Meta-Heuristics for Improved RF Emitter Localization

Sondre A. Engebråten[12], Jonas Moen[1], and Kyrre Glette[12]

[1] Norwegian Defence Research Establishment, P.O. Box 25, 2027 Kjeller, Norway.
Sondre.Engebraten@ffi.no
[2] University of Oslo, P.O. Box 1080, Blindern, 0316 Oslo, Norway

Abstract. Locating Radio Frequency (RF) emitters can be done with a number of methods, but cheap and widely available sensors make the Power Difference of Arrival (PDOA) technique a prominent choice. Predicting the location of an unknown RF emitter can be seen as a continuous optimization problem, minimizing the error w.r.t. the sensor measurements gathered. Most instances of this problem feature multimodality, making these challenging to solve. This paper presents an analysis of the performance of evolutionary computation and other meta-heuristic methods on this real-world problem. We applied the Nelder-Mead method, Genetic Algorithm, Covariance Matrix Adaptation Evolutionary Strategies, Particle Swarm Optimization and Differential Evolution. The use of meta-heuristics solved the minimization problem more efficiently and precisely, compared to brute force search, potentially allowing for a more widespread use of the PDOA method. To compare algorithms two different metrics were proposed: average distance miss and median distance miss, giving insight into the algorithms' performance. Finally, the use of an adaptive mutation step proved important.

Keywords: search heuristics, continuous optimization, multilateration

1 Introduction

Radio Frequency (RF) emitters are becoming increasingly common in everyday use. Most people carry at least one RF emitter on them at any given time, for example a cellphone or smart watch. In the case of an emergency, the ability to locate people trapped in an avalanche or in distress, would greatly relieve the search effort and possibly save lives. Locating RF emitters can, for instance, be done using a number of inexpensive quadcopter sampling the RF signal at different points in space. Figure 1 shows 10 quadcopters sampling an RF signal at multiple points in space.

There are many different methods for locating or geolocating RF signals based on sampling of signal properties [1,4,10,15,17], including: Angle of arrival, Time difference of arrival, Power difference of arrival, and Frequency Difference of arrival. Most methods for geolocation require the RF signal to be sampled at multiple distinct locations in order to achieve a prediction of the emitter

DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

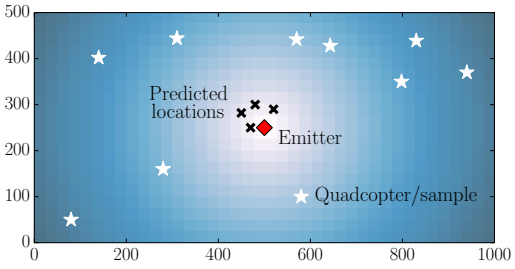

Fig. 1. Illustration of predicting the location (crosses) of an RF emitter (diamond) using 10 quadcopters/sampling locations (stars). Lighter areas have stronger signal.

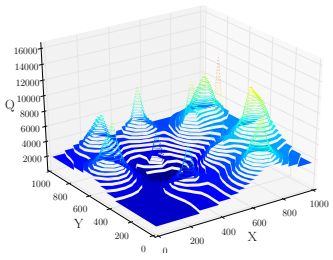

Fig. 2. An example search landscape; lower Q values are better. The example shown is for 10 spatially different measurements with noise.

location. The different methods all have their strengths and weaknesses, in practical applications it is likely that multiple methods or a combination of methods will be applied [17]. Regardless of the method applied, it is important for an implementation to be as efficient as possible.

One method of locating an RF emitter is based on Received Signal Strength (RSS), or Power Difference of Arrival (PDOA) [4,10]. This method can be implemented using cheap and readily available sensors, based on simple power measurements. An issue with this method is the high amount of computation required in order to make a prediction of the emitter location. The computational requirements includes the brute force minimization of a function over a discrete grid. This minimization can be implemented on a hardware accelerated system [4]. For many applications, where locating RF emitter would be useful, the use of hardware acceleration may be impossible due to energy constraints or inability to carry a specialized computing unit for this purpose.

The goal of this work is to reduce the computational requirements of PDOA geolocation in order to facilitate implementation, on simple and energy restricted platforms. By reducing the required computational resources it would be possible to implement this using minimal hardware, for instance on a small board computer. Evolutionary computation methods or numerical methods may assist in solving the minimization problem faster and more efficiently. The use of evolutionary computation methods would also allow for potentially infinite resolution, compared to a brute force solver.

In this work, a few of the most common heuristics for continuous optimization are compared for performance on this RF emitter localization problem. These were chosen for being common and frequently used algorithms in the literature, and were used without significant modification or customization. The algorithms chosen are examples of hill-climber methods, population based methods and higher-order search algorithms. The tested heuristics are: Random sampler, Nelder-Mead (NM) [12], Preselect NM, Genetic Algorithm (GA) [3,6], Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) [7,8], Particle Swarm Optimization (PSO) [2], and Differential Evolution (DE) [16].

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

This is the first paper to our knowledge describing the application of search heuristics in an effort to make the minimization of the error function more effective. Several papers exists on the topic of locating RF emitters [1,10,15,17]. Contrary to previous work, we apply evolutionary computation methods, instead of a brute force optimization, in order to increase the speed and precision of the emitter location predictions. A GA have previously been applied to optimize the location of the sampling points, both for the static and dynamic cases [4]. Using RSS to locate RF emitters have also been attempted in the context of a swarm system [13]. However, in this paper the focus is shifted from optimizing the behavior and positioning of the sample locations, to increasing the efficiency of the location prediction algorithm itself using meta-heuristics instead of a brute force optimization.

Section 2 describes the problem of locating RF emitters and defines a benchmark. Section 3 defines the heuristics used for this optimization problem. Section 4 describes the test cases used and the extensive parameter variation required for optimal algorithm performance. Sections 5 and 6 features results, with multiple metrics for comparison, and discussion. Finally, Section 7 concludes the paper.

2 RF Emitter Localization

In locating objects in space, there are three common exploitable methods: triangulation, trilateration and multilateration [1,15]. Triangulation estimates the location of an unknown object by measuring the angle towards the object from multiple directions. This gives an intersection where the object is estimated to be located. Trilateration uses the distance from at least three points to find an intersection between circles (in 2D) to locate an object. Multilateration combines measurements of the differences in distances, at multiple known points, in order to estimate the location of the object. These fundamental geolocation methods makes up the basis of the search for an RF emitter. There are multiple ways of locating an RF emitter [1,4,10,15], including:

1. Angle of arrival (triangulation)
2. RSS (trilateration)/PDOA (multilateration)
3. Frequency difference of arrival (triangulation)
4. Time of arrival (trilateration)/Time difference of arrival (multilateration)

Using a simple RSS method is problematic as it is fairly common that the power of the emitter is not known. Many transceivers (radio transmitters and receivers) today implement power saving schemes, where they vary emitted power. By varying the emitted signal strength, using only RSS for geolocation (with trilateration) becomes impossible. However by combining multiple RSS measurements and using PDOA it is possible to remove the unknown emitted effect at the cost of additional complexity.

2.1 Power Difference Of Arrival

PDOA compares the RSS at multiple distinct locations in order to get an estimate for the emitter location. This is based on an estimation of the loss in

signal strength at a given distance. A common model for propagation loss of an RF signal is the path loss model [10,11,14]. This model gives the RSS L at a distance r and can be expressed as follows:

$$L(r) = L_f(r_0) - 10\alpha \log_{10} \frac{r}{r_0} \quad (1) \qquad P(r) = L(r) + \mathcal{N}(0, \sigma) \quad (2)$$

$L_f(r_0)$ is signal strength "a short distance" from the emitter; this is typically determined experimentally and is a fixed value. For these experiments the distance r_0 was set to 1. $P(r)$ is a sample, with added noise, a distance r from the emitter. By attaining a number of samples $P(r)$ of the RSS at multiple different points in space it is possible to estimate the location of an RF emitter. The exact constant value of $L_f(r_0)$ is irrelevant, as it is canceled out when calculating the difference between pairs of samples.

Simulated samples are generated by adding white noise to the estimated signal strength $L(r)$, where σ is the standard deviation of the white noise. α is the path loss factor, depending on the environment this constant may vary from 2 to 6. Free space/line-of-sight gives an α of 2.0.

There are several methods of using the attained power measurement to obtain a prediction of the emitter location [10]. Some examples are: Non-Linear Least Squares (NLLS), maximum likelihood, and discrete probability density. All of these methods are fairly computationally expensive, in the order of $O(I \cdot J \cdot M^2)$. $I \cdot J$ is given by the grid resolution and M is the number of measurements. The physical sensors used for PDOA are capable of gathering several hundred samples per second. Using all the information available will result in a large M, making the optimization slow.

NLLS can be expressed as an optimization to minimize the error, given a set of measurements. By comparing the measured difference in RSS to the expected signal strength, an expression for the error can be formulated [10] as:

$$P_{kl} = P_k - P_l \quad (3)$$

$$Q(x, y) = \sum_{k<l} [P_{kl} - 5\alpha \log(\frac{(x - x_l)^2 + (y - y_l)^2}{(x - x_k)^2 + (y - y_k)^2})]^2 \quad (4)$$

The proposed location of the emitter is (x, y). k and l denotes indexes into the list of samples. Predicting, or finding, the most likely emitter location can be done by minimizing the function $Q(x, y)$ over the desired area. Analytic methods are problematic for this expression due to the non-linearity found in the expression. Figure 2 is an example of the search landscape as defined by $Q(x, y)$. The search landscape is smooth and can be highly multi-modal.

The conventional way of solving this problem would be to use regular grid, over which the function is evaluated and the smallest value located. Using a grid suffers from a number of undesirable features; such as finite resolution and high computational cost. Practical implementations may even have problems defining the grid boundaries, over which to minimize the function, as this makes

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

an assumption about the emitter location before any predictions have been made. It is also impossible to predict a location outside of the grid.

2.2 Error Metric

In order to evaluate the performance of each algorithm, a suitable benchmark metric has to be defined. All of the search and optimization algorithms will return a single best solution found through the search, the position in which $Q(x, y)$ takes on the least value seen. This solution is used to calculate an error measurement (e_{avg}), given as follows:

$$d_i = ||S_i - S_{\text{ref}}|| \tag{5}$$

$$e_i = \begin{cases} d_i, & \text{if } Q_{\text{i}} > Q_{\text{ref}} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

S_{i} is the best found solution (by the search algorithm) and S_{ref} the solution calculated by brute force using 40.000 evaluations (a fine grid of 200 by 200 cells). Both of these are two-dimensional coordinates. The Euclidean distance between the two solutions is d_i, for a single run of the optimization algorithm. An error e_i is calculated, only penalizing those solutions that have worse fitness value Q_i compare to the reference Q_{ref}. All of the errors are aggregated and an average is calculated, indicating the performance of the given algorithm on the given case. For the same set of values e_i, median and standard deviation is also calculated.

It is important to note that the true emitter location, where the emitter was placed during simulation, may not be the location of the global optimum. Due to the noise added to the samples (Equation 2), the global optimum may shift away from the true emitter location. For this reason, the global optimum has to be located using a brute force search.

2.3 Error Bound for Brute Force Search

Brute force divides the grid into a number of cells, this limits the maximal error e_i by the size of the grid cell. Similarly, it is possible to estimate the expected error assuming a uniform distribution of global optima. Two independent uniform distributions X and Y give the x and y coordinates, respectively. The exhaustive grid search (brute force) will divide the area of interest into bins of equal dimensions. Since all the bins are equal, we only need to consider the case of a single bin to find the expected miss distance. A 2D-grid of size (G_x, G_y) is divided equally into bins of size (B_x, B_y). We can then define the uniform distributions of X and Y as follows:

$$X = \text{Uniform}(-\frac{1}{2}B_x, \frac{1}{2}B_x) \tag{7}$$

$$Y = \text{Uniform}(-\frac{1}{2}B_y, \frac{1}{2}B_y) \tag{8}$$

$$D = \sqrt{X^2 + Y^2} \tag{9}$$

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

Table 1. Metrics ($B_x = B_y = 100$)

Miss dist.	Estimated	Expr.
Avg.	38.3	$0.383B_x$
Median	39.9	$0.399B_x$

$$N_{\text{eval}} = \frac{G_x^2}{B_x^2} = \frac{0.15G_x^2}{D_{\text{avg}}^2} \quad (10)$$

Monte-Carlo simulations were used to determine the expected average miss distance $E(D)$.

The expression, in Table 1, for average expected miss distance was determined numerically, using regression on a number of different values for B_x and B_y. Using this expression it is possible to devise the resolution required (using a discrete grid optimization) to achieve any desired maximal error.

In Equation 10, N_{eval} is the number of evaluations required to achieve an average prediction error of D_{avg}. For example, if the desired average miss should be less than 20m, at least 375 evaluations would be required. The area of interest (G_x, G_y) was set to $(1000, 1000)$, as used for the test cases.

This is a tight and optimistic bound for the average error. Most algorithms will not be able to attain this bound. Experiments found that even with a reasonably fine grid, the global optimum would not be sampled close enough. This resulted in choosing a local optimum instead, with a better fitness value. Missing the global optimum, and instead choosing a local optimum, gives a severe penalty to the average distance miss. The local optima are often located far away from the global optimum (See Figure 2).

3 Optimization Heuristics

Seven different common optimization heuristics were implemented in this paper: Random sampling, NM, Preselect NM, GA, CMA-ES, PSO and DE. For all of the heuristics tested, the solutions are encoded as a 2D real-valued vector. Some of these algorithms had standard implementations found in the DEAP framework [5]. The full source code can be found here [3].

For comparison, a basic random sampler was implemented. This uses a uniform distribution in x and y dimensions to generate a random sampling of solutions within the area of interest. The best solution, of the random selection, is returned by this method. This is similar to the brute force method, but does not restrict the solutions to a regular grid.

The NM algorithm [12] is a continuous numerical optimization algorithm, that does not require gradient information in the fitness space. The initial step of the NM algorithm requires a simplex to be defined. A simplex is a set of three points, for a two dimensional search landscape. Further iterations will manipulate the simplex through reflection, expansion and contraction, eventually converging to a small area of the search landscape. NM has four parameters governing the rate of contraction and simplex permutation; Alpha, Gamma, Sigma, and Phi. In the case of multiple minima, NM can get stuck or fail to converge to a single

[3] `https://github.com/ForsvaretsForskningsinstitutt/Paper-NLLS-speedup`

solution. For the problem as described, the fitness landscape may have more than one minimum. This makes the NM algorithm a poor choice by itself, but is included for comparison.

NM will typically struggle on non-smooth landscapes, or in cases where there are multiple optima. If the NM algorithm could be initialized in a way as to exclude these two problems, this algorithm could be a prime contender due to very fast convergence to a single solution. Choosing the simplex carefully it may be possible to reduce or even eliminate the adverse properties of the NM method alone. The Preselect NM method samples a (large) number of points in the search space before applying the NM method. These points are used to select the initial simplex for the NM algorithm. From the points sampled, the three best solutions (according to $Q(x, y)$ values) are chosen and given to the NM method. This allows the algorithm to start closer to the global optimum and may assist in reducing the chance of getting stuck in local optima.

In this paper, the GA [6,3] is applied as a search heuristic in the two-dimensional optimization problem defined in the previous sections. A direct real encoding is used, and the fitness function is $Q(x, y)$ as defined in the problem description. Furthermore, to apply a GA to this problem a suitable mutation, crossover and selection operator has to be chosen. Tournament selection is used as a selection operator. The mutation operator is implemented as a simple Gaussian additive mutation, which requires a sigma to be specified. The crossover operator takes two parent solutions and creates two children by selecting two new random solutions on the line segment joining the two parent solutions. In total, the GA requires five parameters to be specified: Mutation probability, Mutation sigma, Crossover probability, number of elites, and finally, a population size.

CMA-ES [7,8] is described as a second degree method building on Evolutionary Strategies [9]. As opposed to the GA, this method does not use a crossover operator, only a mutation operator and selection. The CMA-ES algorithm will attempt to adapt the mutation step size dynamically (from an initial guess), maximizing the likelihood that favorable solutions are generated by the mutations. This is an important feature, as it allows for rapid convergence once an area of solutions with good fitness has been found. The main drawback of this method is the use of a population size of 1 (in the standard implementation). Having only a single individual makes the algorithm prone to getting stuck in local optima, converging prematurely. The same properties that make the algorithm exceptional at quickly converging and find a good solution are a disadvantage in respect to robustness when faced with multi-modal fitness landscapes.CMA-ES requires two parameters: a population size and an initial permutation standard deviation.

The concept of the PSO [2] is to have a number of particles moving in the N-dimensional space and being attracted to the various optima that both the particle itself finds, and the global optimum as found by all the particles. This algorithm required minimal modification to suit the described problem, but has a number of parameters that will significantly impact the performance of the algorithm. The parameters are: population size (number of particles), attraction

weight toward local best (ϕ_1), attraction weight toward global best (ϕ_2) and a maximum particle speed.

DE [16] is a variation of an evolutionary optimization method. The main difference is the use of a differential operator as a mutation operator. In short, the mutation step used by DE is defined by the difference between two solution vectors. This allows the mutation step size to adapt and converge as the solutions in the population converge to a single optimum. DE uses three parameters: F a scaling factor for the mutation size, CR for controlling the chance of a mutation, and the population size.

4 Testing Methodology

50 cases were randomly generated, in order to make sure that algorithms excelling at a single problem instance were not preferred. Each case consists of 10 distinct randomly located simulated PDOA samples with randomly generated noise (Equation 2). In this work free space/line-of-sight was assumed, α of 2 and σ of 3.0. The optimization area (G_x, G_y) is $(1000, 1000)$. For all of the test cases the true emitter location is in the middle of the search area, but the location of the global optimum will vary depending on the noise added. In a real-world implemention of PDOA, averaging may be used to reduce the percieved standard deviation of the noise (σ). Varying the position of samples and measured RSS will drastically alter the fitness landscape. All the algorithms are benchmarked on the exact same cases. Due to the non-determinism found in several of the heuristics, each algorithm was tested 50 times on each case, using a new random seed for each trial. For all experiments algorithms were given a limited number of function evaluations, effectively limiting the number of iterations. The number of test cases and samples per case was limited by the computational resources available.

4.1 Heuristics Parameters

Most of the analyzed algorithms require one or more parameters to be defined. Common for all the population based methods is a population size, which inevitably affects the number of iterations, as each algorithm is limited in the number of function evaluations. Deciding the best set of parameters for each algorithm is non-trivial and may require expert knowledge about both the fitness landscape and the properties of the search algorithm. In this paper, a fairly extensive set of parameter combinations were tested for each algorithm. The best set of parameters, for any given algorithm, depends heavily on the metric used for comparison. The optimal set of parameters may also vary depending on the limit on function evaluations.

Based around recommended settings, the parameters seen in Table 2 were used for the different algorithms. Experiments were conducted for all the possible combinations of parameter settings. This resulted in each algorithm having around 32 different sets of parameters to test. The exception to this was the NM

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

Table 2. Parameters for all experiments and algorithms

Algorithm	Parameters
NM	Alpha 1.0, Gamma 2.0, Sigma 0.5, Phi -0.5
Preselect NM	NM evals. 10%, Alpha 1.0, Gamma 2.0, Sigma 0.5, Phi -0.5
GA	Pop. size [200 160 100 80 50 40 20 10], Mut. sigma [25.0 50.0], Mut. prob. [0.1 0.2], Crossover prob. [0.4 0.6], Elites max(2, 5%), Tournament N=2
CMA-ES	Pop. size [200 160 100 80 50 40 20 10], Sigma [100.0 125.0 150.0 200.0]
PSO	Pop. size [200 160 100 80 50 40 20 10], ϕ_1 [1.0 2.0], ϕ_2 [1.0 2.0], Speed max [20.0 50.0]
DE	Pop. size [200 160 100 80 50 40 20 10], Crs [0.25 0.50], Fr [0.5 1.0]

algorithm, which was only run with the single parameter set recommended for the algorithm. The default parameters were found to work well for this problem.

All algorithms were compared on the 50 test cases. As a baseline; a brute force optimizer, i.e. grid search, was also implemented. The brute force optimizer used an additional 450 randomly generated test instances (in addition to the 50 used for the other algorithms). Without the additional cases, brute force would only have 50 sample points (compared to 2500 for each of the other algorithms), as it is a fully deterministic algorithm.

In addition, the effect of the function evaluation limit was tested. Tests were run for 100, 200, 400, 800, and 1600 evaluations per optimization run. The performance of each algorithm will vary depending on the number of evaluations allowed to solve the problem. In total approx. 500.000 optimization runs were conducted, evaluating 300 million solutions in the search space.

5 Results

One of the main challenges for this problem, is to determine the optimal parameters for each algorithm. These parameters may change depending on evaluation budget size. In order to address this problem, extensive parameter variation tests was conducted based on values given in Table 2. Initially, the case of a fixed evaluation budget of 400 evaluations will be examined, before extending the same methods and analysis to a set of different and variable evaluation budgets.

5.1 Fixed Evaluation Budget Size

A histogram over error values (e_i) can be made for each algorithm and parameter set tested using a fixed evaluation budget size. An example of this can be seen in Figure 3 for the random sampler using 400 evaluations. For most trials, a simple random sampling will succeed in finding a good solution to the problem instance. Most of the weight of the distribution is found at less than 100m away from the reference solution S_{ref}. In some cases, a random sampling will miss the global optimum and instead choose a local optimum. This can be seen at a distance 750m in Figure 3.

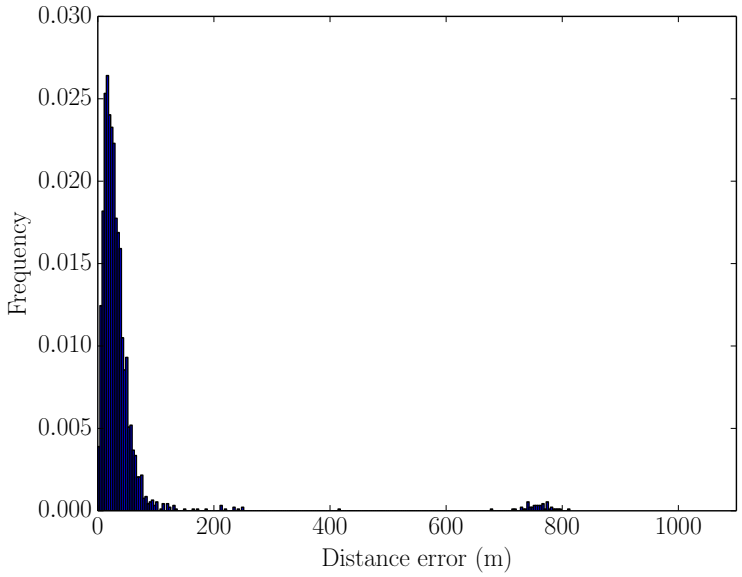

Fig. 3. Random sampling - Error distribution for 2500 tests using 400 evaluations

Left part of Figure 4 shows an example of the error distribution for the GA for a single set of parameters. The GA has a similar distribution as the random sampling, but is more likely to get stuck and converge prematurely in the given example. This can be seen by the long and heavy tail of the distribution.

DE outperforms both a random sampling and the GA and has more consistent performance. Right part of Figure 4 shows an example where the DE often is able to converge to the global optimum as defined by S_{ref}. This distribution does not have the same tail as the random sampling and the GA, indicating that it is less likely to get stuck in local optima.

However, none of the histogram plots are symmetric distributions. This leads to problems when attempting to rank the methods and generate useful statistics. In particular, the average miss of each algorithm will be significantly skewed by the outliers, favoring reliable algorithms. By using the median as a measure for comparison, this is mitigated, but it also hides some of the issues when using search heuristics on the problem. For some applications, the loss of reliability may be acceptable, but not in others. Table 3 shows performance on the metrics median, average and standard deviation for selected parameter combinations. The parameters associated can be found in Table 3. This is based on Euclidean distance, as defined in Equation 6. The selected subset of combinations was chosen based on its performance on the median metric. Normally, the average metric would give a better indication of performance, but in this case, the average metric is insufficient.

The search landscape is smooth (an example can be seen in Figure 2), but still poses a challenge for search algorithms. Multi-modality makes these problem instances hard, and often makes the search heuristic converge prematurely, or

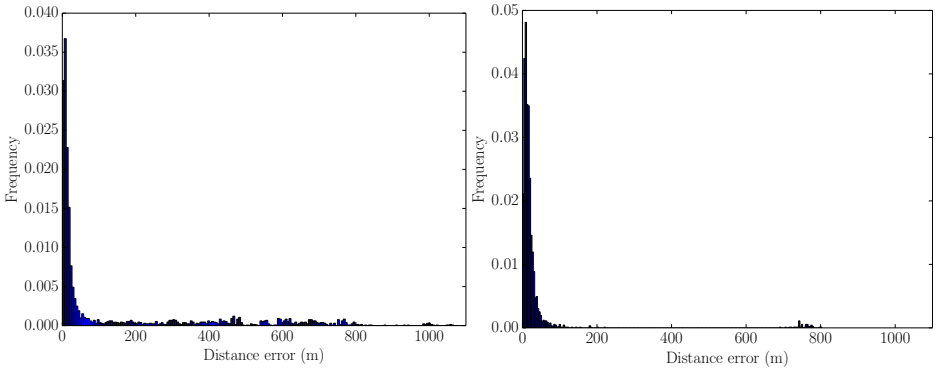

Fig. 4. Example of error distribution for 2500 tests using 400 evaluations and a single set of parameters - GA (left) and DE (right)

Table 3. Results overview - 400 evaluations

	Med	Avg	Stddev
Brute force	19.7	25.9	53.9
Random	26.1	43.9	100.8
NM	0.0	164.7	290.4
Preselect NM	0.0	17.1	103.6
GA	16.8	148.0	240.9
CMA-ES	0.0	84.0	222.7
PSO	15.7	67.8	173.5
DE	0.0	35.8	140.6

Algorithm	Params
GA	Population 10, Elites 2, Mut. prob. 0.2, Mut. Sigma 50.0, Crossover prob. 0.6
CMA-ES	Population 20, Sigma 200.0
PSO	Population 10, Phi1 1.0 , Phi2 1.0, Smax 20.0
DE	Population 10, Cr 0.5, F 0.5

get stuck in local optima. For this problem, the average metric is indicative of the likelihood for getting stuck. A single miss (on the global optimum) gives a severe penalty to the average. If the average was used to select the best parameter for each algorithm, a single generation/iteration with the maximum possible number of evaluations would be preferred for many algorithms. With such parameter settings, any heuristic becomes a random search.

An alternative to focusing on a single metric, such as median or average, could be to use a combination of metrics, as commonly done in multi-objective optimization. The metrics are defined as follows:

1. Probability of getting stuck in a local minimum
2. Average Euclidean distance error, given that the global minimum was found

The first of these two metrics acts a filter, effectively removing the outliers seen in the histogram plots. In order to quantify the chance of getting stuck, a solution has to be classified as either a part of a local optimum or the global optimum. While classifying the solutions like this is non-trivial, a simple approach would be to define some radius around the global minimum, and use this as a selection threshold. Based on the clustering of solutions seen in the experiments (Figure 3 and 4), a radius of 100m was selected. This is a relaxed limit and

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

includes most, if not all, solutions that were in the basin of attraction of the global optimum.

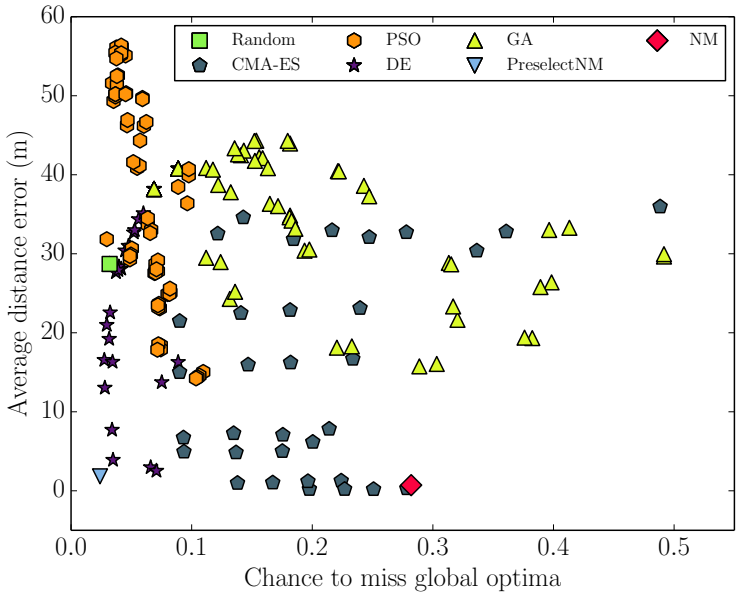

Fig. 5. Algorithm comparison using two alternative metrics. Each dot represent a parameter configuration. Clustered dots from the same algorithm typically share parameters significantly affecting performance.

Figure 5 shows the different algorithms-parameter combinations and how they perform on the two objectives. Preselect NM is able to approach the origin, i.e reference solution. CMA-ES shows excellent ability to find equal or better solutions as the reference solution, but lacks in reliability; as seen by a fairly high chance of missing the global optimum completely. DE is the opposite of CMA-ES and is very reliable, but lacks somewhat in its ability to converge to solutions equal or better than the reference.

5.2 Variable Evaluation Budget Size

Another interesting view of this problem would be to examine the performance of each algorithm across evaluation budget sizes. All the algorithms were tested with 100, 200, 400, 800 and 1600 evaluations, and the number of evaluations are likely to affect the performance of the algorithm. Figure 6 shows a comparison across evaluations using the average metric. In this figure, the parameters for each algorithm were selected based on average metric.

Only DE and Preselect NM manage to compete with brute force search on the average metric. The rest of the heuristics were found *on* or *above* the line for random sampler and are excluded from the plot for readability. Preselect NM

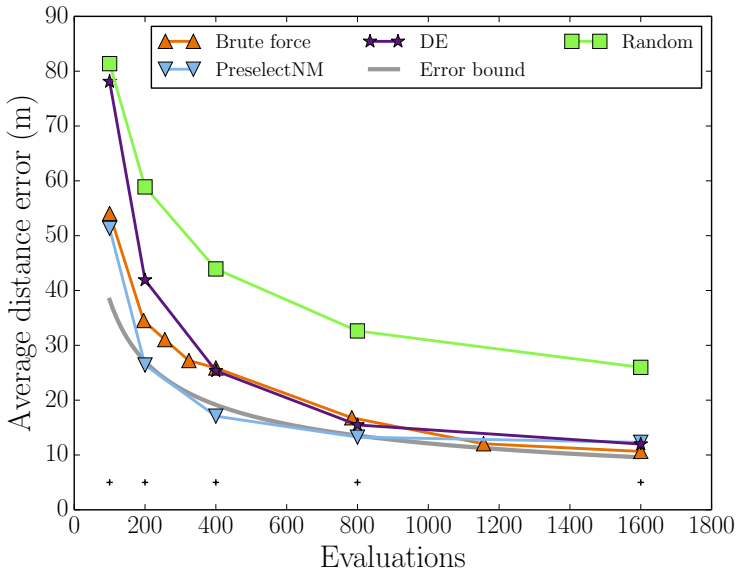

Fig. 6. Comparison of algorithms across number of evaluations using average metric. Plus signs indicate statistical significant difference between Preselect NM and Brute force algorithms for the given number of evaluations.

performs well across all numbers of evaluations; particularly in the area of 200 to 800 evaluations, where it is comparable to using twice as many evaluations on a brute force search. In other words, applying the Preselect NM algorithm resulted in a speed-up of at least 2x on this real-world problem.

In Figure 6 a plus (+) signs indicate the positive result of a Wilcoxon Rank and Sum test, testing the Brute force algorithm against Preselect NM using comparable number of evaluations. This test was applied for each number of evaluations as indicated in the figure. The distribution of errors, between Preselect NM and the Brute force algorithm, was found to be significantly different at the level of 0.01 for all tests applied. The results are the same for the median plot because the underlying dataset is the same.

The challenge of this real-world problem is not to find a good solution, but to find the best solution. For several of the heuristics this is problematic, as they readily will converge to local optima. Yet, the problem may greatly benefit from a heuristic in converging to the global optimum from nearby areas in the solution space.

Using a median metric shows another view of the algorithms performance (Figure 7). This plot highlights the benefit of using a search heuristics in providing solutions that often are better than the reference solution found by a brute force optimization using 40.000 evaluations. Parameters were here chosen based on median. DE and Preselect NM are repeatedly able to find solutions that outperform a brute force optimization with an equal number of evaluations.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

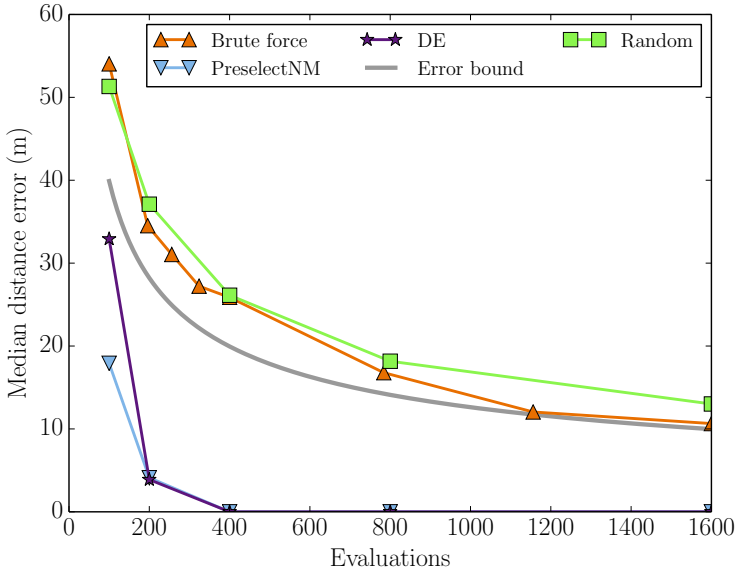

Fig. 7. Comparison of algorithms across number of evaluations using median metric.

Finally, considering the the effect of increasing the number of evaluations for Figure 5 is a gradual tendency for all points in the plot to converge on the origin. As more evaluations are allowed, most algorithms and parameter combinations perform better (i.e. provide better solutions). Decreasing the limit has the opposite effect.

6 Discussion

For this particular real-world problem, finding a suitable metric for comparing algorithms and parameter combinations proved challenging. With the average miss as the metric, many of the search heuristics had problems caused by their unreliability to find the global optimum. Using the median metric instead allows for a partial solution to this problem, but will camouflage how unreliable each algorithm is to a certain degree.

Many of the heuristics are able to often find solutions that outperform those found by a brute force search using several orders of magnitude more resources. In particular, DE, Preselect NM and CMA-ES proved to excel at this. What differentiates these algorithms from the remaining (PSO and GA) is the use of adaptive step lengths when converging on an optimum. In order to maximize the solution performance, this proved an important trait. It becomes clear that there is a trade-off between the reliability of an algorithm and the ability to narrow in on an optimum. With a limited evaluation budget a heuristics cannot do both, and depending on parameters, may focus on one or the other.

One of the main premises for using search heuristics is that there is no guarantee that the optimal solution will be found. For some problems this may be acceptable, and not for others. In this case, it depends on the application scenario of the method. In presenting a prediction of the location of an RF emitter

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1007/978-3-319-55792-2_14

to a user, there is an expectation of reliability and predictability. Automated systems may to a greater degree be able to accept the chance that, in certain cases, a prediction might miss, as long as the miss is not too great or too frequent. In future work it would be interesting to investigate the use of a search heuristic, as described in this paper, in a context of swarm system. Commonly, swarm systems rely on simple and cheap units with limited capabilities. In such a context, the acceptance of suboptimal performance (a chance to miss) may be unavoidable and must be dealt with on a higher algorithmic level.

In the case of a miss being unacceptable and absolute reliability required, significant performance increase is still possible using a search heuristic. As shown for Preselect NM, a no-cost speed-up can be achieved, effectively giving double the performance on limited resources. As described previously, the issue of a suitable metric camouflages some of the characteristics of the search heuristic. For this particular case, the use of a hill-climber allows a system implementing this to be at least as reliable as a system using twice the amount of resources on a brute force search. In addition, the system gains an infinite resolution. What previously was limited to the resolution of the brute force grid is only limited by the resolution of the number representation, and how quickly the hill-climber can converge. This is an important result of this work.

7 Conclusions

This paper shows the viability of using search heuristics on the problem of geolocating RF emitters. By using a search heuristic, multiple favorable attributes can be achieved, such as: infinite resolution, reduced and flexible computational cost, and greater robustness against deceptive objective functions when restricted in computational resources. Comparing a number of common search algorithms, such as GA, PSO and CMA-ES, it is clear that these strategies may not always be the best option given a limited computational budget. The challenge for these algorithms is to converge quickly enough while at the same time avoiding local optima. If the search space is reasonably small, applying the NM algorithm with a preselect may be an option resulting in high performance even with little resources.

One of the biggest issues in this particular problem was the multi-modality of the fitness landscape. Multiple local optima made this a deceptive problem and required algorithms that were robust and had an exploratory behavior.

This work may not only allow for a practical real-world implementation of a system locating RF emitters, but also a wider range of concepts to be explored. The ability to locate an RF emitter can also be used as part of a higher level simulation, investigating into the behaviors or how multiple agents should interact. One intriguing idea is a swarm of flying platforms able to autonomously locate RF emitters. This is a topic for future research.

8 Acknowledgments

We would like to thank Ingebjørg Kåsen and Eilif Solberg for their assistance with the statistical issues in this paper and Jørgen Nordmoen for enlightening discussions and excellent feedback.

References

1. F. Berle. Mixed triangulation/trilateration technique for emitter location. *Communications, Radar and Signal Processing, IEE Proceedings F*, 133(7):638–641, 1986.
2. R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.
3. A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.
4. S. A. Engebråten. RF Emitter geolocation using PDOA algorithms and UAVs. Master's thesis, Norwegian University of Science and Technology, 2015.
5. F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.
6. D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.
7. N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.
8. N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.
9. A. Huning, I. Rechenberg, and M. Eigen. Evolutionsstrategie. optimierung technischer systeme nach prinzipien der biologischen evolution, 1976.
10. B. Jackson, S. Wang, and R. Inkol. Emitter geolocation estimation using power difference of arrival. *Defence R&D Canada Technical Report DRDC Ottawa TR*, 40, 2011.
11. N. Levanon. Radar principles. *New York, Wiley-Interscience, 1988, 320 p.*, 1, 1988.
12. J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.
13. J. Nordmoen. Detecting a hidden radio frequency transmitter in noise based on amplitude using swarm intelligence. Master's thesis, Norwegian University of Science and Technology, 6 2014.
14. S. Saunders and A. Aragón-Zavala. *Antennas and propagation for wireless communication systems*. John Wiley & Sons, 2007.
15. H. Staras and S. N. Honickman. The accuracy of vehicle location by trilateration in a dense urban environment. *Vehicular Technology, IEEE Transactions on*, 21(1):38–43, 1972.
16. R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
17. Z. Wang, E. Blasch, G. Chen, D. Shen, X. Lin, and K. Pham. A low-cost, near-real-time two-UAS-based UWB emitter monitoring system. *Aerospace and Electronic Systems Magazine, IEEE*, 30(11):4–11, 2015.