

## **SAGA: The Aerial Spray Advisor Genetic Algorithm**

W. D. Potter, W. Bi,

D. Twardus, H. Thistle,  
M. J. Twery, J. Ghent,

M. Teske

A. I. Center  
University of Georgia,  
Athens, Georgia USA  
potter@cs.uga.edu

USDA Forest Service

Continuum Dynamics  
Princeton, NJ

### ***Abstract***

*Improving aerial spray application results is a major concern for the USDA Forest Service and Environmental Protection Agency. The AGDISP Aerial Spray Simulation Model is used to predict the deposition of spray material released from an aircraft. The prediction is based on a well-defined set of input parameter values (e.g., release height, and droplet size) as well as constant data (e.g., aircraft and nozzle type). But, for a given deposition, what are the optimal parameter values? This problem is considered to be a parametric design problem or more generally a configuration problem. Attempting to optimize a configuration based on some set of constraints is known to be extremely difficult (NP-Hard). We use the popular Genetic Algorithm to heuristically search for an optimal or near-optimal set of input parameters needed to achieve a certain aerial spray deposition. Having this knowledge can benefit forest managers substantially, especially regarding such issues as cost, environmental safety, and forest treatment accuracy.*

Keywords: Genetic Algorithms, Aerial Spray Deposition, Parameter Optimization.

## **Introduction**

Determining the parameter value settings to use as input to the AGDISP Aerial Spray Simulation Model [Bila89] in order to produce a desired spray material deposition is considered an instance of a parametric design problem [Davi91]. Parametric design is a specialization or subtype of the more generic design problem. Typically, when working on a design problem, the solution representation is a set of instructions or components for achieving the design goals. This representation can also be called a configuration, especially if the elements comprising the configuration are predefined. For the parametric design problem we are dealing with, these elements correspond to the AGDISP simulation input parameters. Each parameter has its own domain and range of values. If we arrange the parameters in a one-dimensional array or vector, and select some value for each parameter from that parameter's range then we would have an input parameter configuration. Using this configuration (set of values) as input to the AGDISP simulation model would yield a prediction of the spray deposition.

For this type of problem, the total number of possible configurations can be extremely large. For example, we can calculate this value simply by multiplying together the number of possible values for all of the parameters. That is, if there were twelve parameters, and each parameter had a range of twelve values then the total number of configurations would be  $12^{12}$ . This is indeed a very large number. Now, if we wanted to find the best configuration to achieve a desired spray deposition, then we could enumerate all the possible configurations and run the simulation on each one to see which

configuration gave the best deposition. Clearly, this sort of computational task is outside the scope of current computing technology. The configuration problem suffers from what is called combinatorial explosion, that is, as the number of elements increases (e.g., add more parameters), the number of possible configurations also increases but at an exponential rate. See the discussion by Mittal and Frayman in [Mitt89] for more on generic configuration tasks and their complexity.

One method we can use to reduce the computational burden of finding a particular configuration is a heuristic search technique. Heuristic search techniques have been shown to be effective techniques for finding acceptable solutions to problems with very large solution spaces (total number of possible solutions or configurations). The major advantage of a heuristic approach is its speed. The major disadvantage is that there is no guarantee that the heuristic search will find the best solution or configuration. However, typically the solution found is of high enough quality that the trade-off is well worth it. In other words, we may not find the best solution but the solution we do find is typically good enough for our purpose and we found it very quickly. The heuristic search technique we use is the Genetic Algorithm. In the following sections we discuss in more detail the configuration problem, the genetic algorithm, the aerial spray deposition problem, and our approach to the problem along with some recent results.

## **Previous Work**

In the development of a good heuristic approach, two methods or knowledge-based system approaches are available to us. These are the rule-based (experiential) approach using typical if-then rules, and the functional (deep or associative) approach based on knowledge about the structure and behavior of a system and its components (see [Chan83, Chan91] for more on the two general approaches). The functional approach follows the "reasoning from first principles" paradigm. This is

quite different from the rule-based approach driven by "rules of thumb". Regardless of the approach, the major point of emphasis is the heuristic synthesis of a satisfactory solution or, in our case, a satisfactory configuration. Our Spray Advisor Genetic Algorithm (SAGA) approach, however, could be considered a combination of the rule-based and functional paradigms (although we do not have a typical collection of if-then rules, expert knowledge is incorporated into SAGA in the form of the sophisticated AGDISP simulation model). For a configuration application, synthesis means incorporating a given set of parameter settings and a set of constraints associated with the parameters into a configuration that satisfies the deposition goals and constraints. Synthesis can be thought of as the design of a solution.

Probably the most famous expert system to be developed for design applications is R1 (XCON) which is used to configure computer systems from customer specifications [Bach84, McDe81]. An early example of an engineering design system for configuring networks using heuristics is DESIGNET, developed by Bolt, Beranek, and Newman in the early 1980's [Mant86]. DESIGNET is a rule based design aid that focuses on an iterative user interface approach to configuration based on the process a decision maker goes through during the design process.

Our own experience with configuration deals with designing battlefield communication networks to support specific missions. Our system, called IDA-NET, configures a "shopping list" of communication equipment indicating type of equipment and number of components [Pott92]. The "shopping list" represents the required amount of equipment to support a particular mission. The goal is to minimize the number and types of components yet still satisfy a set of constraints associated with the mission, the equipment connectivity, and the available components in inventory.

## **Aerial Spray Models**

For many years, computer simulation models for predicting what happens to spray material released from aircraft have been a major research interest of the USDA Forest Service [Teske98b]. The Forest Service Cramer-Barry-Grim (FSCBG) aerial spray model [Teske89, Teske93a] and the Agricultural Dispersal (AGDISP) model [Bila89, Teske98a] are examples of this research. AGDISP simulates the effects of aircraft movement and wake on material released from the aircraft. The model predicts the behavior of spray material droplet movement when sprayed from an airplane or helicopter. FSCBG predicts the dispersion of the spray material and the deposition of the material (that is, how much material settles on the ground and where). Both models analyze the movement of the spray material above the forest canopy, the movement among the trees, and the amount of material that actually reaches the ground. Getting the spray material to reach the proper location depends on many factors. These factors include: (1) the altitude of the aircraft when the material is released, (2) the speed of the aircraft, (3) whether the aircraft is an airplane or a helicopter, (4) the type of boom and nozzle system used to discharge the spray material, (5) the swath width of each pass of the aircraft, (6) the type and density of the forest, (7) wind speed and direction, (8) relative humidity, and (9) spray material characteristics. Determining the optimal set of factors in order to provide accurate (getting the spray material exactly where it should be), and inexpensive (using the exact amount of material; not too much and not too little) spraying is the goal of our research. We are currently investigating the use of a genetic algorithm to determine the parameters.

The output of the various computer simulation models typically includes three important values: the deposition composed of volume median diameter (VMD) and drift fraction, and the coefficient of variance (COV). VMD is a measure of spray material droplet size. It is important to know the

expected droplet size of the spray material as it leaves the aircraft nozzle, and also to know the droplet size that hits the ground. Variations in these two values are due to a number of factors including evaporation. Ideally, the spray material is evenly distributed over the entire spray block. The coefficient of variance gives an indication of the uniformity of the deposited spray material. The simulation models track the droplets leaving the aircraft and estimate the events encountered by the droplets as they make their way through the aircraft wake and descending onto the spray block (forest or crop area). Some of the spray material is likely to drift away from the target area onto adjacent lands. The amount of spray material deposited outside the spray block is identified via the drift fraction (smaller drift is better since that means the spray material stayed within the spray block or evaporated).

### **The Genetic Algorithm**

Genetic Algorithms [Davi91, Gold89, Holl75] are heuristic search routines that are guided by a model of Darwin's theory of natural selection or the survival of the fittest. Here the fittest means the most highly ranked solution in a large solution space. The basic idea behind the genetic search strategy is to generate solutions that converge on the global maximum (i.e., the best solution in the search space) regardless of the "terrain" of the search space. A typical terrain might resemble the Great Smoky Mountains with many peaks and valleys, an area that is relatively flat, and a highest peak (Clingman's Dome). One characteristic of genetic algorithms is that they are relatively unaffected by hill-climbing or being misled by some local maximum such as ascending Mt. LeConte and assuming that you are on the highest peak in the Smokies since other nearby peaks appear lower, depending on visibility. Likewise, with genetic algorithms the key to finding the global maximum lies in the ability to evaluate and compare possible optimal solutions.

The basic operations involved in a genetic algorithm (GA) are: 1) mate selection, 2) crossover, and 3) mutation. Typically, the major data structure is a binary string representing the possible solutions. In GA terms, a bit string corresponds to an individual, and a set of individuals is called a population. The fitness or strength of an individual is computed using some objective or fitness function, and is used to compare an individual with other individuals in the same population. During mate selection, parent strings are stochastically selected, according to their fitness, from the current population. Then, parent strings are "mated" via crossover to produce offspring for the next generation. Fitter parents contribute more offspring to the next generation than weaker parents because they have a higher probability of being selected for mating. This is the step that models the process of natural selection in nature.

Crossover, the second operation, determines the characteristics of a child or next generation individual. In nature, children inherit good as well as bad features of their parents in varying degrees of dominance. Crossover performs this same function in a GA. One of the simplest crossover approaches is to split each parent string at the same randomly chosen location and swap their tail sections. This ensures a certain amount of inheritance and ideally, the good/strong features will dominate the children. The inheritance of features that produce stronger children throughout the generations is the source of the GA's ability to converge on the global maximum in a relatively short time.

The last basic operation is called mutation. Mutation is that extremely rare "glitch" in the inheritance mechanism that introduces or modifies some feature with unpredictable consequences. Mutation occurs in a GA immediately after the creation of a next generation individual yet before the next generation has become static. Once the new generation becomes static, we move forward in

order for it to become the new current generation. Ideally, mutants would contain some useful features that may have been inadvertently lost in earlier generations.

The simple genetic algorithm described in Goldberg follows these three basic steps [Gold89]. Additional operations and modifications are described as well. One major modification to the simple crossover approach, called two-point crossover, has been shown to be an easily implemented and effective alternate to simple crossover. With two-point crossover, an individual bit string is viewed as a ring and sections of parents are interchanged. This is like cutting equal sized sections from two donuts and swapping the sections to form a new (more appetizing) pair of snacks. Another effective crossover approach is the "greedy" approach described in [Liep90]. They report encouraging results using the "greedy" approach for general set covering problems. Other variations and improvements of the GA operators can be found in [Davi91, Jog89, Pott90, Pott91].

## **SAGA**

Figure 1 shows the architecture for our spray advisor GA. The GA sends a set of AGDISP parameters to the AGDISP simulation model. The AGDISP model calculates and sends back spray results such as the deposition, COV, VMD, and the drift fraction. Based on the fitness function values mapped from these spray results, the GA evolves an improved set of parameters and sends it to AGDISP. This process is repeated from generation to generation for each individual in the population until a satisfactory deposition is found. The corresponding parameter set is returned as the proposed set-up to achieve the desired deposition.

Our development of SAGA so far consists of three stages: Fortran-SAGA, Visual Basic-SAGA (VB-SAGA) 1.0 and VB-SAGA2.0. Fortran-SAGA involved connecting the AGDISP DOS Version

7.0 with a Fortran GA driver. This initial stage of SAGA provided us some preliminary results that indicated high feasibility of optimizing spray parameters using GAs. VB-SAGA1.0 connected a modified AGDISP simulation model (AGDISP DLL) with a VB-GA driver that has many new GA control features. It presents very user-friendly interfaces and much faster speed. VB-SAGA2.0 is an improved version of VB-SAGA1.0 and it has some new features such as the self-adaptive GA and the menu bar.

### **Fortran-SAGA**

The spray simulation model (AGDISP DOS Version 7.0) we used in Fortran-SAGA was implemented using Fortran. Our first thought was therefore naturally to use a Fortran-GA driver to reduce possible compatibility problems. We decided to use a shareware Fortran version of the Simple GA [Gold89] implemented by David L. Carroll [<http://www.staff.uiuc.edu/~carroll/ga.html>]. We added some GA control features such as roulette wheel selection, and 1 and 2-point crossover to this Fortran-GA to suit our project purpose. We focused on twelve specific spray parameters (listed in Table 1) at this initial stage. Other less important or more static parameters are kept constant during our experiments. However, they can become part of the variable parameter set (i.e., we can easily include additional parameters to the parameter set we are searching for) by specifying them at the beginning of each SAGA run.

Both AGDISP DOS 7.0 and the Fortran GA driver read and write information through text files. Our approach to connecting the Fortran-GA with AGDISP was therefore based on these intermediate files. First, we specify the GA characteristics in the GA input file (saga.inp). We

altered the simple GA in order to generate a text file containing the twelve key parameters and all other necessary AGDISP parameters in the format of the input file for AGDISP 7.0. This file is named 'agdisp.inp'. Then AGDISP is initialized by the GA main routine to compute the deposition. Since the GA and AGDISP are two separate programs that run as separate processes, the GA program halts until AGDISP generates and saves the deposition results in an output file, 'agdisp.dep'. This file contains two columns of data, one for downwind distance and the other for deposition. Then the GA resumes execution. It reads in the deposition values from 'agdisp.dep'. The COV of depositions would be computed and combined with the deposition to map the objective function to form the fitness function. Based on the fitness value, the GA evolves an improved set of parameters to send back to AGDISP. This process is repeated for each individual in every generation until a satisfactory deposition and acceptable parameter set are found. Our goal is to maximize the deposition and minimize the COV. That is, get the exact amount of spray material evenly distributed over the spray block. We followed the rule of thumb suggested in [Park82] and set the COV to 0.3. We tested and compared different mapping functions having linear and exponential characteristics, and settled on using the exponential function formulated below. The GA parameters we used for Fortran-SAGA are: population size between 40 and 100, generations between 50 and 200, crossover probability between 0.6 and 0.9, jump mutation probability set between 0.005 and 0.05, and creep mutation probability set between 0.002 and 0.05. Table 2 shows some results from this Fortran-SAGA.

$$Fitness = 3.0 - \exp \left( 1 - 0.04 \left( a \times Deposition - b \times COV \right) \right)$$

There are a few simplifications that we embedded during these testing stages such as setting the user specified COV to a constant, and limiting the droplet size range. The primary reason for these simplifications is that it allows us to begin the spray parameter optimization process fairly quickly after setting up the genetic algorithm. The COV computation was added to a revised version of the AGDISP simulation model for the Forest Service (we use this new version in SAGA1.0 and SAGA2.0, see below). The other simplification deals with droplet size distribution. Here we set the range for droplet size to be between 100 $\mu$ m and 200 $\mu$ m. This range is subdivided into ten droplet size categories with an increment of 10 $\mu$ m. Each droplet size category is assigned a mass fraction of 0.1. We relax these conditions in later versions of SAGA.

### **VB-SAGA1.0**

We use a variant component of AGDISP DOS Version 7.0 for the AGDISP computation engine in VB-SAGA. It is a special dynamic link library (DLL) version that allows VB-SAGA to interface with the DLL through a system of procedure calls and message passing. The DLL has direct access to the aircraft characteristics database (a file containing specific physical and flight characteristics of 124 recognized aerial spray aircraft), and to the spray materials database (a file containing the specifications for a variety of aerial spray materials such as fire retardants, pesticides, and herbicides). The DLL was developed using the FORTRAN programming language while SAGA (both the genetic algorithm search engine and the user interface) was developed using Microsoft Visual Basic 5.0. This new DLL gave us the ability to make full use of the AGDISP simulation model without having to deal with the standard AGDISP user interface.

We implemented a new VB-GA using Microsoft Visual Basic. We made use of one of the convenient features of visual basic, the "Type" statement, to define a new data structure that consists

of the eleven spray parameters (defined as a Single array), the three return values from the DLL, and the fitness value. This new data type is named "individual". We use a real number representation for the individuals. Besides the basic GA parameters such as selection, crossover and mutation, we also added some other features such as Elitism, which will enable the GA to inherit the best individual from the previous generation when turned on. Another useful option is Fitness Scaling which is an advanced GA feature used to overcome "local maximum" convergence problems. With Elitism and Fitness Scaling turned on, VB-SAGA1.0 normally converges in less than 30 generations. The GA population becomes basically homogenous after that and there is no necessity to run the program much longer. We thus provide a Stable Generations option so the user can specify how many stable generations (no changes in maximum fitness) are allowed before stopping SAGA.

The inter-connections between the VB-GA and the AGDISP DLL are now based on the inter-program calls instead of reading and writing through intermediate files. This change led to a significant improvement in speed. A typical VB-SAGA run usually finishes within 2 hours, while a Fortran-SAGA normally takes about 15 hours to finish.

The main interface windows of VB-SAGA1.0 were designed to provide users with flexibility and convenience to group user-defined GA parameters, preset necessary spray parameters, and dynamically view the output information. The top half of the main interface is for GA control parameters specification and the bottom half is for spray parameters display. The user is able to fill in their preferred values for the GA parameters or use our recommended default values. The spray parameter values are displayed dynamically during the VB-SAGA running. The user can also switch the bottom half to a dynamic chart that shows the evolution of the maximum and average fitness. Another main functionality of this VB-SAGA is that we provide the user the option to preset certain

spray parameters due to the practical consideration that it is quite common that some spray parameters can and should be fixed based on the spray requirements.

For comparison and to test the feasibility of SAGA, we designed a pseudo exhaustive test as requested by the Forest Service. We fixed eight spray parameters (listed below) and used the exhaustive combinations of the other three parameters (also listed below). We also ran VB-SAGA1.0 under the same conditions and let SAGA optimize the three remaining spray parameters.

#### Fixed parameters

DSD-VMD	100.0 micron
Temp	10.0 degC
Humidity	75%
AircraftNum	7
BoomWidth	75%
NumNoz	42
BlockWidth	400.0 m
SwathWidth	1.2 m

#### Evolving Parameters

	Lower Bound	Upper Bound	Step
NvFrac	0.001	1.0	0.05
Wind Speed	0.23	4.47	0.1
Boom Height	3.0	30.0	1.0

The exhaustive experiment took four days and the VB-SAGA experiment took 1.5 hours. The results are listed in Table 3. We were glad to find that the best result obtained by VB-SAGA was

among the top 0.1% of the exhaustive search results. These results are good indications that SAGA is capable of finding near-optimal solutions for our spray application in a relatively short time.

After the validation test, we then ran two sets of experiments based on practical spray parameter specification scenarios provided by USDA Forest Service experts. The first set had two groups. The first group evolved 7 parameters but fixed aircraft, DSD-VMD, block size and swath width. The second group fixed boom height, boom length, aircraft and swath width. We ran 10 experiments for each group with the combination of crossovers (0.65, 0.7, 0.75, 0.8, and 0.85) and mutations (0.007 and 0.02). The maximum fitness obtained was 9710.885 based on the first and 9750.734 based on the second group of specifications. The second set had four groups of experiments with different configurations of fixed aircraft and swath width. For each configuration, we ran 10 experiments with the combinations of crossover 0.65, 0.7, 0.75, 0.8, 0.85 and mutation 0.007 and 0.012. The best fitness of these four groups was 8738.82 with fixed aircraft ID 100 and swath width 2.25. The corresponding crossover is 0.8 and mutation 0.02.

These results were evaluated by spray experts and regarded as excellent predictions with high practical importance. More experiments are to be run to test other scenarios and the results are expected to assist practical spray applications, including selecting optimal spray conditions, estimating spray results, reducing spray cost, and minimizing spray drift.

A key issue in the development of SAGA1.0 is the mapping of the drift fraction, COV and VMD using a new fitness function. Drift fraction receives a larger weight due to the necessity to minimize

$$Fitness = 100 \times (A + B + C)$$

where

$$A = [50 \times (1.0 - DriftFrac)]$$

$$B = [25 \times (1 - COV)]$$

$$C = \left[ 25 \times \exp \left( -8.0 \times \left( \frac{VMD}{\dots} - 1 \right)^2 \right) \right]$$

it in order to achieve maximal spray performance.

## **VB-SAGA2.0**

VB-SAGA2.0 inherits most important features of VB-SAGA1.0 and adds some significant new features. The two most important new features are the menu and the self-adaptive GA. The new menu bar was supposed to replace all button functionality on the VB-SAGA1.0 main interface, however some frequently used buttons still remain as short-cuts on the main interface.

The idea for this self-adaptive GA came from the work by Lee and Takagi [Lee93]. We simplified their approach and designed our rules based on their principles. For our self-adaptive SAGA, there are three inputs and two outputs. The three inputs are:

A1:  $(\text{average fitness})/(\text{best fitness})$

A2:  $(\text{worst fitness})/(\text{average fitness})$

A3: change in fitness since last generation

The two outputs are

B1: the crossover rate change

B2: the mutation rate change

Triangular membership functions are used for this fuzzy control. There are altogether 27 control rules for our self-adaptive GA. Some examples of the rules are as follows:

IF A1 is small, A2 is small, and A3 is small, THEN B1 is small and B2 is small.

IF A1 is small, A2 is medium, and A3 is medium, THEN B1 is big and B2 is medium.

IF A1 is medium, A2 is small, and A3 is medium, THEN B1 is medium and B2 is big.

When the self-adaptive feature is turned on, the GA watches the changes of A1, A2 and A3, and makes modifications to B1 and B2 when one or more rules are fired. We use triangular membership functions in fuzzification and defuzzification to obtain crisp outputs. The goal is to force the GA to evolve to the GA parameters that maximize the fitness based on the underlying rules. The new crossover and mutation parameters are restricted such that they can at most change half of their previous values every time. The valid ranges for both crossover and mutation rates are [0, 1].

As with SAGA1.0, we also ran two sets of experiments to test the SAGA2.0 performance with the same initial spray conditions used to test VB-SAGA1.0. That is, for experiment 1, we fixed DSD-VMD, Aircraft Number, Block Size, and Swath Width, while the other spray parameters are left to evolve by SAGA2.0. Experiment 2 was repeated for SAGA2.0 with the same initial conditions as well. SAGA2.0 results are slightly better than SAGA1.0 (9788.236 vs. 9710.885 for experiment 1 and 9802.384 vs. 9750.743 for experiment 2).

We further ran several more tests with SAGA2.0 repeating the conditions of experiment 3 to 6 to compare the performance of SAGA1.0 and SAGA2.0. SAGA2.0 results are significantly better than SAGA1.0 (9500.97 vs. 8738.82 for experiment 3, 9327.26 vs. 8494.48 for experiment 4, 9316.21 vs. 8444.23 for experiment 5, and 9304.84 vs. 8357.87 for experiment 6). A summary of the experiment results by SAGA2.0 and SAGA1.0 are listed in Table 4.

The self-adaptive GA obtained significantly better results than the regular GA for experiments 3 to 6. However, the results of the self-adaptive GA are only a little better than those of the regular

GA for experiment 1 and 2. One possible reason for this difference is the degree of spray parameter restrictions. Experiment 1 and 2 fixed four and seven spray parameters respectively, while experiment 3 to 6 fixed two parameters only. As we know, the crossover and mutation operators apply to individuals in order to change their characteristics (the parameters) and maintain certain diversity. If many spray parameters are already fixed, the effect of crossover and mutation will be reduced by a large extent. The self-adaptive GA in particular relies more on the proper functioning of crossover and mutation operators to optimize crossover and mutation probabilities as well as to optimize spray parameters as the regular GA does.

## **Summary**

The development of SAGA consists of three stages as discussed in earlier sections, Fortran-SAGA, VB-SAGA1.0, and VB-SAGA2.0. The experimental results from these different versions of SAGA were evaluated by the spray experts and regarded as good predictions for practical applications. By using SAGA, the user is able to find optimal or near-optimal spray parameters in order to achieve minimal drift loss, even deposition and desired droplet size. SAGA can usually find the optimal or near-optimal spray parameters in a few hours. If the user presets one or more of the spray parameters, SAGA will spend even less time to find the optimal/near-optimal values due to the reduced complexity of the problem. The user is also able to use SAGA as a regular spray simulation program by specifying some or all spray parameters to obtain spray results, such as drift fraction, VMD and COV. The newly added user-friendly features such as the menu bar, and the self-adaptive GA are also highly welcome by the Forest Service users.

Based on the users' feedback, we will be able to make further modifications to the user interface and the program operation. The USDA Forest Service is working on improving the AGDISP

simulation model to speed up SAGA. A revised fitness formulation is also being proposed by the Forest Service to map the spray results to the fitness as close as possible. In addition, we are making continuous efforts to improve the GA as well as the overall user friendliness.

One new goal of interest is to apply SAGA to optimize more practical factors in spray practice such as the time and cost. An example of important factors affecting the spray time and cost is the flight path of the spraying aircraft. We currently assume the number of flight lines is determined by dividing the block width by the swath width and the aircraft follows these flight lines. However, many blocks have irregular shapes. The problem of flying these blocks is similar to the famous traveling salesperson problem where a salesperson is expected to visit a group of cities in such an order that the total traveling distance is minimized. We expect to add this new optimization procedure to SAGA so that it will be able to find the optimal or near-optimal flight path to reduce spray time and cost.

## References

[Bach84] Bachant, J., and J. McDermott (1984). "R1 Revisited: Four Years in the Trenches," in *AI Magazine*, Vol. 5, No. 3.

[Bila89] Bilanin, A. J., M. E. Teske, J. W. Barry and R. B. Ekblad (1989). "AGDISP: The Aircraft Spray Dispersion Model, Code Development and Experimental Validation". *Transactions of the ASAE* 32(1):327-334.

[Chan83] Chandrasekaran, B., and S. Mittal (1983). "Deep versus Compiled Knowledge Approaches to Diagnostic Problem Solving," in the *International Journal of Man-Machine Studies*, Vol. 19, No. 5, pp. 425-436, November.

[Chan91] Chandrasekaran, B. (1991). "Models Versus Rules, Deep Versus Compiled, Content Versus Form," in *IEEE Expert*, Vol. 6, No. 5, April.

[Davi91] Davis, L., (ed.) (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

[Gold89] Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co.

[Holl75] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press.

[Jog89] Jog, P., J.Y. Suh, and D.V. Gucht (1989). "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem," in the *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishing, San Mateo, CA.

[Lee93] Lee, M. A., and H. Takagi (1993). "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques", in the Proceeding of 5<sup>th</sup> Int'l Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, pp.76-83, July 17-21, 1993.

[Liep90] Liepins, G.E., M.R. Hilliard, J. Richardson, and M. Palmer (1990). "Genetic Algorithm Applications to Set Covering and Traveling Salesman Problems," in *OR/AI: The Integration of Problem Solving Strategies*, (Brown, ed.).

[Mant86] Mantelman, L. (1986). "AI Carves Inroads: Network Design, Testing, and Management," in *Data Communications*, pp. 106-123.

[McDe81] McDermott, J. (1981). "R1: The Formative Years," in *AI Magazine*, Vol. 2, No. 2.

[Mitt89] Mittal, S., and F. Frayman (1989). "Towards a generic model of configuration tasks," in the *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 1395-1401.

[Park82] Parkin, C.S., and J.C. Wyatt (1982). "The Determination of Flight-Lane Separations for the Aerial Application of Herbicides," in *Crop Protection*, 1 (3), pp. 309-321.

[Pott90] Potter, W.D., J.A. Miller, and O.R. Weyrich (1990). "A Comparison of Methods for Diagnostic Decision Making," in *Expert Systems with Applications: An International Journal*, vol. 1, pp. 425-436.

[Pott91] Potter, W.D., J.A. Miller, B.E. Tonn, R.V. Gandham, and C.N. Lapena (1991). "Improving the Reliability of Heuristic Multiple Fault Diagnosis Via The Environmental Conditioning Operator," in the *International Journal of Applied Intelligence*, vol. 2, pp. 5-23.

[Pott92] Potter, W.D., R. Pitts, P. Gillis, J. Young, and J. Caramadre (1992) "IDA-NET: An Intelligent Decision Aid for Battlefield Communication Network Configuration," in the *Proceedings of the Eighth IEEE Conference on Artificial Intelligence for Applications (CAIA'92)*, pp. 247-253.

[Teske89] Teske, M.E., Curbishley, T.B. (1989), "Forest Service Aerial Spray Computer Model FSCBG 4.0 User Manual", C.D.I. Report No. 90-06.

[Teske93a] Teske, M.E., Bowers, J.F., Rafferty, J.E., and Barry, J.W., (1993). "FSCBG: An Aerial Spray Dispersion Model for Predicting the Fate of Released Material Behind Aircraft" in *Environmental Toxicology and Chemistry*, Vol. 12, pp. 453-464.

[Teske93b] Teske, M.E. and J.W. Barry (1993). "Parameter Sensitivity in Aerial Application" in *Transactions of the ASAE*, Vol.36(1), pp. 27-33.

[Teske98a] Teske, M.E. (1998) "AGDISP DOS Version 7.0 User Manual".

[Teske98b] Teske, M. E., H. W. Thistle and B. Eav. (1998a) "New Ways to Predict Aerial Spray Deposition and Drift," in *Journal of Forestry* 96(6):25-31.

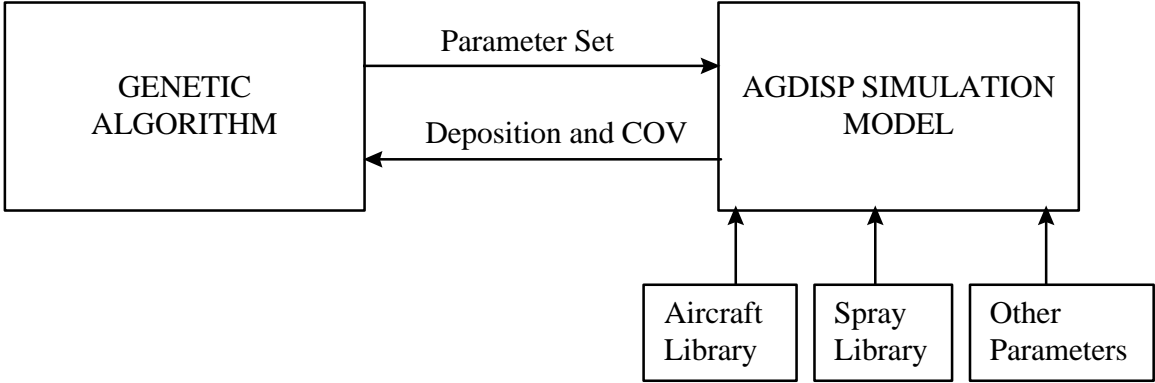


Figure 1. SAGA Architecture.

Table 1. SAGA Parameters and Their Ranges

PARAMETER	LOWER	UPPER
Release Height (m)	1	100
Wind Speed (m/s)	0.5	10.0
Drop Size Distribution ( $\mu\text{m}$ )	100	200
Wind Direction (deg)	-360	360
Number of Nozzles	1	60
Total Flow Rate (gal/min)	0.1	1000.0
Volatile Fraction	0.0	1.0
Flight Speed (m/s)	10	200
Dry Bulb Temperature (degC)	1.0	51.67
Relative Humidity (%)	5.0	100.0
Number of Swaths	1	20
Width of Swath (m)	5	300

Table 2. Preliminary Results from Fortran-SAGA

GENERATION	DEPOSITION (mg/m <sup>2</sup> )
1	98.34
5	99.46
10	102.56
20	108.25
30	116.84
40	119.25
50	124.29
60	137.58
70	146.53

Table 3. Exhaustive Test and SAGA1.0 Comparison

	Exhaustive Test	SAGA1.0 Test
Maximum Fitness	9428.18	9427.26
Non-Volatile Fraction	0.78	0.79
Wind Speed (m/s)	0.28	0.28
Boom Height (m)	6.10	5.78
Drift Fraction	0.031	0.029
COV	0.16	0.17
VMD (micron)	101.63	104.22

Table 4. Results comparison of SAGA2.0 and SAGA 1.0

EXPERIMENT	SAGA2.0 MAX FIT	Cross- over	Mutation		SAGA1.0 MAX FIT.	Cross- over	Mutation
1	9788.236	0.75	0.012		9710.885	0.7	0.02
2	9802.384	0.73	0.011		9750.743	0.75	0.007
3	9500.97	0.89	0.015		8738.82	0.85	0.02
4	9327.26	0.82	0.013		8494.48	0.8	0.02
5	9405.37	0.84	0.017		8444.23	0.85	0.02
6	9386.54	0.83	0.015		8357.87	0.8	0.02