization Problems (COP) arise from the challenge of searching a huge variable space in order to locate feasible points with acceptable solution quality. It becomes even more challenging when the feasible space is very tiny compare to the search space. Usually, the quality of the initial solutions influences the performance of the algorithm in solving such problems. In this paper, we discuss an Evolutionary Agent System (EAS) for solving COPs. In EAS, we treat each individual in the population as an agent. To enhance the performance of EAS for solving COPs with tiny feasible space, we propose a Search Space Reduction Technique (SSRT) as an initial step of our algorithm. SSRT directs the selected infeasible agents in the initial population to move towards the feasible space. The performance of the proposed algorithm is tested on a number of test problems and a real world case problem. The experimental results show that SSRT not only improves the solution quality but also speed up the processing time of the algorithm.

Categories and Subject Descriptors
G.1.6 Constrained optimization

General Terms
Algorithms.

Keywords
Evolutionary algorithms, evolutionary agent systems, genetic algorithms, agent-based systems, nonlinear programming, constrained optimization, search space reduction.

1. INTRODUCTION
Many real world decision processes require solving optimization problems involving a set of equality, non-equality or both constraints. The difficulties in solving constrained optimization problems arise from the challenge of finding good feasible solutions. The problem is much more challenging when the feasible space is very tiny compare to the search space. Solving this type of problem has become a challenging area in computer science and operations research due to the presence of high dimensionality, nonlinear parameter interaction, and multimodality of the objective function as well as due to the physical, geometric, and other limitations of different constraints [16].

Evolutionary Algorithms (EAs) have brought a tremendous advancement in the area of computer science and optimization with their ability to solve many numerical and combinatorial optimization, classifier system, and engineering problems [13, 18]. Nevertheless, most EAs developed are unconstrained search techniques and lack an explicit mechanism to bias the search in constrained search spaces [15]. Furthermore traditional EAs suffer from slow convergence to locate a precise enough solution because of their failure to exploit local information, and face difficulties solving multi-modal problems which have many local solutions within in the feasible space . Hence it is well established that they are not well suited for fine tuning search [3, 27] and so, to improve the performance, hybridization of algorithms has been introduced in recent times.

Recently some researche has incorporated evolutionary processes into agent based systems [1-3, 10, 17, 26, 27]. Agent-based computation introduces a new paradigm for conceptualizing, designing and implementing intelligent systems, and has been widely used in many branches of computer science [12]. The agents are discrete individuals situated in an environment having a set of characteristics and rules to govern their behavior and interactions. They sense the environment and act on it in pursuit of a set of goals or tasks for which they are designed. Recently, a number of agent-based hybrid algorithms have appeared in the literature for solving different problems. For example, Dobrowolski *et al.* [9] used evolutionary multi-agent system for solving unconstrained multi-objective problems. Siwik *et al.* [26] developed a semi-elitist evolutionary multi-agent system and they solved so called MaxEx multi-objective problem, which is a quite simple problem [7]. Zhong *et al.* [27] used a multiagent genetic algorithm (MAGA) for solving unconstrained global numerical optimization problems. Liu *et al.* [17] used a multiagent evolutionary algorithm for constraint satisfaction problems. Barkat Ullah *et al.* [2] used an evolutionary agent systems for solving mathematical programming. Multi-agent genetic algorithms [1] and agent-based memetic algorithms [3] by Barkat Ullah *et al.* have solved constrained optimization problems.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

In this paper, we present a Search Space Reduction Technique (SSRT) with Evolutionary Agent System (EAS) for solving Constrained Optimization Problems (COPs) with tiny feasible space. The main idea of SSRT is to move some of the randomly generated initial poor solutions towards the feasible region. It is worth mentioning that clustering is different from the SSRT. When the ratio between the feasible region and the search space of the problem is very small (close to zero), we define those problems as COPs with tiny feasible space. Usually in solving these types of problems the algorithms need a good effort in finding the feasible space. As the initial population of EAs is randomly generated to ensure diversity, it may cause delay (being over diversified) in reaching a reasonably good solution for tiny feasible space. Once a good solution point is obtained, Genetic Algorithms (GAs) usually converge nicely to an acceptable solution. To enhance the performance of the algorithm in reaching the feasible space quickly, in the proposed algorithm, the agents apply SSRT to the initial population before starting the evolutionary process. This technique guides the initial agent population to move towards the feasible region. The agents try to find a centroid from the feasible agents (if any) with some good infeasible agents (here we are considering those agents having less constraint violations). A certain percentage of the worse infeasible agents are then encouraged to move towards the centroid. By applying SSRT the randomly generated agents are no longer random rather they have learnt a direction towards the feasible space which helps the algorithms to reach the feasible region faster and improve the solution quality. However some questions arise:

- When we should apply the SSRT?
- How to calculate the centroid for SSRT?
- How long we shall apply SSRT as it decreases the diversity?

Our experiments aim to find answers to these questions.

In the proposed framework of EAS, an agent represents a candidate solution of the problem, carries out cooperative and competitive behaviors, and selects the appropriate local search adaptively to find optimal solutions for the problem in hand. The agents cooperate and compete through well known Simulated Binary Crossover (SBX) proposed by Deb and Agrawal [8] and our designed different types of life span learning processes to solve a COP. Note that they do not use any mutation operator as the life span learning process would cover more than the purpose of mutation. We have designed the life span learning processes (LSLPs) based on several local and directed search procedures. An agent chooses a LSLP as a local search operator adaptively. As we generally see in GAs, an individual in a generation produces offspring and the offspring may be mutated to change its genetic materials. In reality, beside reproduction, an individual learns and gains experiences in different ways during its life time. This process is represented by the proposed LSLPs in this research. As an individual can decide to have a particular learning process based on its belief, we have incorporated a number of different LSLPs. The main idea of using multiple LSLPs is to achieve improved search performance by selecting a suitable LSLP from the pool and to reduce the chances of utilizing inappropriate LSLP. In EAS the improved agent (after applying LSLP) is sent back into the population which follows the Lamarckian learning. Since we are using multiple LSLPs (i.e. LSs) in the spirit of Lamarckian learning we can say EAS is following Meta-Lamarckian learning [21]. The individual's ability to use its belief, to interact with the environment, and to make independent decision for exploration and learning, qualifies an individual to be called an agent in the population.

In [1], the authors used the neighborhood competition with orthogonal crossover and Gaussian mutation. As the algorithm with neighborhood competition and mutation [2, 3] was computationally expensive to achieve quality solutions, we have used SBX crossover with LSLPs and SSRTs in this research.

The performance of EAS is very competitive when compared with [4, 11, 22] (details can be found in [2]). To test the performance of the algorithm and justify the purpose of SSRT, we have selected a set of benchmark problems with tiny search space from [15] and carried out different type of experiments. We have also solved a real world case problem. The experiments show SSRT improves the solution qualities as well as speeds up the performance of the algorithm. We believe the performance of any algorithms can be enhanced by incorporating SSRT when solving constrained optimization problems with tiny feasible region.

The rest of this paper is organized as follows. Section 2 describes the proposed the Evolutionary Agent System with SSRT and its components. Section 3 describes the performance of the proposed approach on benchmark problems, and compares the results. Finally, Section 4 concludes the paper and provides future research directions.

2. Evolutionary Agent System

In this research, we have incorporated the agent concept with evolutionary algorithms, where an agent represents a candidate solution in the population. In solving constrained optimization problems, the algorithm searches the search space for a feasible region from the randomly generated initial population, and then tries to find out the optimal solution. If the algorithm finds the feasible space quickly, it can save a considerable amount of time in searching for good quality feasible solutions. However, it is not easy to find the feasible space and even more difficult when the feasible space is very tiny. Population based search algorithms like EAs start with initial randomly generated population. In our proposed EAS, the initial population of the agents is also generated randomly. However before starting the evolutionary process we check the fitness landscape of the population. If there are a very few (or none) quality solutions generated in the initial stage, the algorithms may need more time to explore the search space for the feasible space. This is usually the case for COPs with a tiny feasible region. In this situation, the algorithm applies the SSRT. The SSRT provides the initial random agents an approximate direction to move towards the feasible space. This helps them to find the feasible space faster. The detail of SSRT is discussed later in this section. The goal of each agent is to:

- Reach into the feasible region and
- Improve the fitness

Following the natural adaptation process, in the proposed EAS the agents improve their fitness by selecting self-adaptively a suitable life span learning technique along with SSRT. The agents are arranged in a lattice-like environment E of size $M \times M$ (where M is always an integer). The agents communicate with their surrounding neighbor agents and interchange information with

them through comparison and the crossover operator. The overlapped small neighborhoods of the lattice-like environment help in exploring the search space because they induce a slow diffusion of solutions through the population and provide a kind of exploration (diversification), while exploitation (intensification) takes place inside each neighborhood by crossover operations.

For LSLPs, we have proposed a number of search processes, as an appropriate choice of a LSLP is very important for the performance of the algorithm. In each generation an agent may select one of the several LSLPs based on self-adaptively.

The main steps of the proposed algorithm are follows:

Step 1. Create a random population, which consists of $M \times M$ agents.
Step 2. Arrange the agents in a lattice-like environment.
Step 3. Evaluate the agents individually.
- If the stopping criterion has been met, go to step 8; otherwise continue.

Step 4. If the number of feasible individuals is very few (<5%) apply SSRT; otherwise go to step 5.
Step 5. For each agent examine its neighborhood.
- Select an agent from its neighborhood and perform crossover.

Step 6. Select a certain percentage of agents.
- Select self-adaptively a life span learning process.

Step 7. Go to step 3.
Step 8. Stop.

As the search spaces of the optimization problems and the variables under consideration are real and continuous, we have used real numbers to represent the solutions.

2.1 Search Space Reduction Technique

In population based stochastic algorithms like EAs the quality of the randomly generated initial population plays an important role on the performance of the algorithms. If the initial population contains some good solutions the algorithms converge quickly. However it is not expected that the random solutions should always be of good quality. Some algorithms like GENOCOP [18, 19] assumes a feasible starting point (or feasible initial population), which implies that the user or the EA must have a way of generating (in a reasonable time) such starting point [5]. The homomorphous mapping method of Koziel and Michalewicz [14] also requires an initial feasible solution. In this paper we have designed a SSRT which allows the far most infeasible agents to move towards the feasible region, which is basically squeezing the search space. That means the evolutionary process starts with a better population in a reduced search space.

If there are none or few (less than 5%) feasible agents in the initial random population, the EAS applies SSRT. The infeasible agents are ranked based on their constraint violation. The feasible agents (if any) and a certain percentage of the top ranked infeasible agents are then used to find a centroid. If there is no feasible agent, only the top ranked infeasible agents are modified to reduce the constraint violation and then calculate the centroid.

After calculating the centroid a certain percentage of non-allowable infeasible agents (see Figure 1) are allowed to move towards the centroid. Although this process guides the worse infeasible agents towards the feasible space, it reduces the diversity of the population. To ensure diversity we only allow a small number of the worse infeasible agents to follow the centroid, and discontinue the process when the diversity of initial population is decreased to a certain level.

Figure 1 shows how the centroid is calculated from the feasible (if any) and allowable infeasible solutions. A portion of the non-allowable infeasible solutions then follow the centroid, which is shown by the arrows.

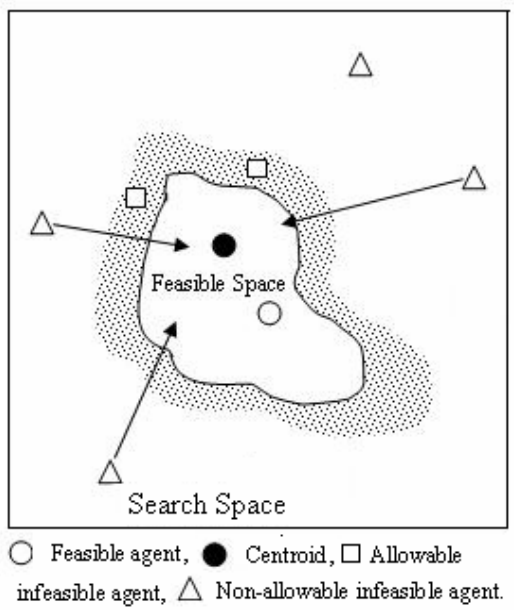

Figure 1. Search Space Reduction Technique.

The proposed algorithm for SSRT is given below:

Step 1. Rank the infeasible solutions based on the Constraint Violation (CV). Define the allowable infeasible range. Calculate the diversity of the population.
Step 2. If there are feasible individuals, calculate the centroid from the feasible and allowable infeasible individuals, and go to step 4, else go to step 3.
Step 3. Find the best infeasible individuals based on the CV.
a. Find the constraint which has maximum CV for this individual.
b. Select a variable randomly, which is involved in the constraint which is not yet modified.
c. Change the variable with $\pm\delta$ and mark as modified.
d. If the individual become feasible, go to step 2, else go to step 3e
e. If all constraints are checked or all the variables are modified, then find the centroid of the allowable infeasible along with this one, otherwise go to step 3.

Step 4. Force the certain percentage of the non-allowable infeasible solutions to follow the centroid.
Step 5. Calculate the diversity of the population. If the diversity decreases up to a certain level then stop, otherwise go to step 2.

The value of δ used here is very small;, δ = |*G* (0, 1)|, where *G* (0, 1) is a Gaussian random number generator with zero mean and standard deviation 1.

For calculating the centroid, for each variable of the centroid x_i^c, we have taken the arithmetic mean of the participating agents' respective variables x_i. While calculating the diversity we have taken the mean Euclidian distance of the all agents from the centroid.

A portion of the non-allowable agents follow the centroid as follows:

$$x_i^n = \alpha x_i + (1-\alpha)x_i^c, \quad i = 1 ..... n \qquad (1)$$

[Where x_i^n and x_i^c are the i^{th} variable of the non-allowable agent and the centroid respectively, n is the number of variables in the solution vector and α is a uniform random number from 0 to1].

2.2 Crossover

In the proposed EAS, we have used Simulated Binary Crossover (SBX) [8]. SBX operator performs well in solving problems having multiple optimal solutions with a narrow global basin, and has been used in different applications successfully [2, 3, 7].

When this crossover operator is applied on an agent $A_{i,j}$ (located in location (i,j) in the lattice), the agent searches for its best neighbor agent e.g. $B(A_{i,j})$ to mate. The better offspring of the two, denoted as $N_{i,j}$, is stored in a pool. After completing the crossover in a given generation, the fitness of each $N_{i,j}$ ($1 \le i,j \le M$) is compared with its parent $A_{i,j}$. If the new agent's fitness is better than its parent then it takes the place of $A_{i,j}$ and $A_{i,j}$ dies. Otherwise $A_{i,j}$ would survive.

2.3 Life Span Learning Process (LSLP)

After crossover a certain percentage of agents (with P_L probability) of the population are selected to apply the LSLP i.e. to apply local and directed search. These LSLPs are designed to improve fitness values by changing the variable vector of the existing solutions in different ways.

Here we have designed four different LSLPs. The first one is totally random in nature, the second is restricted random, the third is gradient-based, and the last is directed search. The random LSLPs ensure diversity, and the directed searches try to move towards a better solution which is not necessarily in the individual's locality.

The first and second types of LSLP of the agent try to exploit the existing solution vector by attempting to change the variables. The third type attempts to move the solution vector in the direction of gradient and the last type of LSLP leads the agents towards the current best solution. The last two LSLPs try to make the agents converge; the first two maintain diversity.

During the LSLP the variables are changed with ±Δ. The direction of Δ (add/ deduct) is selected by observing the modified fitness value of the agent. The value of Δ should be very small and gradually be decreasing with the generation numbers, to obtain a high quality solution (high precision) at the end. We have considered Δ = |*G*(0,1/*g*)| (for the first three types of LSLPs), where *G*(0,1/*g*) is a Gaussian random number generator with zero mean and standard deviation (1/*g*), here *g* is the present generation number. In the fourth LSLP $\Delta = (b_r - a_r)/2$, where b_r is the r^{th} solution variable of the previous best agent, a_r is the r^{th} variable of the present agent and a_r is updated in each step, which speed up the directed search. The details of the LSLPs can be found in [2, 3].

2.4 Fitness Evaluation and Constraint Handling

In EAS, the goal of the individual agent is to minimize the objective function value while satisfying the constraints. To improve the fitness, agents first apply crossover operators with their best neighbors. The best neighbor is found by using pair-wise comparison among the neighbors. The pair-wise comparison indirectly handles the constraints. Like Deb [6], while comparing the fitness of two individual agents we have considered:

1. A feasible individual is always better than an infeasible individual.
2. If both of the individuals are feasible, then the individual with lower objective function value is better (considering minimization problem).
3. If both of them are infeasible, then the one with less constraint violation is better. The total Constraint Violation (CV) of an individual is considered here as the sum of absolute values by which the constraints are violated.

We have assumed that the fitness of the best infeasible agent is worse than the worst feasible agents. As such while comparing two agents, the infeasible agent is penalized and feasible agent is rewarded, so the constraints are handled indirectly. The calculation of improvement index in the following subsection also indirectly handles the constraints.

3. EXPERIMENTAL RESULTS AND DISCUSSION

In this study we have considered a set test problems as COPs with tiny feasible space from the literature [15, 22]. From the set of benchmark problems we have chosen only those problems whose feasible region is very small compared to their search space. To get an estimate of how tiny is the feasible space of these problems, a metric suggested by Michalewicz and Schoenauer [20] is used, $\rho = \frac{|F|}{|S|}$, where $|S|$ is the number of random solutions generated (1,000,000 in this case), and $|F|$ is the number of feasible solutions found (out of the total randomly generated solutions). We have considered only those problems whose ρ is less than 5%. The benchmark problems are completely different in number of variables, type of objective functions, and type of constraints. The characteristics of the benchmark problems such as the number of decision variables (n), type of objective function, ρ, number of linear inequalities (LI), number of nonlinear inequalities (NI), number of linear equalities (LE), number of nonlinear equalities (NE), number of active constraints (AC), and the optimal values are shown in Table 1.

Table 1. Characteristics and the optimal results of the benchmark problems

Prob	*n*	Obj. Fuc.	ρ(%)	LI	NI	LE	NE	AC	Optimal
1	13	Quadratic	0.00	9	0	0	0	6	-15.000
2	20	Nonlinear	1.00	0	2	0	0	1	-0.803619
3	10	Polynomial	0.00	0	0	0	1	1	-1.000
4	5	Quadratic	0.52	0	6	0	0	2	-30665.539
5	4	Cubic	0.00	2	0	0	3	3	5126.498
6	2	Cubic	0.00	0	2	0	0	2	-6961.814
7	10	Quadratic	0.00	3	5	0	0	6	24.306
8	2	Nonlinear	0.01	0	2	0	0	0	-0.095825
9	7	Polynomial	0.01	0	4	0	0	2	680.630
10	8	Linear	0.00	3	3	0	0	6	7049.331
11	2	Quadratic	0.00	0	0	0	1	1	0.750
12	3	Quadratic	0.05	0	9^3	0	0	0	-1.000
13	5	Nonlinear	0.00	0	0	0	3	3	0.053950
14	10	Nonlinear	0.00	0	0	3	0	3	-47.764
15	3	Quadratic	0.00	0	0	1	1	2	961.715
16	5	Nonlinear	0.00	4	34	0	0	4	-1.905
17	6	Nonlinear	0.00	0	0	0	4	4	8853.540
18	9	Quadratic	0.00	0	13	0	0	6	-0.866025
20	24	Linear	0.00	0	6	2	12	16	0.204979

ρ= Ratio between the feasible space and the search space, LI=Linear Inequalities, NI=Nonlinear Inequalities, LE= Linear Equalities, NE= Nonlinear Equalities, AC=Active Constraints

First we have carried out an experiment to justify the necessity of SSRT. In this experiment, EAS is run with and without SSRT. The best, mean, standard deviation, worst, and median results of 30 independent runs (with 30 different random seeds) are given in Table 2. The last column of Table 2 shows the average number of generations required to find the best results as an indication how fast the algorithm achieves the quality solutions.

For an agent's neighborhood, we have considered the combined approach [1-3] which allows an agent to consider its four neighbors and eight neighbors interchangeably as neighborhood agents. During the LSLP the agent is allowed at most m = 10 steps. We have investigated the effect of learning steps: lower values of m slow down the convergence while larger values increase computational cost rather than performance. The maximum number of generations was set in this experiment at 500,000. The population size and P_L we have used are 400 and 0.2 respectively. The allowable range (AR) of infeasible agent for calculating centroid is used maximum 50% of the infeasible agents and maximum diversity reduction from the initial random population is considered 10%. The initial solution vectors for the agents are randomly generated within the bounds of each decision variable.

From Table 2 we can see AMA with SSRT is achieving better results in different aspects than without SSRT (bold fonts indicates better achievements of EAS with SSRT). If we consider the Mean results, EAS with SSRT has achieved a better mean in 76.47% more times than normal EAS. EAS with SSRT has also performed better in achieving the Best results in 35.29% of the problems and same in 41.18% problems. On 76.47% times the Worst results and on 58.82% times the Median results of EAS with SSRT were better than only EAS. If we consider the average number of generations required to find out the best result, in 35% of problems AMA with SSRT is faster than the other approach, and in 47% both approaches need the same number of generations.. So we can say by applying SSRT in most of the problems EAS has improved either the solution or computational time or both.

Table 2. Performance of EAS with SSRT and without SSRT from 30 independent runs

Pb	EAS	Best	Mean	StDev	Worst	Median	AvgGen
1	NST	-15.000	-14.922	2.97E-01	-13.828	-15.000	1080.33
	ST	-15.000	**-15.000**	**1.37E-08**	**-15.000**	-15.000	**1044.48**
3	NST	-1.000	-1.000	1.26E-05	-1.000	-1.000	1250.00
	ST	-1.000	-1.000	**4.10E-06**	-1.000	-1.000	1250.00
5	NST	5249.969	5258.760	3.07E+00	5261.768	5258.919	1250.00
	ST	**5129.057**	**5246.486**	3.00E+01	**5258.905**	**5254.750**	1250.00
6	NST	-6961.810	-6958.554	8.53E+00	-6932.399	-6961.801	1206.67
	ST	**-6961.813**	**-6961.805**	**3.22E-03**	**-6961.801**	**-6961.804**	1211.07
7	NST	24.309	24.389	8.06E-02	24.628	24.360	1179.20
	ST	24.318	**24.355**	**1.77E-02**	**24.379**	**24.356**	**1158.23**
8	NST	-0.095825	-0.095825	1.92E-08	-0.095825	-0.095825	500.67
	ST	-0.095825	-0.095825	**4.23E-17**	-0.095825	-0.095825	606.77
9	NST	680.655	680.930	3.46E-01	682.642	680.933	1032.90
	ST	**680.645**	**680.763**	**7.04E-02**	**680.870**	**680.759**	1173.23
10	NST	7052.071	7330.425	2.50E+02	8005.283	7307.057	1101.10
	ST	7058.760	**7097.425**	**2.95E+01**	**7162.383**	**7096.189**	**1097.50**
11	NST	0.750	0.761885	2.83E-02	0.841718	0.750	1250.00
	ST	0.750	**0.750**	**9.87E-09**	**0.750**	0.750	1250.00
12	NST	-1.000	-1.000	2.01E-10	-1.000	-1.000	153.77
	ST	-1.000	-1.000	**4.55E-11**	-1.000	-1.000	**145.70**
13	NST	0.053960	0.203669	1.88E-01	0.606155	0.129390	1250.00
	ST	0.053962	**0.057688**	**3.15E-03**	**0.064314**	**0.056828**	1250.00
14	NST	-46.944	-46.379	3.84E-01	-45.598	-46.436	1250.00
	ST	**-46.923**	**-46.436**	**2.16E-01**	**-46.129**	-46.389	1250.00
15	NST	961.738	967.518	5.01E+00	974.776	967.511	1250.00
	ST	**961.715**	**965.301**	**3.33E+00**	**970.925**	**966.297**	1250.00
16	NST	-1.905	-1.901	5.44E-03	-1.881	-1.902	1226.67
	ST	-1.905	**-1.905**	**1.86E-07**	**-1.905**	**-1.905**	**1200.90**
17	ST	8927.598	8957.514	6.84E+01	9128.155	8927.638	1250.00
	NST	8927.598	9069.241	1.10E+02	9185.712	9134.988	1250.00
18	NST	-0.866023	-0.865886	1.55E-04	-0.865358	-0.865926	1184.53
	ST	-0.866007	**-0.865958**	**3.56E-05**	**-0.865900**	**-0.865956**	**1152.68**
20	NST	0.315067	0.331503	9.22E-03	0.351925	0.333757	1250.00
	ST	**0.316781**	**0.323671**	**4.78E-03**	**0.332948**	**0.323025**	1250.00

NST= EAS without SSRT, ST = EAS with SSRT. Avg.Gen.= Average generation required to find best results.

3.1 Effect of parameters used in SSRT

EAS apply SSRT when there are very few feasible agents (less than 5% of the population size) in the initial random population. In applying SRRT, a centroid is calculated using certain feasible

and top ranked infeasible solutions. In the process, we need to decide the number (or percentage) of infeasible solutions in calculating centroid and a stopping criteria for SSRT. As the diversity of the population decreases with application of SSRT, we use diversity measure as a stopping criterion. We have discussed these two parameters in the following subsections.

3.1.1 Effect of Allowable Range (AR) in calculating SSRT

While calculating the centroid, if we allow all of the infeasible agents with the feasible agents (if any), it may move the centroid towards an infeasible region. In the process, we rank the infeasible agents based on their constraint violation and consider a certain percentage of the top ranked infeasible agents. As these participating agents are better than the others, it is expected that the centroid may have better fitness than the low ranked agents. To see the effect of the percentage of top ranked infeasible agents used in SSRT on the overall solution, we have carried out experiments by varying the percentage of the top ranked infeasible agents (10% to 50% with an increment of 10%) while leaving the other parameters constant e.g. population size, P_L, Diversity Reduction (DR). We then compared with the solution of EAS without SSRT.

We have compared the best, mean, standard deviations, worst, median, and average number of generations required to find out the best result for 30 independent runs. As the centroid guides the worse infeasible solutions, the quality of the centroid plays a vital role to the performance of the algorithms. If we consider a very small number of top ranked infeasible agents with feasible agents (if any, since the feasible space is very tiny), they may not provide a good quality and global solution. If we increase the AR, the centroid contains diversity and the performance of the algorithm is improved. However, the range of AR should not be too large, since the use of higher percentage could mislead the search process where multiple disjointed feasible spaces exist for a problem. In most of these test problems AR ranges 30% to 50% provides better results.

Table 3. Effect of allowable range for calculating centroid on problem 1

AR (%)	Best	Mean	St. Dev.	Worst	Median	Avg Gen
0	-15.000	-14.921875	2.97E-01	-13.828125	-15.000	1080.33
10	-15.000	-14.960937	2.14E-01	-13.828123	-15.000	1027.93
20	-15.000	-14.960937	2.14E-01	-13.828124	-15.000	1058.20
30	-15.000	-14.960937	2.14E-01	-13.828125	-15.000	1075.63
40	-15.000	-14.960937	2.14E-01	-13.828125	-15.000	1069.20
50	-15.000	-15.000000	1.46E-07	-15.000000	-15.000	1036.97

AR= Allowable range of infeasible agents for centroid, Avg Gen = Average number of generation required to find the best result.

In Table 3 we have shown the results of problem 1 as an example. For the best results of the 30 runs in all the cases EAS has achieved the optimal. When we have considered the top 10% infeasible agents to calculate the centroid the performance of the algorithm has improved by improving the mean result. Considering up to 40% infeasible agents, only the worst result improves. However when we have considered 50% top infeasible to find the centroid the performance of the algorithm is the best among the 6 sets of results. We are maintaining enough diversity in the population by using large allowable range, which helps finding a better centroid. Nevertheless we should not consider too many infeasible agents, which may not help the agents but rather direct them to other areas of the search space, resulting in longer processing time. Though the test problems are diverse in nature, in solving most of the problems EAS shows similar behavior. The AvgGen shows that increasing AR speeds up the processing time. However if we consider a very large AR it may slow down the process as it will have the affect of poor quality solutions.

3.1.2 Effect of Diversity Reduction(DR)

When the low ranked infeasible agents move towards the centriod, the diversity of the population decreases. For population based search ensuring diversity is an important issue. If the diversity of the population decreases too much then the performance of the algorithm also decreases. So for SSRT it is a critical issue to maintain diversity while attracting the low ranked infeasible agents towards the centroid. A small reduction of the diversity of the population by applying SSRT improves the performance of EAS. Since the initial population is randomly generated this may be over diversified. By applying SSRT, we can still provide sufficient diversity by controlling the diversity reduction.

To show the effect of reducing diversity during SSRT, we have carried out experiments with different percentage of diversity reduction (10% to 50% with an increment of 10%) while keeping the other parameters constant (Population size, P_L, AR). We stopped SSRT when the diversity reduced to a certain percent (e.g. 10% for the first experiment) from the initial stage. In general, a low range of diversity reduction (10-20%) improves the performance of EAS. However a higher value of diversity deteriorates the quality of performance due to the lack of diversity in the population. The Results of the experiment for 30 independent runs for problem 01 is given in Table 4.

Table 4. Effect of diversity reduction on problem 1

DR (%)	Best	Mean	St. Dev.	Worst	Median	Avg Gen
0	-15.000	-14.921875	2.97E-01	-13.828125	-15.000	1080.33
10	-15.000	-15.000000	2.83E-08	-15.000000	-15.000	1052.30
20	-15.000	-15.000000	2.91E-08	-15.000000	-15.000	1067.13
30	-15.000	-14.960937	2.14E-01	-13.828123	-15.000	1049.07
40	-15.000	-14.960937	2.14E-01	-13.828123	-15.000	1018.47
50	-15.000	-14.960937	2.14E-01	-13.828123	-15.000	1027.93

DR= Relative Diversity Reduction from the initial randomly generated population after SSRT, Avg Gen = Average number of generations required to find the best result.

The experimental results show the small reduction of diversity like 10%-20% gives the algorithm better performance. However a large amount of diversity reduction is not helping the EAS significantly. If we consider the mean, standard deviation and worst results, the performance of EAS is best with 10% relative diversity reduction. That indicates SSRT improves the performance of the algorithm, however we need ensure diversity as well by allowing SSRT with small percentage of relative diversity reduction.

3.2 Solving a Real World Problem

Most of the test problems considered in the previous section are smaller in size. Although the contribution of SSRT is positive in those problems, the real improvements look tiny. To test the performance of SSRT in solving a reasonable size problem, we have used a real world crop planning problem [23-25] in this section. Crop planning is related to many factors such as the type of lands, yield rate, weather conditions, availability of the agricultural inputs, crop demand, capital availability, and the cost of production. The country, under consideration, grows a wide variety of crops in different seasons, and it has different types of lands. For a single-cropped land, there are a number of alternative crops from which the crop to be cultivated in a year may be chosen. Similarly there are many different combinations of crops for double-cropped (two crops in a year) and triple-cropped (three crops in a year) lands. Different combinations give different outputs. The problem is to provide an annual crop production plan that determines the area to be used for different crops while fulfilling the demand, land, capital, import, and region limitations. As our objective is to test the performance of SSRT, we ignore the details of the problem here. However, interested readers can find the details in [23-25]. Here we have solved the constrained non-linear single objective model of the crop planning problem [23-25]. The original model consists of 68 variables and 45 constraints. By applying variable / constraint reduction technique, the model can be reduced to 39 variables and 15 constraints.

We have solved this problem using EAS with and without SSRT. The average results of 30 independent runs are given in Table 5. From the table we can see EAS with SSRT achieves remarkably 4.55% better fitness value than the EAS without SSRT. The EAS with SSRT consistently provides better mean, standard deviation, worst, and median values. However, EAS with SSRT took higher number of generations, as EAS without SSRT prematurely converged to sub-optimal solutions. SSRT improves the result of EAS for these small benchmark problems, though the results for many problems are not statistical significantly improved. However for the real crop planning problem SSRT improves the performance significantly.

Table 5. Performance of EAS with SSRT and with out SSRT in solving Crop problems

EAS	Best	Mean	St. Dev.	Worst	Median	AvgGen
NST	-2.20E+07	-2.09E+07	1.03E+06	-1.87E+07	-2.13E+07	512.37
ST	-2.30E+07	-2.25E+07	2.76E+05	-2.20E+07	-2.25E+07	592.00

NST= EAS with out SSRT, ST = EAS with SSRT. Avg.Gen.= Average generation required to find best results.

4. CONCLUSIONS

This paper has discussed an evolutionary agent system (EAS) for solving constrained optimization problems (COPs) by tailoring agent concepts into evolutionary algorithms. A search space reduction technique (SSRT) is proposed to incorporate with EAS before applying the evolutionary process to solve the COPs with tiny feasible space. The proposed SSRT allows certain infeasible agents in the initial population to move slowly towards the feasible space. This approach usually improves the performance of EAS in terms of either solution quality or computational time or both. We have investigated the performance of SSRT by solving a set of test problems and a real world case problem. Although the idea of SSRT is very simple, the results justify the use of SSRT with EAS. We believe the performance of any algorithm can be enhanced by incorporating SSRT when solving constrained optimization problems with a tiny feasible region. We have also analyzed the effect of diversity reduction in the initial population, allowable range of infeasible agents to find the centroid for SSRT. In future research, we would like to test the performance of SSRT when incorporated with some other existing algorithms appeared in the literature.

5. REFERENCES

[1] Barkat Ullah, A. S. S. M., Sarker, R., and Cornforth, D. A Combined MA-GA Approach for Solving Constrained Optimization Problems, in *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on* (2007),2007. 382-387.

[2] Barkat Ullah, A. S. S. M., Sarker, R., and Cornforth, D. An Evolutionary Agent System for Mathematical Programming. in *Advances in Computation and Intelligence*: Springer, 2007, 187-196.

[3] Barkat Ullah, A. S. S. M., Sarker, R., Cornforth, D., and Lokan, C. An Agent-based Memetic Algorithm (AMA) for Solving Constrained Optimization Problems, in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (2007),2007. 999-1006.

[4] Chootinan, P. and Chen, A. Constraint handling in genetic algorithms using a gradient-based repair method. *Computers & Operations Research, 33*, *8*, (2006). 2263-2281.

[5] Coello Coello, C. A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191*, *11-12*, (2002). 1245-1287.

[6] Deb, K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*, *2-4*, (2000). 311.

[7] Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Inc., 2001.

[8] Deb, K. and Agrawal, R. B. Simulated Binary Crossover for Continuous Search Space. *Complex Systems, 9*,1995). 115-148.

[9] Dobrowolski, G., Kisiel-Dorohinicki, M., and Nawarecki, E. Evolutionary multiagent system in multiobjective optimisation, in *Proc. of the IASTED Int. Symp.: Applied Informatics* (2001), IASTED/ACTA Press., 2001.

[10] Dreżewski, R. and Marek, K.-D. Maintaining Diversity in Agent-Based Evolutionary Computation. in *Lecture Notes in Computer Science : Computational Science ICCS*, 2006, 908-911.

[11] Farmani, R. and Wright, J. A. Self-adaptive fitness formulation for constrained optimization. *Evolutionary Computation, IEEE Transactions on, 7*, *5*, (2003). 445.

[12] Ferber, J. *Multiagent systems as introduction to distributed artificial intelligence*, Addision-Wesley, 1999.

[13] Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[14] Koziel, S. and Michalewicz, Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation, 7*, *1*, (1999),

[15] Liang, J. J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello, C. A. C., and Deb, K. *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization*, Nanyang Technological University, http://www.ntu.edu.sg/home/epnsugan/, Singapore 2006.

[16] Liang, J. J. and Suganthan, P. N. Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism, in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on* (2006),2006. 9-16.

[17] Liu, J., Zhong, W., and Jiao, L. A multiagent evolutionary algorithm for constraint satisfaction problems. *Systems, Man and Cybernetics, Part B, IEEE Transactions on, 36*, *1*, (2006). 54-73.

[18] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1994.

[19] Michalewicz, Z. and Janikow, C. Z. GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints. *Communications of the ACM, 39*,1996),

[20] Michalewicz, Z. and Schoenauer, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation, 4*, *1*, (1996). 1-32.

[21] Ong, Y. S. and Keane, A. J. Meta-Lamarckian learning in memetic algorithms. *Evolutionary Computation, IEEE Transactions on, 8*, *2*, (2004). 99-110.

[22] Runarsson, T. P. and Yao, X. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on, 4*, *3*, (2000). 284.

[23] Sarker, R. and Quaddus, M. Modelling a Nationwide Crop Planning Problem Using a Multiple Criteria Decision Making Tool. *Computers and Industrial Engineering, 42(2-4), pp541-553.*,2002),

[24] Sarker, R. and Ray, T. Multiobjective Evolutionary Algorithms for solving Constrained Optimization Problems, in *International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA2005)* (Vienna, Austria.,28 - 30 November 2005,,2005), IEEE Press-USA, 2005. Pages.

[25] Sarker, R. A., Talukder, S., and Haque, A. Determination of Optimum Crop-Mix for Crop Cultivation in Bangladesh. *Applied Mathematical Modelling 21*,1997). 621-632.

[26] Siwik, L. and Kisiel-Dorohinicki, M. Semi-elitist Evolutionary Multi-agent System for Multiobjective Optimization. in *Lecture Notes in Computer Science : Computational Science ICCS*, 2006, 831-838.

[27] Zhong, W., Liu, J., Xue, M., and Jiao, L. A multiagent genetic algorithm for global numerical optimization. *Systems, Man and Cybernetics, Part B, IEEE Transactions on, 34*, *2*, (2004). 1128-1141.