Accelerating Automated Antenna Optimization with Parallel Recombinative Simulated Annealing and Search Space Discretization

Kevin Bi and Griffin Whybra[1]

***Abstract*— Antenna design is an important, yet difficult part of any embedded system that requires wireless communication. Efforts have been made to automate the design process with Genetic Algorithms (GAs) and Simulated Annealing (SA). However, GAs and SA can converge slowly or not at all. In this paper, we use Parallel Recombinative Simulated Annealing (PRSA) along with Search Space Discretization (SSD) to create an optimization algorithm which converges in less iterations than both GAs and SA. We first test GAs, SA, PRSA, and PRSA-SSD with a simpler problem to show the difference in convergence speeds. We show that SSD does not exclude any significantly different antennas from the optimization process, and we provide examples of using GAs, SA, PRSA, and PRSA-SSD to optimize trace antennas.**

I. Introduction

Antennas are designed by starting with a reference design and iterating on the design until reasonable values for impedance are reached or taking a reference design and matching the impedance of the design to the impedance of the source. This process is difficult, and can take multiple man-hours for a single antenna, tying up important resources.

Automated methods for antenna design exist, especially through genetic algorithms (GAs) [1] and simulated annealing (SA) [2]. GAs and SA represent a class of optimization algorithms intended for optimizing NP-hard problems, where finding the best solution requires complete enumeration. GAs and SA arrive at arbitrarily good solutions as determined by the user, and if set up properly, can eventually reach the global optimum. However, setting up GAs and SA properly is hard and thus can converge slowly or get stuck in local optima.

Speeding up automated antenna optimization can also be achieved by decreasing the time it takes to find a viable solution. Most of the time it takes to optimize an antenna is spent in simulation; reducing the number of simulations speeds up optimization. We show that small, local variations in antenna design have little-to-no effect on the impedance and gain pattern of the antenna. Requiring antennas to be significantly different from each other can decrease the number of antennas that are effectively simulated multiple times.

This paper presents a method of automated antenna optimization that can provide convergence in fewer iterations than both SA and GAs using parallel recombinative simulated annealing (PRSA) combined with state space discretization. Section II will cover GAs and SA, the optimization algorithms used in previous papers. Section III will cover parallel recombinative simulated annealing, search space discretization, and both the problem formulation for the simple test problem as well as the antenna optimization problem. Section IV will cover our results, Section V will cover our conclusions, and Section VI will cover future work to be done.

II. Optimization Algorithms

A. Genetic Algorithms

Genetic algorithms are optimization algorithms based on the idea that better solutions can arise from a combination of other good solutions [3]. Genetic algorithms iterate through generations, where each generation contains a population of individuals. Each generation is ideally better than the previous one and this improvement is caused by each generation performing selection, crossover, and then mutation, which produces children that could be better than individuals that exist in the population.

Selection is the process through which parents are chosen. In a problem where the fitness of the solutions moves monotonically towards the global optimum, selection would just pick the two best parents. However, many optimization problems have a plethora of local optima which should be avoided, so the selection process should be capable of selecting suboptimal parents in order to escape local minima. Thus, the selection process randomly selects parents, but weighting the random selection so that better individuals have a higher chance to be selected.

Crossover is the process where children are created from the parents. Children are created by randomly selecting a crossover point, and the first child takes the values of the first parent before the crossover point and the values of the second parent after the crossover point, as shown in Figure 1. This creates new solutions which are combinations of the old solutions that in theory could be better.

Mutation is then applied to the children generated by crossover. In mutation, each value in the children has a chance to be replaced by a random value. Currently we aim for one mutated value in each child, but it is a random process and more or less could occur.

The children then replace the two individuals in the population with the worst fitnesses, producing a new generation. The selection/crossover/mutation process is then repeated until a solution with a desired fitness is found.

B. Simulated Annealing

Simulated annealing [4] is another optimization algorithm where the ability to ignore local optima is provided by Boltzmann trials, which govern the probability that a better solution is accepted over a worse solution.

[1] Equal contribution, order chosen by flipping a 1973 US Quarter.

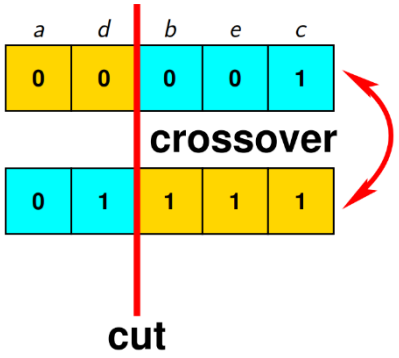

Figure 1: Crossover applied to two individuals. Image taken from [5].

Unlike GAs, SA doesn't have a population of solutions. SA only optimizes a single solution, and an iteration consists of finding a neighboring solution, computing the fitness of that solution, and then selecting between the current solution and the neighboring solution and decreasing the temperature.

Creating a neighboring solution is similar to the process of mutation in a GA, where each value in the individual has a chance to be mutated, creating an individual close to the original one. Selection of the next solution is done by random selection weighted by the result of the Boltzmann trials.

Equation 1 shows the formula for the Boltzmann trials. As Figure 2 shows, if the temperature U is very high, then $e^{(J-K)/U}$ will be approximately 1, and so P_a will be close to 0.5, so a high temperature indicates that the fitness of the solution won't matter much and the selection of the next individual is essentially random. If the temperature is low, then $e^{(J-K)/U}$ will be very large, and P_a will be either 1 or 0 so the best solution will always be selected.

Thus, slowly decreasing the temperature slowly increases the chance that a better solution will be selected, allowing the solution to leave local optima early yet stay near the global optima later on.

$$P_a = \frac{1}{1 + e^{(J-K)/U}}$$

Equation 1: Probability that solution a (P_a) wins the trial, given solutions with costs J and K [6]

III. Proposed Method and Problem Formulations

We propose a combination of PRSA and SSD called PRSA-SSD as a method of decreasing the amount of iterations to convergence. PRSA is derived from a combination of GAs and SA and fixes some of the problems with both, so we expect convergence in a smaller number of iterations. SSD decreases the search space by removing similar solutions, so the combination should provide even faster convergence.

We intend to attempt to optimize antennas, which we can do with an electromagnetic simulator capable of simulating arbitrary antennas and determining their gain pattern and impedance.

However, simulating antennas is an extremely time-consuming process, and so we use another smaller optimization problem to prove our methods work before moving on to optimizing antennas.

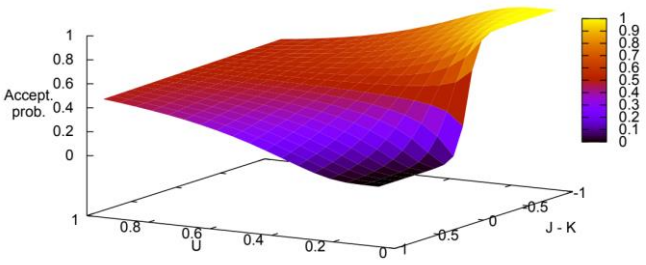

Figure 2: Acceptance probability P_a plotted against temperature U and cost difference J-K. Image taken from [5].

A. Parallel Recombinative Simulated Annealing

Parallel recombinative simulated annealing [6] is a combination of GAs and SA. PRSA can be thought of as SA except the algorithm utilizes a population of individuals instead of a single individual. Alternatively, PRSA can be thought of as GAs, except some more children are created and instead of children being automatically swapped into the population, the children are swapped in with a probability governed by Boltzmann trials.

The PRSA version we implemented was based on [6]. This version iterates until a target temperature is reached, like traditional simulated annealing, and in every iteration, children are created through crossover, mutated, and placed back into the population replacing their parents with a probability determined through a Boltzmann trial. The number of children created is equal to the population size, which is from the original implementation in [6].

B. Search Space Discretization

SSD is based on the idea that checking individuals similar to the individual being tested does not produce useful information. Thus, SSD works by enforcing a minimum difference between individuals. This idea can be expanded so that instead of finding sufficiently different individuals during the optimization algorithm, we discretize the search space into sufficiently different individuals in the first place. This is done by first selecting a minimum difference, computing the number of values inside the valid search area, and then randomly selecting one of the values. For example, if our valid search space is between 0 and 2, and we choose a minimum difference of 0.5, we would randomly pick between 0, 0.5, 1, 1.5, and 2, instead of selecting a random float between 0 and 2, transforming an infinite number of choices to 5.

This idea can be extended to antennas, as small variations in the shape of an antenna (on the order of 0.5% of a wavelength) have little to no effect on the gain pattern and impedance of an antenna. Thus, when antennas are created or mutated, the antenna needs to be different enough to create an antenna with significantly different properties compared to previous antennas.

C. Simple Problem Formulation

Our simple problem is attempting to recreate a black and white image by minimizing the sum-of-squares difference in Equation 2 between target image I_{target} in Figure 4 and generated image $I_{generated}$.

$$Diff = \sum_i \sum_j (I_{target}(i,j) - I_{generated}(i,j))^2$$

Equation 2: Sum-of-squares difference computed between target image I_{target} and generated image $I_{generated}$.

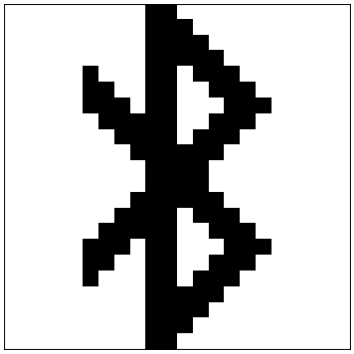

Figure 4: 22x22 Black and White Target Image

Without SSD, the pixel values created by the optimization algorithms are allowed to be any real number between the maximum value (255) and the minimum value (0), inclusive. We show two types of SSD here. One discretizes the search space to the integers between 255 and 0, inclusive, and the other discretizes the search space to exactly the two numbers 255 and 0, as we know the image is black and white and therefore composed only of pixels with the values 255 or 0.

D. Antenna Problem Formulation

The antenna is composed of a series of points. The points are connected in order and form a series of line segments, or physically, a microstrip trace antenna. The thickness of the trace is fixed at the thickness of 2 oz. copper. The width of the lines and the horizontal positions of the lines are fixed as well, and thus the antenna is encoded as an array of floating-point numbers representing vertical positions.

This representation of the antenna preserves spatial locality as well. Spatial locality is the property of the solution representation where the parts of the solution near each other in the representation are also near each other in real life and should interact more. In the case of the antenna, antenna segments close to each other will affect each other. Locality is important for GAs and PRSAs as they both use crossover, where parts from two solutions are combined to create new solutions.

Python is used to convert the array into a series of instructions for GiD, a CAD preprocessing engine, which creates a CAD model of the antenna, converts the CAD model into a mesh, and outputs the mesh into a text file suitable for use with the simulator.

The electromagnetic simulator we use is VirAntenn, which takes the text file with the mesh and outputs the gain pattern and impedance at specified frequencies. In our case, we use 2.4GHz, since it is a common frequency for wireless communication.

An antenna is considered to be better than a different antenna if it has a better fitness. Here, we define better as the

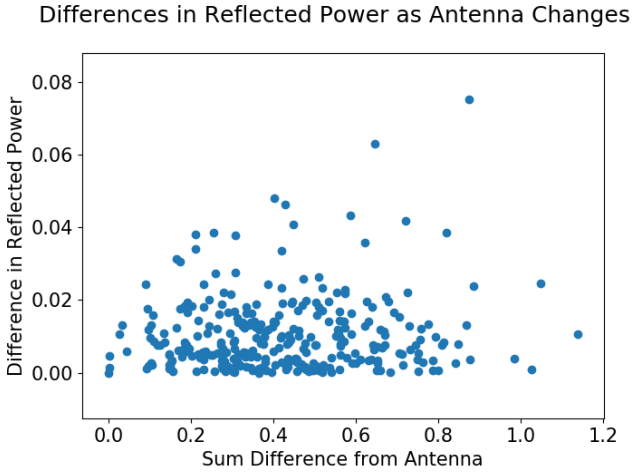

Figure 3: Difference in reflected power between antennas and similar antennas.

antenna being able to radiate more energy. Impedance mismatch between the source (usually a wireless-enabled SoC or similar) and the antenna will cause energy to be reflected instead of radiated, so a better antenna will have an impedance closer to the source impedance, which we define as 50 Ohms.

We use Equation 3 to compute the amount of energy reflected, which we then minimize.

$$P_{reflected} = \frac{\sqrt{(50 - Z_{real})^2 + Z_{imag}^2}}{\sqrt{(50 + Z_{real})^2 + Z_{imag}^2}}$$

Equation 3: Power reflected by the impedance mismatch between a 50 Ohm source and an antenna with impedance $Z = Z_{real} + Z_{imag} * j$

IV. Experimental Results

A. Gain and Impedance Changes Caused by Local Variation

We want to ensure applying SSD to the antenna problem doesn't remove any significantly different antennas. We simulated 10 randomly generated antennas, and for each of the 10 randomly generated antennas, we generated 30 similar antennas. These similar antennas were generated so that each point in the similar antenna is no more than 0.5 away from the original antenna. Since the unit we use is millimeters, the maximum difference per segment is 0.5mm, and at 2.4GHz, 0.5mm is approximately 0.4% of the wavelength.

Figure 3 shows the difference in fitness versus the total point-wise difference in mm. The fitness is computed to be the amount of power reflected by the antenna as in Equation 3. We see that for the majority of antennas, the difference in power reflected is less than 2% which is acceptable, especially for earlier iterations of the optimization functions. For further antenna simulation with SSD, we drop the minimum difference to 0.25mm to stay in the 2% difference range.

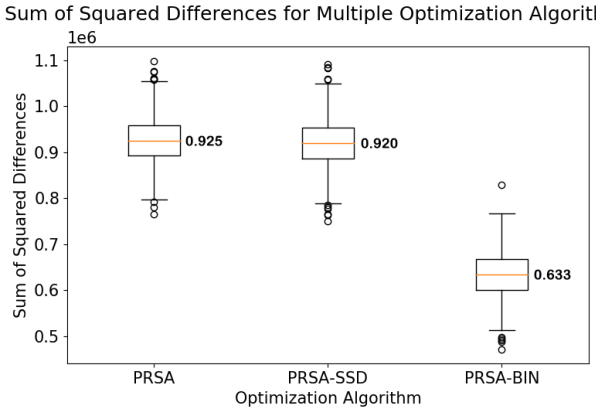

Figure 7: Sum of squared difference for PRSA and PRSA with SSD and BIN after 1700 iterations. Lower is better.

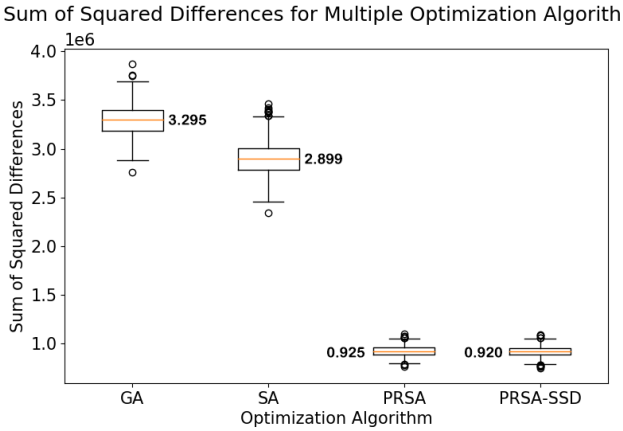

Figure 8: Sum of squared difference after 1700 iterations. Lower is better.

B. Comparison of Convergence Speeds for the Target Image

We started by testing the convergence speeds of our various optimization algorithms on our toy problem of optimizing random images to look like our target image. Figure 8 shows the fitness after a fixed 1700 iterations, and PRSA and PRSA-SSD perform dramatically better than GAs and SA does.

However, this graph is somewhat misleading, as in each iteration, SA does 1 fitness computation, GA does 2, and PRSA and PRSA-SSD do 8 each. Since the computation of the fitness is what requires the large majority of computing power, Figure 5 shows the number of fitness computations required to minimize the difference to a certain point. The data is generated by running each algorithm to for the same number of fitness computations 1000 times.

We expected PRSA-SSD to require the least computation, followed by PRSA, then GA, then SA. However, this was not the case. We think this is because this test problem is actually too simple. PRSA does 8 fitness computations per iteration, but the simplicity of the problem means that finding a better solution is simple, and many of the 8 new solutions are valid, and thus the computing power is somewhat wasted. In SA, if it is simple to find a better solution, then SA operating on an individual instead of a population of 8 means that a majority of the time, doing a single fitness computation will provide the same benefit as doing 8, so the SA is able to converge to a good solution in far less computations. Thus, the computations

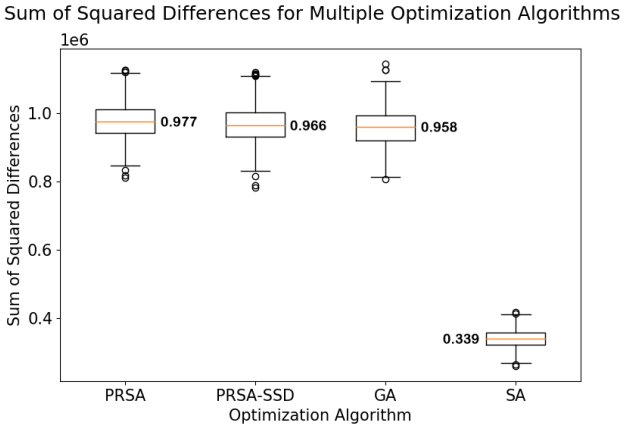

Figure 5: Sum of squared difference after 13000 fitness computations. Lower is better.

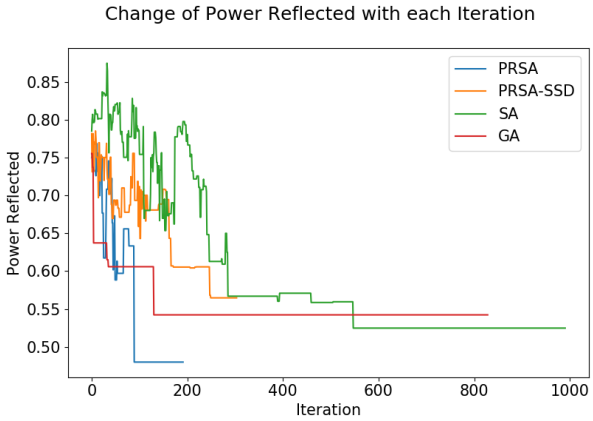

Figure 6: Fitness (power reflected) for each of the optimization algorithms as a function of iterations.

to convergence likely scales almost linearly with the number of computations per iteration instead of with the number of iterations.

We expect for more difficult problems that the computations to convergence scales more with the number of iterations.

C. Effectiveness of State Space Discretization

Figure 7 is a comparison for the small image problem between standard PRSA, where the search space per pixel is floating-point numbers between 0 and 255, PRSA-SSD, where the search space per pixel is integers between 0 and 255, and PRSA-BIN, where the search space per pixel are the integers 0 and 255. This data was generated by running each optimization algorithm 1000 times, stopping at 1700 iterations per run, after which we collected the best fitness.

This comparison shows that for standard SSD, PRSA-SSD does converge to a slightly better solution, and PRSA-BIN converges to a significantly better solution.

Thus, if it is possible to discretize the search space so that an optimization algorithm doesn't repeat effectively similar values, it is likely worth doing so, as depending on the degree of discretization, large speedups can be had. However, it is important to ensure that the discretization doesn't significantly impact the optimization algorithm's ability to find a good solution, as discretizing the search space can also remove the optimal solution from the pool of possible solutions.

D. Comparison of Convergence Speeds for Antenna Optimization

Since we had limited time, we were only able to run the optimization algorithms once each on the antennas. The data is shown in Figure 6. We are aware this data is not enough to prove that our methodology works. However, the data seems encouraging in that the SA converges slower than the rest, all of which aren't too far off of each other. Unfortunately, the large amount of randomness inherent in the process makes comparing individual runs useless, but at least these metrics show that PRSA and PRSA-SSD are capable of optimizing antennas.

V. Conclusions

PRSA along with SSD may allow the acceleration of antenna optimization. In this paper, we show that SSD does accelerate PRSA for our small image optimization task. We also show that SSD is a valid methodology to use with antennas, as the gain patterns and impedances of antennas does not vary much with small variations in antenna structure.

Furthermore, we show that PRSA and PRSA-SSD are capable of optimizing antennas, but the sample size was too small to conclusively determine which optimization method converged faster.

We conclude that it is possible that the combination of PRSA and SSD is potentially capable of speeding up antenna optimization.

VI. Future Work

As mentioned previously, running an optimization algorithm to convergence once doesn't provide any useful results, given the large amount of randomness inherent in the process, and more optimization runs should be done in the future.

Another method of potentially speeding up antenna optimization is replacing the simulation step with a neural network that roughly approximates the simulation. High accuracy doesn't matter too much if a large solution space is being explored; the amount that a solution is better likely matters less than which solution is better. As the solution starts converging, the optimization can move back to utilizing simulation as the accuracy needs grow.

One unexplored idea in this space is alternative cooling schedules. For PRSA and SA, we used fixed cooling schedules, where the temperature lowers by a set percentage after every iteration. We can use adaptive cooling schedules that are based on the current fitness, or applying a technique called reannealing, where after the error gets to a certain point the temperature is raised a little to allow for more exploration in the neighborhood of a good solution.

Finally, in SSD, minimum difference is a variable that could be tweaked. An adaptive minimum difference also based on the fitness would allow the optimization algorithm to explore solutions closer and closer to the current solution as the solution gets better. Essentially, as the solution moves closer to the global optimum, the solution is allowed to make smaller steps so that it doesn't step over the global optimum.

References

[1] G. Hornby, A. Globus, D. S. Linden, and J. D. Lohn, "Automated Antenna Design with Evolutionary Algorithms," *Space 2006*, San Jose, CA, USA, September 19-21, 2006.

[2] J. E. Richie, C. Ababei, "Synthesis of Cylindrical Antenna Arrays Using Simulated Annealing," *Journal of Computational Design and Engineering*, vol. 4, no. 4, pp. 249-255, October 2017.

[3] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," *Journal of the ACM*, vol. 9, iss. 3, pp. 297-314, July 1962.

[4] S. Kirkpatrick, C. D. Gelatt, Mario P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[5] R. Dick, EECS 598-013 Lecture 3.

[6] S. W. Mahfoud, D.E. Goldberg, "Parallel Recombinative Simulated Annealing: A Genetic Algorithm," Parallel Computing, 21, pp. 1-28, 1995.