Comparison of Evolutionary Algorithms on the Minimax Sensor Location Problem

Whitney Conner
The Pennsylvania State University
310 Leonhard Building
University Park, PA 16802
wac129@psu.edu

ABSTRACT
The Minimax Sensor Location Problem (SELP) is a nonlinear, nonconvex programming problem which aims to locate sensors to monitor a planar region. The objective is to determine the locations that will minimize the maximum probability of "missing" an event in the region. Two evolutionary algorithms, differential evolution (DE) and the co-variance matrix adaptation evolutionary strategy (CMA-ES), are used to solve the SELP when 2 to 10 sensors are to be located in the unit square. Several variants of each algorithm were tested and results show that the DE/best/2/bin variant of DE and the weighted recombination variant of CMA-ES yield the best performance. A comparison of DE and CMA-ES, in terms of the objective value obtained, reveals that the algorithms obtain comparable solutions. However, CMA-ES requires approximately half as many function evaluations as DE.

Categories and Subject Descriptors
G.4 [**Mathematics of Computing**]: MATLAB

General Terms
Algorithms, Performance, Reliability, Experimentation

Keywords
Sensor location, Differential Evolution, Co-variance Matrix Adaptation Evolutionary Strategy

1. INTRODUCTION
Sensor and detection systems pervade our daily lives, ranging from smoke alarms in our bedrooms to cellular base stations used in wireless communication systems. As technology and innovation progress, sensors are becoming smaller and cheaper while their capabilities are expanding. Krishnamachari [11] notes that wireless sensor networks (WSNs) have potential applications in "ecological habitat monitoring, structure health monitoring, environmental contaminant detection, industrial process control, and military target tracking." With such a wide range of applications and obvious societal impact, the need to study and optimize these systems is of growing importance.

Research in the field of WSNs has focused on several areas including network security and energy management but few have focused on sensor deployment. In light of this, consider the following problem: Suppose a planar region, *S*, is to be monitored by a system of sensors which detect events (e.g. hazardous material spills or someone placing a cell phone call) occurring in the region. Each sensor detects an event with a certain probability which depends only on the distance between the event and the sensor. The object of the Minimax Sensor Location Problem (SELP) is to determine the locations of sensors which will minimize the maximum probability of missing an event (a.k.a. the probability of non-detection). Because the sensor system is only as strong as its weakest point, the goal of the problem is to optimize the worst-case scenario.

SELP is a nonlinear, nonconvex programming problem and as such, is challenging for traditional math programming techniques. Instead, this paper will employ evolutionary optimization techniques to solve the SELP. Evolutionary algorithms (EAs) are direct search techniques based on the ideas of natural selection to "evolve" good solutions to problems. This paper investigates the use of Differential Evolution (DE) and the Co-variance Matrix Adaptation Evolutionary Strategy (CMA-ES) to solve the problem when 2 to 10 sensors are to be located in a square region. Section 2 will review relevant literature on the SELP and present the mathematical formulation of the problem. Section 3 will discuss the method used to parameterize the EAs. Section 4 will introduce Differential Evolution and describe the procedure used to parameterize the algorithm. Section 5 will do the same with respect to CMA-ES. Section 6 will compare results of DE and CMA-ES to the SELP problem. Section 7 will summarize the findings of the paper and suggest future work. All algorithms were implemented in the MATLAB environment. The appendix contains parameter settings for the CMA-ES algorithm and contour plots of the solutions obtained for the 10 sensor problem.

2. SENSOR LOCATION PROBLEMS
2.1 Literature Review
Dhillon *et al.* [3][4] addressed a discrete version of a closely related problem. Their objective was to minimize the number of sensors required to satisfy a threshold value on the probability of non-detection. Dhillon *et al.* suggest two greedy heuristics which sequentially add sensors to the region being monitored.

Recent work by Drezner and Wesolowsky [5] and Cavalier *et al.* [2] has addressed the continuous SELP problem. Drezner and Wesolowsky considered locating sensors on both the unit line and within the unit square. For the planar problem, Drezner and Wesolowsky discretized the event locations using a grid and then solved the problem by univariate search, math programming, simulated annealing, and a Demjanov algorithm. They found the Demjanov method provided the best solutions with reasonable computation times.

Cavalier *et al.*[2], extended the work by Drezner and Wesolowsky to consider locating identical sensors in arbitrary convex planar

regions. Instead of using standard math programming approaches to the problem, they used the computational geometry of the problem to develop a heuristic solution to the continuous problem. Their solution algorithm, called Towards the Largest Peak (TLP), uses Voronoi Diagrams to identify "peaks," or points of high probability of non-detection and moves sensors towards these peaks. TLP was compared with MATLAB's built-in fminimax solver, which uses a sequential quadratic programming (SQP) method, and shown to give comparable and often superior results with significantly faster computation times.

It is important to note that the SELP model is based on several assumptions which may not be applicable in real scenarios. However, insight can still be gained by solving the simplified version of the problem.

* Identical Sensors: All sensors in the system can be described by the same detection probability function.
* Events occur with equal probability throughout the region
* There are no existing sensors in the region
* The probability of detection by each sensor is independent of other sensors
* Sensors are ideal in the sense that there is no risk of a false alarm

Relaxation of these assumptions will be the topic of future research. Since traditional nonlinear programming techniques like SQP were shown to be inferior on the basic SELP, other solution methods should be explored. Also, because TLP relies on the geometry of the SELP, the method will fail when complications, such as non-identical sensors, are introduced. The purpose of this paper is to explore the viability of EAs as reliable solvers for the SELP, in hopes that they will also be appropriate when some of the simplifying assumptions are relaxed. As in the previous two papers, this study will consider the continuous problem with events occurring within a square.

2.2 Problem Formulation

By assumption, the performance of a sensor depends only on the distance between the event and the sensor. As such, the detection capability of a single sensor can be modeled by a detection probability function (dpf) such as a gravity decay function of the form $\pi(d) = 1 - e^{-k/d^n}$, or an exponential decay function of the form $\pi(d) = e^{-kd^n}$, where $k>0$ and $n>0$ are parameters of the sensor and d is the Euclidean distance between the event and the sensor. Using these functions, we can now consider the capability of an entire sensor system. The joint probability of non-detection by the system for an event located at $\mathbf{z}$ is the product of the individual probabilities of non-detection: $\prod_{i=1}^{m}(1-\pi(d(\mathbf{x}_i,\mathbf{z})))$, where $\mathbf{x}_i$ is the location of sensor i, $d(\mathbf{x}_i,\mathbf{z})$ is the Euclidean distance between sensor i and the event, and m is the number of sensors in the system. Then the objective function is:

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_m} \max_{\mathbf{z}} \prod_{i=1}^{m}(1-\pi(d(\mathbf{x}_i,\mathbf{z})))$$

The only constraint on the problem is that the sensors remain within the region, S. For the specific case where S is a square, this can be written as $\mathbf{x}_i \in S$ where $S=\{\mathbf{x}_i=(x_i,y_i) \mid x_l \le x_i \le x_u, y_l \le y_i \le y_u\}$ for all $i=1, 2,\ldots, m$ and x_l and x_u and y_l and y_u are the lower and upper bounds, on the x-coordinate and y-coordinate respectively, which define the square.

Within DE, a simple feasibility check was performed on each sensor location. If a sensor fell outside the region, it was repaired by forcing violated constraints to be tight. That is, the infeasible coordinate was set to the closest boundary. For example, if a sensor has coordinates $\mathbf{x}_i=(x_i > x_u, y_l \le y_i \le y_u)$, it was repaired with the following coordinates: $\mathbf{x}_i'=(x_u, y_l \le y_i \le y_u)$ This Lamarckian approach of repair and replace was chosen for its simplicity and ease of implementation. Lamarckian approaches can lead to a lack of diversity within EAs, but preliminary testing revealed that DE worked well with this implementation. Thus, more complicated strategies were not explored.

When the above mentioned Lamarckian approach was implemented in CMA-ES, the algorithm failed to reliably solve the SELP. Variance in the best objective value (fitness) and the number of function evaluations were high. As a result, the implementation in CMA-ES uses a Baldwinian adaptive penalty method as recommended in [6]. In this method, the repaired solution is used to evaluate the fitness of the individual and then the "repaired fitness" is penalized by adding a weighted distance between the population member and the repaired individual. Note the key difference is that the repaired individual does not replace the infeasible population member. It is only used in evaluating the fitness of the individual. This boundary handling method was already coded in version 2.50 of CMA-ES, which was used for this study. For more details on the boundary handling, see [6].

2.3 Test Problems

This paper considers locating 2 to 10 sensors in the unit square, as studied by Drezner and Wesolowsky [5]. They considered sensor capabilities modeled by the following exponential decay detection probability function $\pi(d) = e^{-d}$. For simplicity, this will be referred to as the Drezner problem. As in their study, in order to approximate the continuous objective function, the event locations were discretized using a 10 by 10 grid, but sensor locations remain continuous decision variables. Figure 1 shows the probability of non-detection of a single sensor for the Drezner problem.

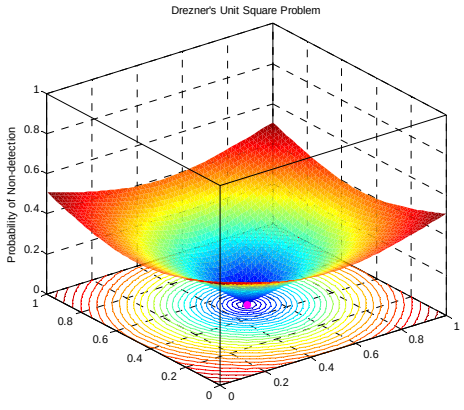

Figure 1: Exponential Decay for Drezner's problem

3. PARAMETERIZING EAs

Often, EAs have a sweet spot, or range, of parameter settings within which the algorithm can solve a particular problem well. This is because the EA must balance the need to explore the search space and exploit good solutions. If too much exploration is done with low selective pressure, the algorithm will follow a random walk, also known as drift. The algorithm will eventually converge but not necessarily to a good solution. On the other hand, if selective pressure is high, even if mutation and crossover are high, the solutions converge quickly and innovation will be impossible. Thus finding the right balance to parameterize evolutionary algorithms is important, but can be a time-consuming process.

In a 2004 paper, Reed and Yamaguchi [12] suggest a three-step method to avoid the trial-and-error approach of parameter setting. The three steps are 1.) consider the time constraints of your problem, 2.) automatically set mating/mutation parameters based on available literature, and 3.) implement adaptive population sizing. Adaptive population sizing means that initially, an arbitrarily small population is used. After certain intermediate stopping criteria are met, the population size is doubled and the algorithm is implemented again with the larger population. In addition, the best solution found in the previous run is injected into the larger population. This process repeats until there is little or no improvement in the objective value or a maximum time or number of function evaluations is reached. Reed and Yamaguchi implemented their method using DE.

Similarly, in [1], Auger and Hansen suggest an adaptive population sizing approach for CMA-ES. Like the Reed and Yamaguchi approach, default parameters settings are chosen based on suggestions in the literature. Then the algorithm is implemented with a small population size. After certain stopping criteria, based on convergence, are satisfied, the population size is doubled and the process is repeated until a pre-defined number of function evaluations is reached. However, because CMA-ES is not an elitist EA, in the Auger and Hansen implementation, no information from the small population runs were injected into the larger populations.

Because of the time involved in finding the optimal parameter settings for EAs and the fact that several variants of each algorithm are implemented, a similar adaptive population sizing was used in this paper. It is important to note that using this approach implicitly assumes that population size is the parameter with the largest impact on EA performance. Evidence from [1] and [12] support this assumption for both DE and CMA-ES.

3.1.1 Preliminary Analysis

On a Dell XPS MXC051 PC with a 2 GHz Pentium M processor, using a 10 by 10 grid to evaluate the probability of non-detection for one configuration of 10 sensors requires approximately 0.0012 seconds. The maximum acceptable run time is determined to be 5 minutes ≈ 250,000 evaluations, since multiple variations of each EA will be solved for many random seeds. Because each algorithm will require different computation time, the maximum number of function evaluations was used to stop each algorithm instead of a maximum cpu time. Note that this limit exceeds the overall stopping criteria set for CMA-ES in [1], which was $10^4 D$ function evaluations, where D is the problem dimension.

Because the optimal solution to the SELP is unknown, the search was terminated when doubling the population size no longer improved the objective value or 250,000 function evaluations had elapsed. Improvement is measured by a greater than 1% decrease in the objective function value between runs, where "run" refers to the process of doubling the population size. The arbitrarily small initial population size was chosen to be 10. This is the value used in [12] and is also the recommended default population size for a 10 dimensional problem in CMA-ES [8].

The overall stopping criteria of 250,000 function evaluations or no improvement in the objective value was implemented for both DE and CMA-ES, but the internal stopping criteria was specific to each algorithm. The specific implementations will be discussed in the respective sections for each algorithm. Also note that time continuation was used in both algorithms. In the case of DE, the best population member from the previous run was injected into the initial population of the next run. In CMA-ES, the "best ever" solution was tracked in each run and used as the mean of the initial population in subsequent runs.

4. DIFFERENTIAL EVOLUTION (DE)

4.1 Description

Differential Evolution is a search algorithm developed by Storn and Price [13] used to solve continuous space minimization problems. Decision variables are represented as real-valued vectors with dimension D. Like most EAs, Differential Evolution uses mutation and recombination operators to explore the search space and a tournament selection operator to exploit strong population members. The details of the algorithm follow, using the notation and terminology of Storn and Price.

Begin by generating a parent population with $\mu=NP$ members of D-dimensional vectors by random sampling from a uniform distribution in the feasible decision space. Each parent, also called a target vector, is denote $x_{i,G}$, where i=1, 2,..., NP and G denotes the generation. A child population of size $\lambda=NP$ is created through mutation and crossover. Mutation involves adding the weighted difference of two randomly chosen population members to a third randomly chosen individual (called the base of the mutation), where all of the individuals involved are different. The weight is called a scaling factor, denoted $F \in (0,2]$. Next, uniform crossover occurs between the mutant vector and a parent from generation G, but with one randomly chosen parameter forced to be chosen from the mutant vector. Thus the parent and the child differ by at least one decision. Cross-over is controlled by CR, the probability that the jth parameter (decision variable) will be drawn from the mutant vector as opposed to the parent vector. The resulting child, called the trial vector, competes with the parent to determine which individual moves on to the next generation. The process is repeated until the population converges or a maximum number of generations is reached. The MATLAB code for DE provided at [14] was modified as necessary to solve the SELP. The mathematical representation of the mutation. crossover, and selection operators follow:

4.1.1 DE's Mutation

The mutant vector denoted $v_{i,G+1}$, is determined by:

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G})$$

where $r1, r2, r3$ are distinct, randomly chosen indices from (1, 2, …, NP).

4.1.2 DE's Cross-over

The trial vector, or child, is denoted by $u_{i,G+1}$ and each element (decision) of the vector is determined by:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, \text{if } (rand(j) \le CR) \text{ or } j = randindex(i) \\ x_{ji,G+1}, \text{if } (rand(j) > CR) \text{ and } j \ne randindex(i) \end{cases}$$

where $CR \in [0,1]$, *rand(j)* is a random number from U[0,1], and *randindex(i)* is a random index in (1, 2, .., *D*) so that at least one of the mutated parameters is present in the trial vector.

4.1.3 DE's Selection

Selection occurs via binary tournaments between the target vector, or parent, $x_{i,G}$ and the trial vector, or child, $u_{i,G+1}$. If fitness($u_{i,G+1}$) < fitness($x_{i,G}$), then in generation *G+1*, $x_{i,G+1} = u_{i,G+1}$. Otherwise, $x_{i,G+1} = x_{i,G}$. Note that this approach is elitist.

4.2 Methodology

4.2.1 Default Parameter Settings

After testing DE on three different suites of test functions, Storn and Price [13] suggest *F*=0.5 as a good first choice for the scaling factor and *CR* = 0.9 for the probability of cross-over when possible. Since the population size will be adaptive, *NP* is no longer a parameter set by the user. The initial population will be drawn from a uniform distribution within *S*.

4.2.2 Adaptive Population Sizing

Again, following the suggestions of Reed and Yamaguchi [12], the dynamic population sizing was implemented as follows:

1. Set *i*=0 (run 0)
2. Set the population size to NP_0=10
3. Allow the population to mate and mutate until 3.a. or 3.b. occur
   - 3.a. The population converges, as measured by:
   $$\frac{100^* | Fitness_{Best} - Fitness_{Worst} |}{Fitness_{Best}} \le 1$$
   - 3.b. At least $t_{min} = NP_i * D$, where *D* is the problem dimension, generations have passed and there is less than 1% improvement in the objective between generations
4. Double the population size ($NP_{i+1} = 2NP_i$). Include the best population member from run *i* in the new population of run *i+1*. Set *i= i+1*. Return to step 3.

The search was terminated when doubling the population size did not result in at least a 1% improvement of the objective function or the function evaluation limit (250,000) was reached.

4.2.3 Reliability

To ensure that the three-step implementation would reliably solve the SELP, 50 trial runs (random seeds) of the Drezner problem were investigated when locating 2, 5, and 10 sensors. The results are presented in Figures 2 and 3 and Table 1. Figure 2 plots the progression of the best fitness value versus number of function evaluations for the 2, 5, and 10 sensor problems. Note that the scales, in terms of objective value and number of function evaluations, vary with problem dimension. Figure 3 plots the frequency, over the 50 random seeds, of the maximum population size required under adaptive population sizing.

As can be seen by examining Table 1, over 50 trials, the percent difference between the best and worst solutions, calculated as $100 * (Worst - Best) / Average$, was less than 5% for all three problems. Because this measure relies on the entire range of objective values, it can be thought of as a worst case performance metric. The standard deviation as a percent of the average fitness is less than 2% for all three problems. Thus DE with adaptive population sizing yields reliable results for the SELP. The maximum population size reached was 160 for the 10 sensor problem.

Table 1: DE Reliability Results for the Drezner Problem

Sensors	Average Fitness (std. dev.)	Range of Fitness Values	Average Function Evaluations (std. dev.)
2	0.255243 (0.000642)	(0.254342, 0.256989)	1625.4 (931.0)
5	0.017525 (0.000034)	(0.017477, 0.017633)	23224.8 (21816.3)
10	0.000232 (0.000003)	(0.000229, 0.000240)	191638.4 (74264.5)

4.3 Variants of DE

Using the implementation described in section 3.2, eight variants of DE were studied to determine whether a significant improvement in solution quality and/or function evaluations could be obtained over the basic form of DE. Table 2 lists the 8 variants under consideration, where "Pairs" indicates crossover according to coordinate pairs rather than individual parameters. See section 4.3.2 for more details.

Pairs, DE/best/1/bin	DE/best/1/bin
Pairs, DE/rand/1/bin	DE/rand/1/bin
Pairs, DE/best/2/bin	DE/best/2/bin
Pairs, DE/rand/2/bin	DE/rand/2/bin

Table 2: Table of DE Variants

4.3.1 Common DE Variants

In addition to the basic DE presented earlier, several variations exist. The basic version of DE is denoted DE/rand/1/bin which is read as Differential Evolution with a randomly chosen base for mutation, one difference vector involved in mutation, and binary experiments during crossover.

The other variants tested in this study are: DE/best/1/bin, DE/rand/2/bin, and DE/best/2/bin. Best indicates that the most fit individual from the previous generation will be chosen for the

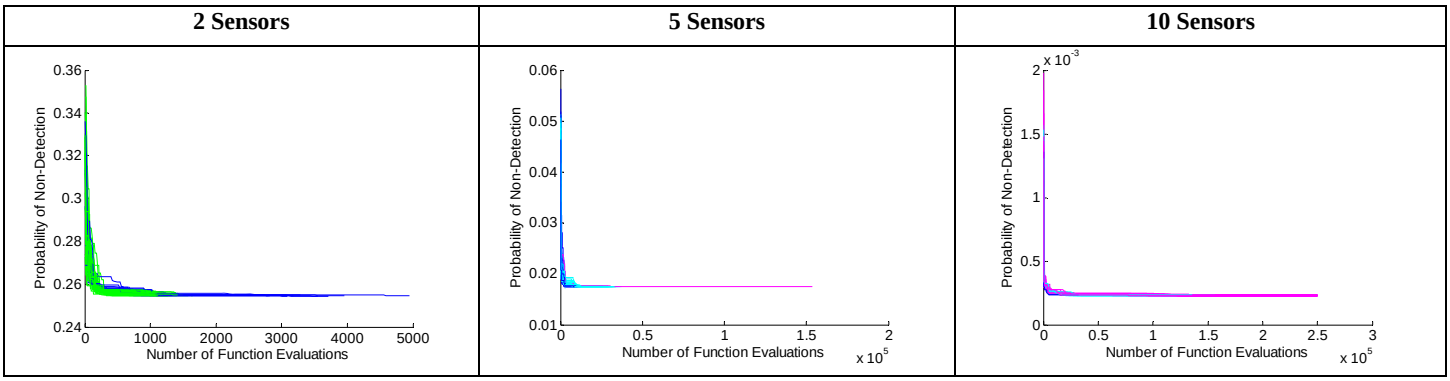

Figure 2: DE Dynamics Plots of Objective Value versus Function Evaluations over all random seeds for 2, 5, and 10 sensors

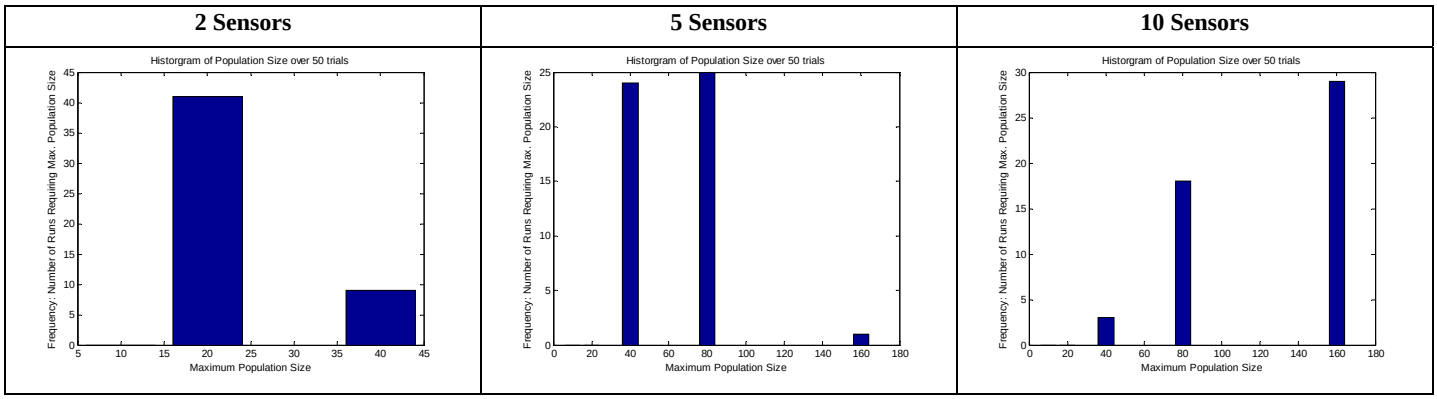

Figure 3: Frequency of Random Seeds and Maximum Population Size Required for 2, 5, and 10 sensors

base of the mutation in the next generation. The 2 indicates that two difference vectors will be involved in mutation as follows:

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} + x_{r3,G} - x_{r4,G} - x_{r5,G})$$

4.3.2 *Pair-wise Cross-over*

In addition to the four variants mentioned, a variation in the cross-over mechanism, specific to the SELP, is considered. Note that in the usual implementation of DE, each individual parameter undergoes a binary experiment to determine whether the parameter will come from the parent vector or the mutated vector. But in the SELP, individual parameters are actually part of a coordinate pair which together form a sensor location. Thus, intuition suggests that crossover should be done according to sensor locations (coordinate pairs) rather than individual parameters. Thus for each variant listed above, crossover will be performed two ways: 1.) cross-over using individual parameters (as usual) and 2.) cross-over as coordinate pairs. The mathematical representation of cross-over in pairs follows:

The trial vector, or child, is denoted by $u_{i,G+1}$ and each *sensor location* $(u_{ji,G+1}, u_{j+1i,G+1})$ of the vector is determined by:

$$\begin{cases} (v_{ji,G+1}, v_{j+1i,G+1}) \text{ if } (rand(j) \le CR) \text{ or } j \text{ or } j+1 = randind(i) \\ (x_{ji,G+1}, x_{j+1i,G+1}) \text{ if } (rand(j) > CR) \text{ and } j,\ j+1 \ne randind(i) \end{cases}$$

where $CR \in [0,1]$, *rand(j)* is a random number from U[0,1], and *randindex(i)* is a random index in (1, 2, …, *D*) so that at least one of the mutated sensor locations is present in the trial vector.

4.3.3 *Results*

Each variant was run over 50 random seeds for the 2, 5, and 10 sensor Drezner problem. The results of the DE variant testing are summarized in Figure 4. The left column plots the mean objective value attained by each variant, along with the 95% confidence interval around the mean, calculated as $Mean \pm 1.96\left(\frac{St.Deviation}{\sqrt{NumberOfSeeds}}\right)$. The right column plots the mean number of function evaluations and the 95% confidence interval around the mean. The dashed bars indicate the variant which achieved the minimum function value in the minimum number of function evaluations. The DE/best/2/bin variants outperformed other variants, in terms of fitness value, when function evaluations are used to break "statistical" ties between the means. However, no determination could be made as to whether "pair-wise" crossover was beneficial. In all three problems, the DE/best/2/bin and Paired-DE/best/2/bin performed the same in terms of function evaluations and objective value. This is likely due to the high value of the crossover parameter (CR=0.9). At this setting, most of the offspring's elements are chosen from the mutant vector rather than the parent vector. Therefore the high probability of choosing parameters from the mutant vector as opposed to the parent vector may mask the effect of the pair-wise cross-over mechanism. The recommended variant of DE is DE/best/2/bin with crossover according to individual decision variables.

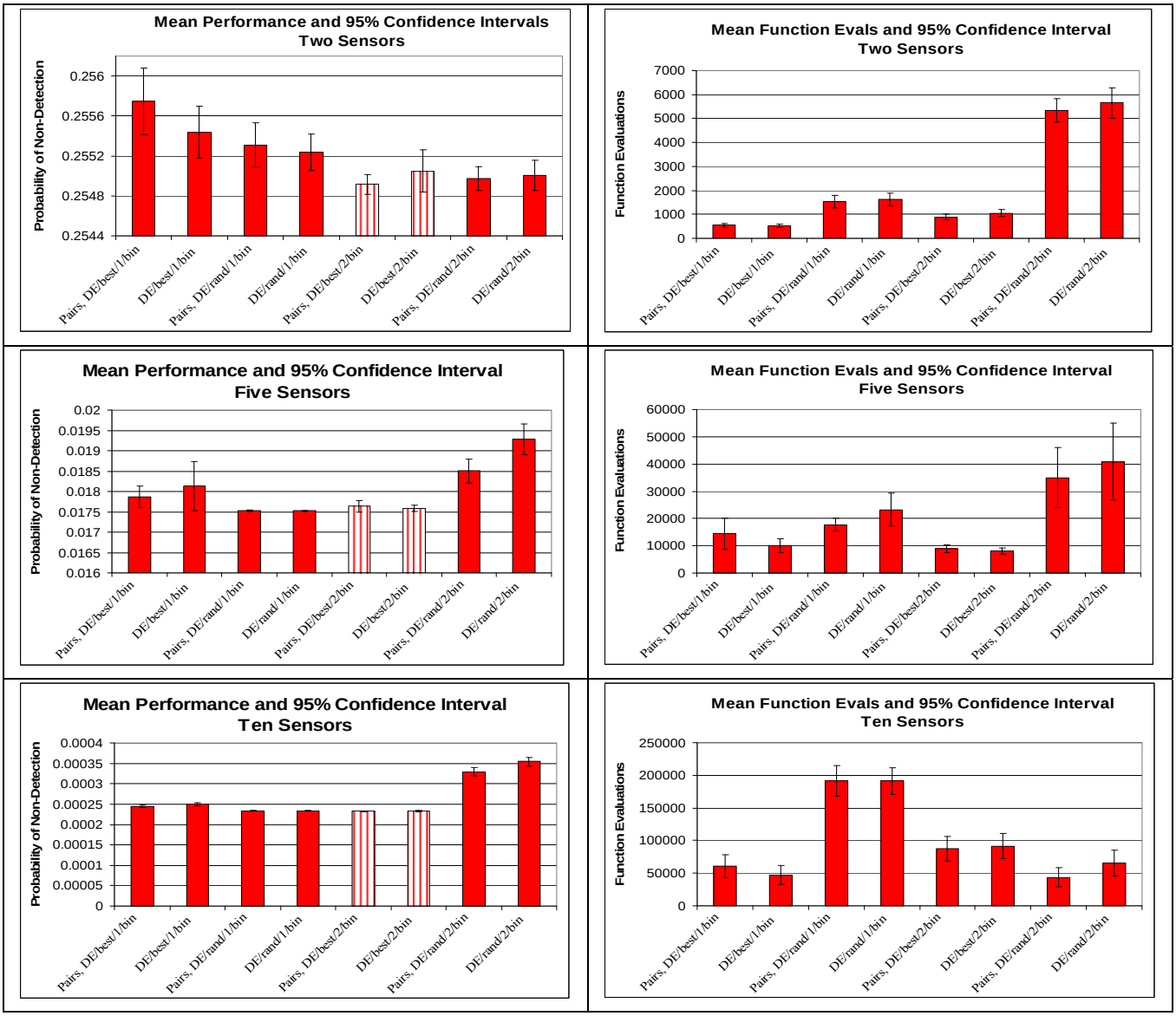

Figure 4: DE Variants. Mean Objective Value with 95% CI and Mean Function Evaluations with 95% CI for 2, 5, and 10 sensors

5. COVARIANCE MATRIX ADAPTATION

5.1 Description

Co-variance Matrix Adaptation is a (μ, λ) evolutionary strategy developed by Hansen and Ostermeier [9]. As suggested by the (μ, λ) notation, this ES is non-elitist and uses truncation selection. Like typical ESs, CMA-ES's recombination operator consists of calculating the weighted mean of μ best parents. An unbounded mutation operator is used, which means that a random vector is drawn from a multivariate normal distribution with zero mean and added to the centroid of the parents. CMA-ES dynamically adapts the normal mutation distribution throughout its search and the standard deviation, or step size, of the normal distribution represents the strength of the mutation operator. The method is appropriate for non-linear, non-separable objective functions and because of its adaptive step size, is unlikely to pre-converge [7]. The main components of the algorithm are described below, using the notation and terminology of Hansen and Ostermeier [9].

The algorithm begins by sampling a new population as described in secion 5.1.2. Next, selection and recombination take place. Based on the selected individuals, the evolution paths for the covariance matrix and the step size are updated. Finally, the covariance matrix, **C**, and the step size, σ are adapted as described in sections 5.1.4 and 5.1.5. CMA-ES was implemented in MATLAB using version 2.50 of the code available at [15]. Note that while CMA-ES is not elitist, the best solution ever found was tracked in an archive and used for all comparisons.

5.1.1 CMA-ES Mutation

The offspring population is generated according to the following relationship:

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)} N(\mathbf{0}, \mathbf{C}^{(g)}) \quad \text{for } k = 1, 2, .., \lambda$$

Where $\mathbf{m}^{(g)}$ is the mean vector as calculated in 5.1.3, $\sigma^{(g)}$ is the adapted step size, and $\mathbf{C}^{(g)}$ is the adapted covariance matrix.

5.1.2 CMA-ES Selection and Recombination

The mean vector is calculated as the weighted sum of the μ best population members according to:

$$\mathbf{m}^{(g)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_i^{(g)}$$

$$\text{where } \sum_{i=1}^{\mu} w_i = 1, \text{ and } w_1 \geq w_2 \geq \ldots \geq w_\mu > 0$$

When the weights are equal, i.e. $w_1 = w_2 = \ldots w_\mu = 1/\mu$, the operator is called "intermediate" recombination, denoted (μ_I, λ). When more fit individuals receive a higher weight than less fit individuals, the operator is termed "weighted recombination," denoted (μ_W, λ).

5.1.3 CMA-ES Covariance Matrix Adaptation

The basic assumption behind adapting the mutation distribution is that steps which were successful in the past are likely to be successful in the future. In light of this intuitive assumption, the covariance matrix is built using the cumulation (sum) of mutation steps from previous generations, also called a covariance evolution path, in order to increase the odds of reproducing the previous successful steps. The adaptation of the covariance matrix is similar to the updating of the Hessian matrix in quasi-Newton methods [9]. That is, the covariance matrix can be thought of as an empirically calculated Hessian, which uses information about the gradient of the objective function to adapt the search. See [7] for implementation details.

5.1.4 CMA-ES Cumulation Step Size Adaptation

In order to adapt the mutation step size, CMA-ES tracks the cumulation (sum) of successive steps, called the step size evolution path, and compares its length to the expected length of the path under random selection. When the actual path length exceeds the expected path length, the step size is increased. Otherwise, the step size is decreased. The rationale behind this process follows: If successive steps are parallel correlated (i.e. point in the same direction), then fewer but longer steps could have reached the same point faster. The resulting cumulation path will be longer than expected, indicating that the step length should be increased. If successive steps are anti-parallel correlated (i.e. point in opposite directions), then the steps counteract one another and smaller steps would have been more efficient. Similarly, the cumulation path will be shorter than expected, indicating that the step size should be decreased. Implementation details of this process can be found in [7].

5.2 Methodology

5.2.1 Default Parameter Settings

CMA-ES has several parameters which must be specified by the user, including μ, λ, and several parameters related to the adaptation of the normal mutation distribution. These were set using the default parameter values listed in [8], with the exception that the initial population size is set to λ = 10 (regardless of problem dimension) because of the adaptive population sizing. Table A in the Appendix lists these parameter settings. Similar to the initialization of DE, the mean was drawn from a uniform distribution on $[0,1]^D$, where D is the problem dimension, and the initial step size was set, as suggested in [1], at $\sigma^{(0)} = \frac{1}{2}(b-a)$, where (a,b) = (0,1) for the unit square problem.

5.2.2 Adaptive Population Sizing

Again, following the suggestions of Reed and Yamaguchi [12], dynamic population sizing with time continuation was implemented. However, the intra-run convergence criteria were expanded to include the criteria suggested in [1]. These criteria are specific to CMA-ES and all were used, except the criteria that the range of function values in a generation remain above 10^{-12}. Instead, this criterion was replaced with the convergence measure listed in the implementation scheme below.

Thus, the adaptive population sizing was implemented as:

1. Set i=0 (run 0)
2. Set the population size to λ_0=10
3. Allow the population to mate and mutate until 3.a. or 3.b. occur
   - 3.a. The population converges, as measured by: $\frac{100^* | Fitness_{Best} - Fitness_{Worst} |}{Fitness_{Best}} \leq 1$ or one of the intra-run convergence criteria in [1] are met.
   - 3.b. At least $t_{min} = \lambda_i * D$, where D is the problem dimension, generations have passed and there is less than 1% improvement in the objective between generations
4. Double the population size ($\lambda_{i+1} = 2\,\lambda_i$). Set the mean of the new population in run $i+1$ equal to the best known solution from previous runs. Return to step 3.

The search was terminated when doubling the population size did not result in at least a 1% improvement of the objective function or the function evaluation limit (250,000) was reached.

5.2.3 Reliability

To ensure that the adaptive population sizing of CMA-ES would reliably solve the SELP, 50 trial runs (random seeds) of the Drezner problem were investigated when locating 2, 5, and 10 sensors. The reliability results are based on the weighted recombination variant of the algorithm and are presented in Figures 5 and 6 and Table 3. Figure 5 plots the progression of the best fitness value versus number of function evaluations for the 2, 5, and 10 sensor problems. Note that the scales, in terms of objective value and number of function evaluations, vary with problem dimension. Figure 6 plots the frequency, over the 50 random seeds, of the maximum population size required under adaptive population sizing.

Table 3: CMA-ES Reliability Results for the Drezner Problem

Sensors	Average Fitness (std. dev.)	Range of Fitness Values	Average Function Evaluations (std. dev.)
2	0.255484 (0.000899)	(0.254303, 0.25715)	891.16 (422.2)
5	0.017515 (0.0001)	(0.017447, 0.017962)	4888.8 (1863.0)
10	0.000235 (0.000005)	(0.000227, 0.000252)	20400.76 (20026.5)

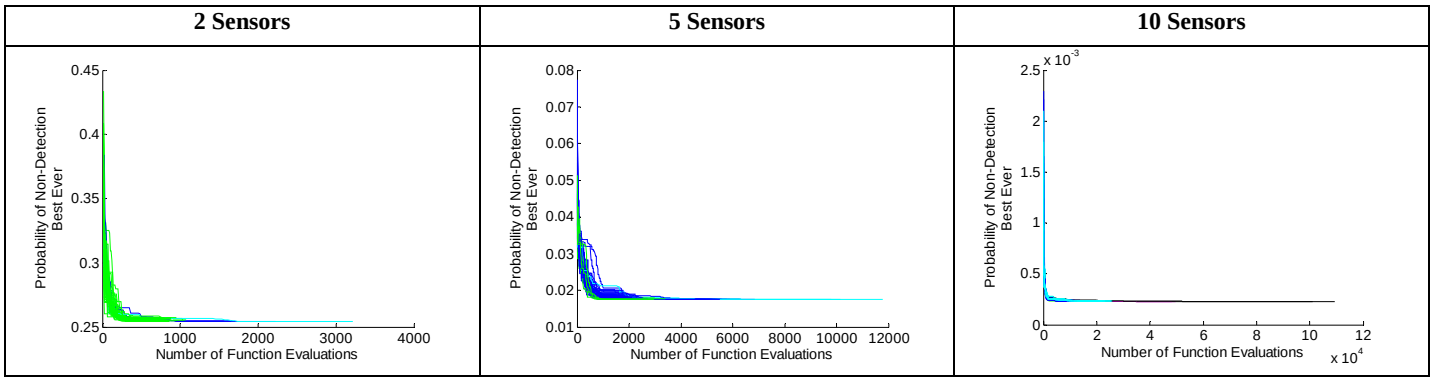

Figure 5: CMA-ES Dynamics Plots of Objective Value versus Function Evaluations over all random seeds for 2, 5, and 10 sensors

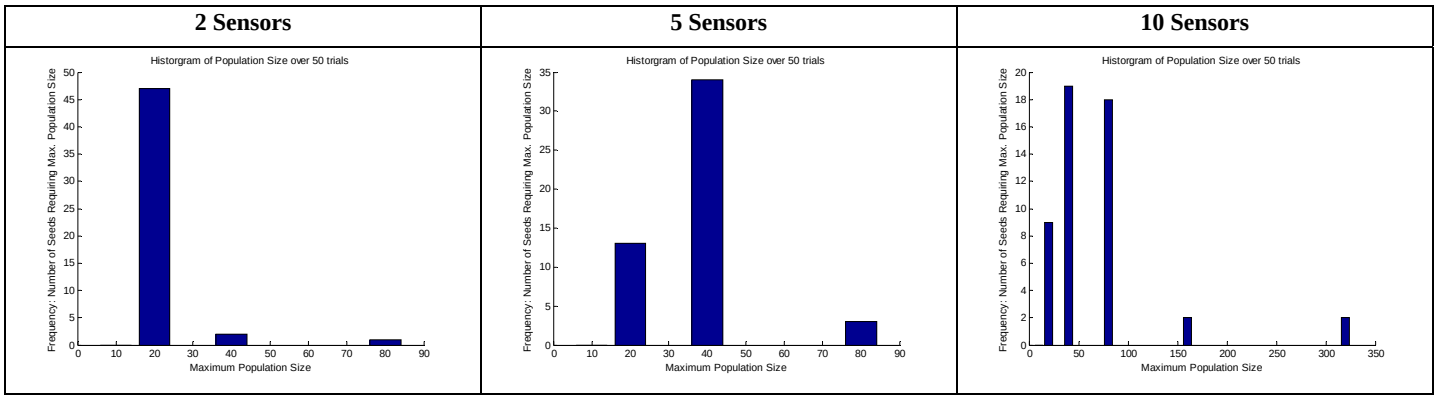

Figure 6: Frequency of Random Seeds and Maximum Population Size required for 2, 5, and 10 sensors

As can be seen by examining Table 3, over 50 trials, the percent difference between the best and worst solutions, calculated as $100 * (Worst - Best) / Average$, was less than 11% for all three problems. Recall again that this measure relies on the entire range of objective values and therefore can be thought of as a worst case performance metric. The standard deviation as a percent of the average fitness is less than 3% for all three problems. Thus, while not as reliable as DE, CMA-ES with adaptive population sizing still yields reliable results for the SELP. The maximum populations size required by CMA-ES was 320 for the 10 sensor problem.

5.3 Variants of CMA-ES

As described in section 5.1.3, both the intermediate recombination and weighted recombination variations of CMA-ES were implemented on the Drezner Problem. For (μ_I, λ) CMA-ES, the weights are given by: $w_1 = w_2 = \ldots = w_\mu = \frac{1}{\mu}$. For ($\mu_W$, λ) CMA-ES, the weights are given by: $w_i = \frac{\ln(\mu+1) - \ln(i)}{\sum_{j=1}^{\mu} \ln(\mu+1) - \ln(j)} \forall i.$

5.3.1 Results

Both variants were run over 50 random seeds for the 2, 5, and 10 sensor Drezner problem. The results of the CMA-ES variant testing are summarized in Figure 7 and Table 4. Figure 7 plots the mean objective value attained by each variant, along with the 95% confidence interval around the mean, calculated as $Mean \pm 1.96\left(\frac{St.Deviation}{\sqrt{NumberOfSeeds}}\right)$. Table 4 shows the 95% confidence interval on the difference of the mean objective value and the mean function evaluations of the two variants, calculated as $(Mean_1 - Mean_2) \pm 1.96\left(\sqrt{\frac{StDeviation_1^2}{\#OfSeeds_1} + \frac{StDeviation_2^2}{\#OfSeed_2}}\right)$. As can be seen from Figure 7 and Table 4, the difference in the means of the objective value is not statistically different than zero. However, the .difference in the means of the function evaluations is statistically significant. This means that in terms of the objective value, both variants work equally well. But, the weighted recombination variant requires significantly fewer function evaluations when compared with the intermediate recombination variant. The recommended variant of CMA-ES is the weighted recombination variant.

Table 4: 95% Confidence Intervals on the Difference in Mean Performance for Intermediate versus Weighted Recombination

Sensors	CI on the Difference: Mean Objective Value	CI on the Difference: Mean Function Evaluations
2	(-6.91 E-04, 3.17 E-05)	(-454.37, -140.18)
5	(-4.94 E-05, 5.94 E-05)	(-6744.35, -4430.68)
10	(-1.64 E-07, 4.16 E-06)	(-55872.07, -26181.28)

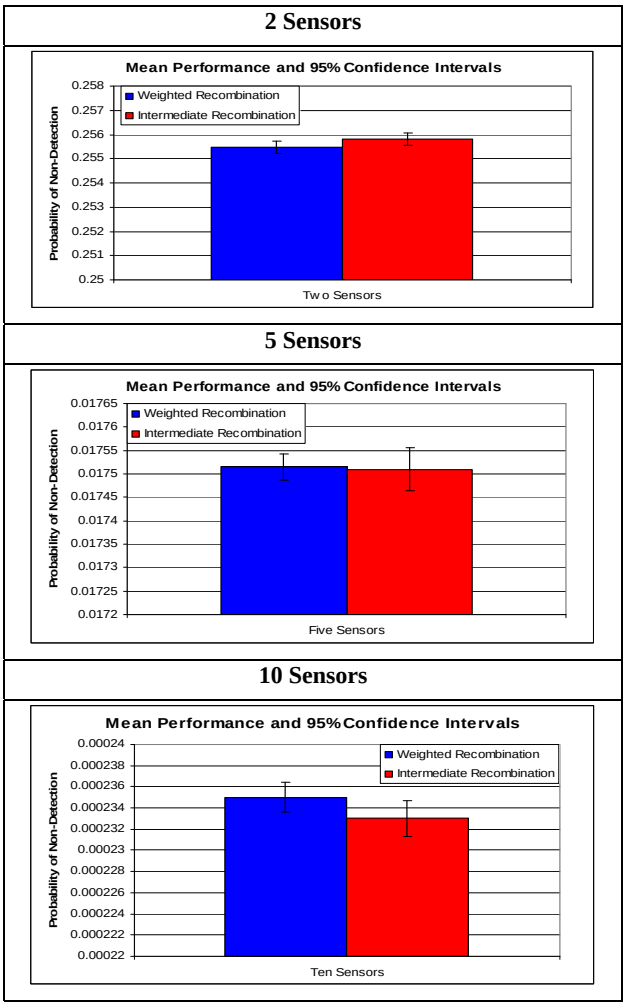

Figure 7: Mean Objective and 95% CI for CMA-ES variants

6. COMPARISON

6.1 Methodology

Using the DE/best/2/bin variant of DE and the weighted recombination variant of CMA-ES, the random seed analysis from the previous sections was employed to determine whether one algorithm outperformed the other both in terms of quality of the objective value and the number of function evaluations.

6.2 Results

Figure 8 and Table 5 summarize the comparison of DE and CMA-ES. The left column of Figure 8 shows the mean objective value obtained by each algorithm, along with the 95% confidence interval on the mean. The right column shows the same information with respect to the mean number of function evaluations required. Table 5 lists the 95% confidence interval on the difference in the mean objective value and number of function evaluations from the two algorithms.

In terms of statistical significance, Table 5 reveals that for the 2 and 10 sensor problems, DE, on average, finds solutions with better objective values than CMA-ES. For the 5 sensor problem, no difference in the mean objective values could be distinguished. However, depending on the exact application and the desired level of precision in the probability of non-detection, the difference in the objective value found by CMA-ES versus DE may not have any practical significance. For example, in the two sensor problem, the average probability of non-detection found by DE was 0.255049, and the average found by CMA-ES was 0.255484. These values are different in terms of statistical significance, but if the application only requires a probability to the thousandths, i.e. 0.255, then the solutions are not different in terms of "practical significance."

Table 5: 95% Confidence Intervals on the Difference in Mean Performance for best variants of DE and CMA-ES

Sensors	CI on the Difference: Mean Objective Value	CI on the Difference: Mean Function Evaluations
2	(-7.63 E04, -1.07 E04)	(-23.79, 352.27)
5	(-3.15 E06, 1.47 E04)	(2102.75, 4572.45)
10	(-3.96 E06, -4.00 E08)	(50927.67, 90899.61)

Figure 8 and Table 5 also reveal that for the 5 and 10 sensor problems, CMA-ES requires significantly fewer function evaluations than DE. And, as problem dimension increases, this difference becomes more pronounced. This can be explained by noting that the SELP is a highly non-separable problem since 1.) each parameter is part of a coordinate pair of a single sensor location and 2.) each sensor has some degree of detection capability in the vicinity of other sensors. According to [7], CMA-ES can improve performance on non-separable problems by orders of magnitude because it uses information from the covariance matrix, a technique akin to using the Hessian matrix of a quasi-Newton method, to guide search. Note that for more difficult problems, when a better-quality estimate of the probability of non-detection requires a finer grid in discretizing event locations, computation time of the function evaluations will increase. In this case, CMA-ES will be more advantageous because it saves functional evaluations.

7. CONCLUSIONS AND FUTURE WORK

This paper applied CMA-ES and DE to the minimax sensor location problem. Of the variants tested, the DE/best/2/bin and weighted recombination variants performed the best in terms of objective value and/or number of function evaluations. Statistical significance and practical significance should be considered when determining whether DE and CMA-ES yield different results in terms of the objective value. If a high degree of precision is required, DE will yield better solutions in terms of the objective value. But, if precision after the thousandth decimal is irrelevant, then CMA-ES can give equivalent results in fewer function evaluations.

Future work will investigate solving higher dimensional problems with 20, 50, or even more sensors, modeled with different detection probability functions. As problem dimension increases and/or the grid used to estimate the probability of non-detection becomes finer, CMA-ES is expected to become more advantageous because it uses fewer function evaluations. Futurer work will also investigate problems in which previous heuristics, like TLP, will fail. (e.g. problems with non-identical sensors).

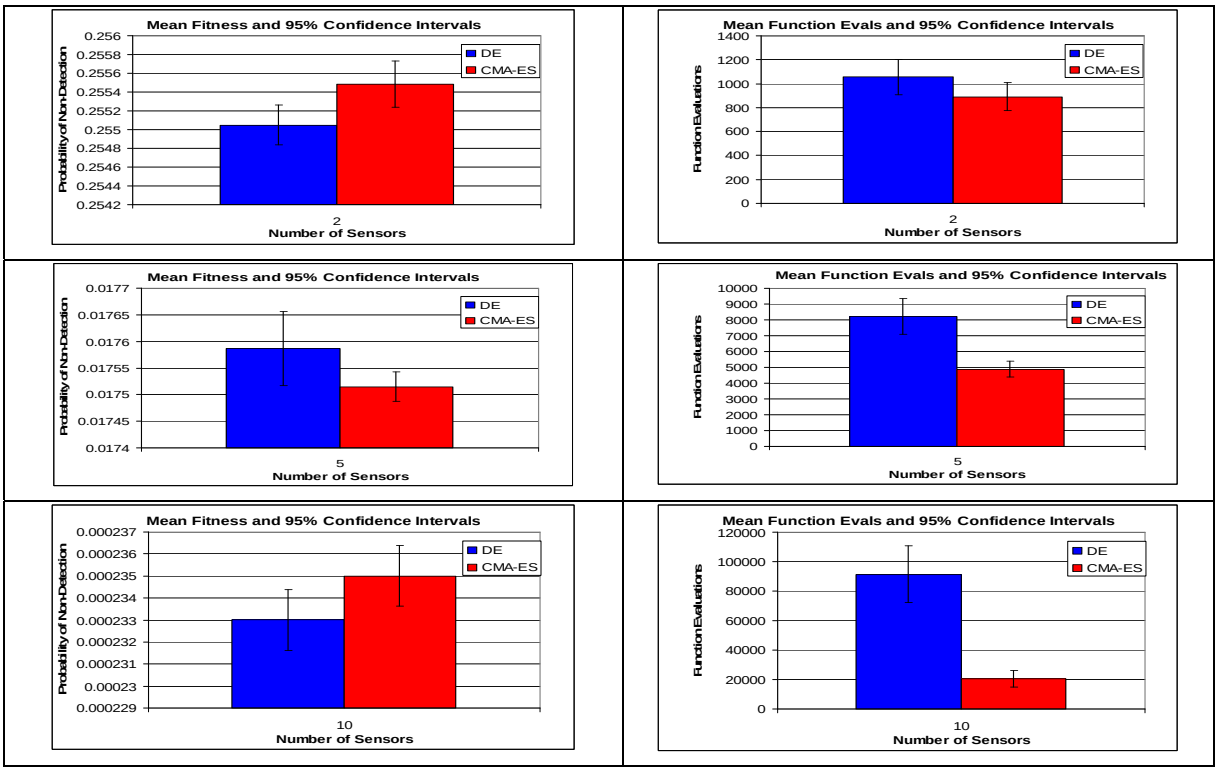

Figure 8: Mean Objective and Mean Function Evaluations with 95% CIs for the best variants of DE and CMA-ES

8. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. 0400140. Thanks to Dr. Reed for his lectures on Evolutionary Algorithms.

9. REFERENCES

[1] Auger, A. and Hansen, N.; "A Restart CMA Evolution Strategy With Increasing Population Size," *Proc. of the IEEE International Conference on Evolutionary Computation*, **2005**, 1769-1776.

[2] Cavalier, T.; Conner, W.; del Castillo, E.; Brown, S.; "A Heuristic Algorithm for Minimax Sensor Location in the Plane," *European Journal of Operations Research*, accepted **2006**.

[3] Dhillon, S.; and Chakrabarty, K.; "Sensor placement for effective coverage and surveillance in distributed sensor networks," *Wireless Commuications and Networking*, **2003**, 1609-1614.

[4] Dhillon, S.; Chakrabarty, K.; and Iyengar, S.; "Sensor placement for grid coverage under imprecise detection," *International Conference on Information Fusion*, **2002**, *2*, 1581-1587.

[5] Drezner, Z. and Wesolowsky, G.; "On the best location of signal detectors," *IIE Transactions*, **1997**, *29*, 1007-1015.

[6] Erni, Dominik; "Boundary Handling Methods for CMA Evolution Strategy and PSO Algorithm," Semester Thesis, Institute of Computational Science, February 21, 2005.

[7] Hansen, Nickolas, "The CMA Evolution Strategy: A Tutorial", November 11, **2005**, http://lautaro.bionik.tu-berlin.de/user/niko/cmatutorial.pdf

[8] Hansen, N. and Kern, S.; "Evaluating the CMA Evolution Strategy on Multimodal Test Functions," *Parallel Problem Solving from Nature*, **2004**, 1-10.

[9] Hansen, N. and Ostermeier, A.; "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation," *Evolutionary Computation,* **2001**, 9(2), 159-195.

[10] Hansen, N. and Ostermeier, A.; "Completely derandomized self-adaptation in evolutionary strategies," *Proc. of the IEEE International Conference on Evolutionary Computation*, **1996**, 312-317.

[11] Krishnamachari, Bhaskar; *Networking Wireless Sensors*, Cambridge University Press **2005**.

[12] Reed, P. and Yamaguchi, S.; "Simplifying the Paramterization of Real-Coded Evoluationary Algorithms," *ASCE World Water and Environmental Resources Congress*, Salt Lake City, Utah, June **2004**.

[13] Storn, R. and Price, K., "Differential Evolution- A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, **1997, *11*,** 341-359.

[14] http://www.icsi.berkeley.edu/~storn/code.html#matl

[15] http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html

[16] *Handbook of Evolutionary Computation*, **1997**, IOP Publishing Ltd. and Oxford University Press.

10. APPENDIX

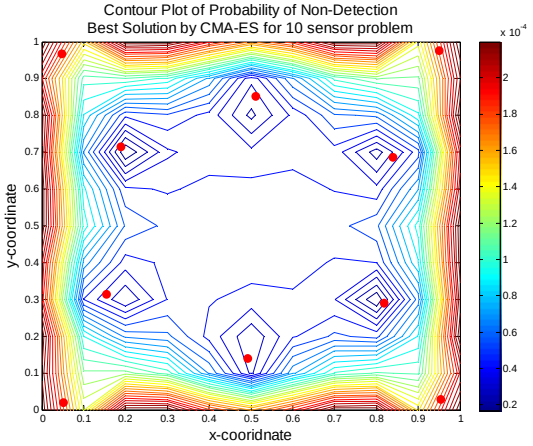

Figure 7: Contour of Best Solution by CMA-ES for 10 Sensors

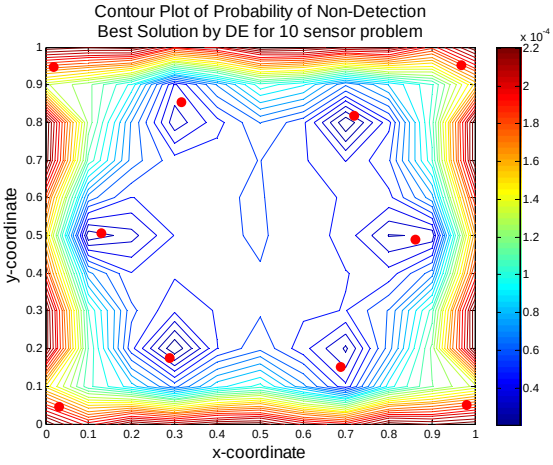

Figure 8: Contour of Best Solution by DE for 10 sensors

Table A: Parameter Settings for CMA-ES

Parameter	Setting
λ	10 (Initially, doubled as necessary)
μ	$\lfloor \lambda/2 \rfloor$
c_{cov}	$\frac{1}{\mu_{\text{cov}}}\frac{2}{(D+\sqrt{2})^2}+\left(1-\frac{1}{\mu_{\text{cov}}}\right)\min\left(1,\frac{2\mu_{eff}-1}{(D+2)^2+\mu_{eff}}\right)$
d_σ	$1+2\max\left(0,\sqrt{\frac{\mu_{eff}-1}{D+1}}-1\right)+c_\sigma$
c_σ	$\frac{\mu_{eff}+2}{D+\mu_{eff}+3}$
c_c	$\frac{4}{D+4}$
μ_{cov}	$\mu_{ceff}=1/\sum_{i=1}^{\mu} w_i^2$
w_i	see Section 5.3