

SAND2000-2846  
Unlimited Release  
Printed November 2000

## **Development and Application of Genetic Algorithms For Sandia's RATLER Robotic Vehicles**

Daniel W. Barnette  
Parallel Computational Sciences Department

Richard J. Pryor  
Evolutionary Computing Methods

John T. Feddema  
Intelligent Systems Sensors & Controls

Sandia National Laboratories  
PO Box 5800  
Albuquerque, New Mexico 87185-0316

### **ABSTRACT**

This report describes the development and application of genetic algorithms for the purpose of directing robotic vehicles to various signal sources. The use of such vehicles for surveillance and detection operations has become increasingly important in defense and humanitarian applications. The computationally parallel programming model as implemented on Sandia's parallel compute cluster Siberia and used to develop the genetic algorithm is discussed in detail. The model generates a computer program that, when loaded into a robotic vehicle's on-board computer, is designed to guide the robot to successfully accomplish its task. A significant finding is that a genetic algorithm derived for a simple, steady state signal source is robust enough to be applied to much more complex, time-varying signals. Also, algorithms for significantly noisy signals were found to be difficult to generate and should be the focus of future research. The methodology may be used for a genetic programming model to develop tracking behaviors for autonomous, micro-scale robotic vehicles.

## Acknowledgments

The authors wish to acknowledge the following:

Funding for this project was obtained through Sandia's Laboratory Directed Research and Development (LDRD) office (Project 10690, Task 03).

Johnny Hurtado, Department 15211, supplied the mathematical models for the signal sources.

Wolfram Research's *Mathematica 4.0* was used to generate signal source and simulated robot path illustrations.

## Contents

Acknowledgments	2
Introduction	4
The Genetic Algorithm	5
Program Representation of the Genetic Algorithm	7
Problem Setup	8
Signal Source Models	9
Results	9
A. Simulation: Genetic Algorithm for Non-Noisy Signal Sources	9
B. Field Test: Coupling the Genetic Algorithm with Sandia's RATLERs	10
C. Simulation: Non-Noisy Genetic Algorithm Applied to Noisy Signals	12
D. Simulation: Genetic Algorithm for Noisy Signal Sources	12
Summary and Conclusions	13
References	13
Tables	
1. Functions and Terminals Available for Genetic Algorithm Decision Tree	6
2. Equations for Signal Source Models	9
3. Configuration of Sandia's Parallel Compute Clusters	10
Figures	
1. Example of a 5-node, 3-level decision tree representing $y=2.3 + 5.9x$ .	
2. Select, top-view convergence sequences of a representative genetic algorithm.	
3. Sandia's parallel compute cluster, ALASKA/SIBERIA.	
4. Various source signal models used.	
5. Peak movement for the various signal source models.	
6. Simulated vehicle movement for the signal source models.	
7. Image of a typical RATLER vehicle.	
8. Model 0 source illustrating algorithmic convergence for 5% signal noise and non-convergence for 8% signal noise.	
Appendices	
A – CEDAR Pseudo-Code Program Listing	
B – ROBOCOP.C Program Listing	
C – ROBOCOP.C Sample Output Listing for Model 0	
D – <i>Mathematica 4.0</i> Graphics Program Listing	
Distribution	

## **Development and Application of Genetic Algorithms For Sandia's RATLER Robotic Vehicles**

### **Introduction**

The emerging technical approach to deal with a challenging, possibly hostile, environment is likely to involve a large number of small, but fairly intelligent, robots. It is envisioned that these can covertly infiltrate a designated area, enter buildings, gather appropriate information, and communicate with and learn from each other. They would also communicate with a smaller number of on-the-scene soldiers backed up by powerful off-line computers that can carry out large-scale information collection, analyses, and simulations. Each robot would have on-board electronics, ground-positioning and communication equipment, an obstacle detector, and some source-analysis capability. Each robot would also have a motor, wheels, and a motor control system. Although the deployed robots would behave autonomously, each robot would communicate information with other robots during the task.

This report documents the effort to generate and apply a robust genetic algorithm to act as a vehicle controlling program for robotic behavior. In a typical scenario, robots are initially distributed randomly in a field and given the goal of locating the emitting source, be it sound or smell. An onboard processor running the algorithm provides instructions to the motor control system that directs the robot to the source location while navigating around obstacles.

The controlling algorithm is generated by a computer code designed to assemble, test, and compare many similar algorithms simultaneously. The code uses trial and error, tournament play, and best fits to generate a decision tree appropriate for the task. Once chosen, the decision tree then becomes the controlling algorithm of choice. The algorithm in decision-tree form is then translated into high-level computer language such as FORTRAN or C, compiled, and downloaded to the robotic vehicles deployed in the task. The robotic vehicles are then controlled by execution of the code using onboard processors, sensors, and memory.

The Sandia RATLER robotic vehicle serves as a research platform for the current effort. Although the RATLER's size precludes its large-scale use at present, further research will see the capabilities of RATLER reduced to micro-scale vehicles. Operationally, it is envisioned that tens to hundreds of these small robots would be deployed to complete a given task.

## The Genetic Algorithm

Building a genetic algorithm is a compute-intensive process by virtue of the fact that it continually attempts to create successive generations of more fit algorithms. Improvement occurs in discrete steps called generations. A generation is composed of a population of individual algorithms each of which is a complete computer program. The number of algorithms considered at one time varies based on the problem; however, hundreds, if not thousands, of genetic algorithms can be considered simultaneously by judicious use of parallel computers. Typically, some algorithms will be more effective than others at doing the prescribed task. Each algorithm is scored for applicability, and its fitness is given a numerical score such that the higher the fitness, the better the algorithm. The goal is to generate an algorithm that correctly solves the problem of interest. However, there is no guarantee that the chosen algorithm is necessarily the best – usually, it suffices.

To create subsequent generations, genetic operators of selection, reproduction, crossover, and mutation are used. The purpose of selection is to choose an algorithm from the current population. In general, this algorithm will be better than most, but it may not be the very best. Reproduction moves a selected algorithm directly into the next generation. Crossover uses the selection operator twice to select two algorithms from the current population that will be combined in some way to form a hybrid algorithm that will be placed in the next generation. Mutation uses the selection operator once to choose an algorithm that will be changed in some way and placed in the next generation. The four genetic operators are discussed in more detail by Koza[1] and Pryor[2]. The development described above proceeds across many generations until a single algorithm is found that meets a convergence criteria. This algorithm is then tested and, if found to be sufficiently robust, implemented as the robotic controlling program.

Pryor[2] gives an example of the program representation of a decision tree making up a genetic algorithm. The basic building block of a tree is called a *node*, with all nodes in the tree having the same fixed structure. The first element of a node specifies the node type, which can either be a function or a terminal. A *function node* performs a mathematical or Boolean operation and generally has branches (nonzero pointers) that point to other nodes. The number of branches depends on the *kind* of function, e.g., add, subtract, multiply. A *terminal node* normally returns a value, does not have any branches (all pointers are zero), and terminates that section of the tree. Other elements within a node are a value position and pointers to other nodes. All of the nodes result in a decision tree that performs a specified task. More detail is given in the next section.

Noise can have a significant impact in actual applications where genetic algorithms are employed. Unless noise is accounted for during its creation, the genetic algorithm may not be able to respond in an appropriate manner. Such was the case in the present applications, as will be shown.

Table 1 lists functions allowed to make up the genetic algorithm.

**Table 1: Functions and Terminals Available for Genetic Algorithm Decision Tree**

No.	Function Name	Mathematical Expression	Comments
<b>FUNCTIONS:</b>			
1	RETURN		Returns a value
2	ADD	+	Adds two values
3	SUBTRACT	-	Subtracts two values
4	MULTIPLY	*	Multiplies two values
5	IFGTEQ	IF (value1 >= value2)	Compares two values
6	COMPUTE ANGLE		Determines which direction robot faces
7	STORE A-REG		Register for each robot
8	STORE B-REG		Register for each robot
9	STORE C-REG		Register for each robot
10	INTEGER ROUND	value=FLOOR(value1+0.5)	Round to nearest integer value
11	STORE AVG X-REG	$(1 - \kappa) * (\text{AVG X-REG}) + \kappa * (\text{AVG X-REG})$	Exponential moving average for value stored ( $\kappa=0.5$ )
12	STORE AVG Y-REG	$(1 - \kappa) * (\text{AVG Y-REG}) + \kappa * (\text{AVG Y-REG})$	Exponential moving average for value stored ( $\kappa=0.5$ )
<b>TERMINALS:</b>			
13	NEIGH 1 XPOS		First nearest neighbor's X location
14	NEIGH 2 XPOS		Second nearest neighbor's X location
15	NEIGH 1 YPOS		First nearest neighbor's Y location
16	NEIGH 2 YPOS		Second nearest neighbor's Y location
17	ROBUG XPOS		X location of robot
18	ROBUG YPOS		Y location of robot
19	ROBUG DIRECTION	1=North, 2=East, 3=South, 4=West	Direction heading of robot (1,2,3, or 4) on grid
20	NEIGH 1 SIGNAL		Signal detected by first nearest neighbor
21	NEIGH 2 SIGNAL		Signal detected by second nearest neighbor
22	ROBUG SIGNAL		Signal detected by robot
23	V-WALL XPOS		North-South wall's X location of corner
24	V-WALL YPOS		North-South wall's Y location of corner
25	H-WALL XPOS		East-West wall's X location of corner
26	H-WALL YPOS		East-West wall's Y location of corner
27	RECALL A-REG		Use A-register's contents
28	RECALL B-REG		Use B-register's contents
29	RECALL C-REG		Use C-register's contents
30	TURN NORTH		Directs robot to face North
31	TURN EAST		Directs robot to face East
32	TURN SOUTH		Directs robot to face South
33	TURN WEST		Directs robot to face West
34	TURN RIGHT		Directs robot to turn right
35	MOVE AHEAD		Directs robot to move ahead
36	VALUE		Store a value

## Program Representation of the Genetic Algorithm

This section describes how the genetic algorithm is represented by individual program elements that make up a decision tree. The representation should always allow complete flexibility in defining programs, yet it must also ensure that the performance of the genetic operations is not too cumbersome. A tree-like structure best meets these requirements.

The basic building block of a tree is called a *node*, with all nodes in the tree having the same fixed structure. The first element of a node specifies the node type, which can either be a function or a terminal. A *function node* performs a mathematical or Boolean operation and generally has branches (nonzero pointers) that point to other nodes. The number of branches depends on the *kind* of function, e.g., add, subtract, multiply. A *terminal node* normally returns a value, does not have any branches (all pointers are zero), and terminates that section of the tree. Other elements within a node are a value position and pointers to other nodes.

Consider the sample decision tree shown in Fig. 1. This tree has five nodes and is three levels deep. The tree is evaluated by starting at its root, or top, and working downward until a terminal node is reached. A terminal node returns a value that is then processed upward in the tree.

To evaluate the sample tree, we begin at the first node denoted by **Start**, a function node, whose kind is specified as **Add**. This kind of function node points to two other nodes that return values to be summed by the **Add** node. At Pointer 1, there is a terminal node that returns a constant value of 2.3. At Pointer 2, there is a function node whose kind is **Multiply**. This node uses Pointers 3 and 4 to point to two **Value** nodes: one that returns the value 5.9, and the other the value of a global variable **x**. These two values are processed by the **Multiply** node, which returns the resultant along with the value of 2.3 to the **Add** node above it. The tree is equivalent to the expression

$$y = 2.3 + 5.9 x$$

where **y** is the value returned by the root node at the top of the tree.

Larger trees used in the robotics program have many more function and terminal types than in the sample tree. The user specifies a maximum allowable number of nodes and depths in the code that generates the genetic algorithm, but typical values are around 800 nodes with a maximum tree depth of 80 levels.

## Problem Setup

The computer code CEDAR has been written to assemble and test genetic algorithms using computer-simulated robots. A listing of pseudo-code for CEDAR is given in Appendix A. Using CEDAR's most current genetic algorithm, the computer-simulated robots solve a set of 90 problems to determine robustness and best fit for search-and-find behavior. At the start of each problem, the simulated robots are placed in arbitrary positions onto a two-dimensional grid and are tasked with finding an arbitrarily placed target. The option exists to arbitrarily place two walls onto the same grid to have the robots learn to avoid obstacles. The walls, if simulated, follow the grid lines in either the  $x$  or  $y$  directions. The goal for the simulated robots is to learn to navigate to the target and to avoid walls if present. The robots have no foreknowledge of either their own positions or the positions of the walls and the target.

Fig. 2 illustrates a representative configuration at startup. Select sequences of graphs show how the simulated robots converge on the two targets. Two signal sources, or targets, are shown in blue. Each signal represents a  $1/r^2$  source whose strength is indicated by gray contours. Red circles randomly spaced about the blue targets represent the robots. In this application, walls were included in the simulation and are shown by heavy, intersecting lines. The position of each robot is given by a coordinate pair  $(x,y)$  which are positive integers. A robot's orientation can be in one of four directions, N, S, E, or W, where north is towards the top of the page. The direction impacts the robot's sensing ability: a robot is programmed to only sense an obstruction if it is positioned in the direction the robot is facing. As shown in the sequence, the simulated robots successfully avoid the walls and converge on the targets, *i.e.*, the signal peaks.

Certain assumptions related to actual robots are inherent in the problem setup. For example, it is assumed that memory on the robot's on-board computer is limited, and only four values of data are stored. Also, communication between robots is limited to the two nearest neighbors. The data that can be communicated consist of positions and signal strengths. Because of assumed limits in the motor control system, only one movement instruction can be returned with each execution of the behavior program. This instruction allows the robot to move ahead one grid point or to turn to a new direction while maintaining position.

CEDAR's computer simulations to generate a suitable genetic algorithm consisted of a population of 200 to 500 robots on each processor, running up to 128 processors on Sandia's compute cluster Siberia (<http://www.cs.sandia.gov/cplant>). Obstructive walls were sometimes included so that the simulated robots would learn to maneuver around obstacles. A photograph of Siberia, a cluster visually similar to an older cluster named Alaska, is presented in Fig. 3. Siberia and Alaska's configuration at the time of this writing is given in Table 3.

**Table 2. Equations for Signal Source Models**

Model #	Equation(s)
0	$1/r^2$
1	$-e^{-2r^2} [\cos(\pi + 20r + 5\theta - t) - 4]$
2	$-e^{-2r^2} [\cos \pi \cos(10r - 4t) \cos(5\theta - 4t) - 4]$
3	$-e^{\frac{-(x^2 - y^2)}{100\sigma^2}} \frac{1}{\sigma\sqrt{2\pi}} [\cos \pi \cos(2x - 4t) \cos(4y - 4t) - 4], \quad x \geq 0$ <p>and</p> $-e^{\frac{-(x^2 - y^2)}{\sigma^2}} \frac{1}{\sigma\sqrt{2\pi}} [\cos \pi \cos(2x - 4t) \cos(4y - 4t) - 4], \quad x < 0$ <p>where</p> $r = \sqrt{x^2 + y^2}, \quad \theta = \arctan\left(\frac{y}{x}\right), \quad t = \text{time}, \quad \text{and} \quad \sigma = x^2 + 2$

### Signal Source Models

A total of four signal source models were considered. The equations governing each are given in Table 2. Mathematical representations of the models, generated using *Mathematica*[3], are shown in Fig. 4. The simplest model, Model 0, consists of a  $1/r^2$  steady-state signal, as illustrated in Fig. 4a. Three time- and spatially-varying models were also considered. Illustrated in Figs. 4b, c, and d are Models 1, 2, and 3, respectively. Each of these functions has multiple local peaks that move around considerably as the robots search for the most likely signal peak. A sample of the maximum-peak movement for the unsteady models is given in Figs. 5a, b, and c.

### Results

#### A. Simulation: Genetic Algorithm for Non-Noisy Signal Sources

The first attempts at generating genetic algorithms centered on modeling each signal source model. Wall-like obstructions were placed randomly on the grid so that the simulated robots would learn to maneuver around them. This became a very computationally intensive process since the signal sources for Models 1, 2, and 3 were time-varying and complicated, especially near the multiple center peaks.

Various attempts were made to accelerate the convergence. For example, more functions, such as exponential and sinusoidal, were added to the code from which the genetic algorithm would be generated. The reasoning behind this approach was that if the algorithm needed an exponential function that would otherwise be built from Taylor-series-like terms, for example, then adding this function to the list of possible functions to

**Table 3. Configuration of Sandia’s Parallel Compute Clusters**  
(go to <http://www.cs.sandia.gov/cplant> for more information).

	ALASKA	SIBERIA
Processor	DEC Alpha EV 56	DEC Alpha EV6
Processor speed	500 MHz	500 MHz
Operating system	Linux	Linux
Total number of nodes	270	592
Number of processors per node	1	1
Memory per node	196 MBytes	560 nodes have 256 MBytes; 32 nodes have 1 GByte
Parallel I/O bandwidth (scalable)	40 MBytes/sec over 4 network connections (enfs)	40 MBytes/sec over 4 network connections (enfs)

be chosen would negate the necessity to build the Taylor series. However, this also proved to be slowly convergent, possibly because the functions added too much complexity to the simpler algorithm being generated at the time. Also, adding more choices to the list of available functions algebraically increased the algorithm’s number of options from which to choose as a decision tree was formed. That is, the function could be considered for use in each node in a tree. As a result, convergence to a best-fit algorithm became extremely tedious.

To alleviate the convergence problems, the authors decided to examine the possibilities of using the algorithm generated for the steady-state Model 0 for the time-varying models. No walls were simulated since it was decided that the to-be-conducted field tests would not initially contain obstructions.

Simulated vehicle movement using the genetic algorithm derived for Model 0 was generated using the code ROBOCOP, listed in Appendix B. The genetic algorithm generated by CEDAR is inserted in the function *MoveGA*, the last function listed in Appendix B, to complete the code. Sample output of ROBOCOP is presented in Appendix C. The *Mathematica* program used to graphically display the results is given in Appendix D.

The Model 0 result is illustrated in Fig. 6a, while the application of the identical algorithm to the remaining models is shown in Figs. 6b, c, and d. As shown, the algorithm worked very well for all signals for the sampling rates considered. This seems a surprising result considering the complexity of the other models compared to Model 0. However, further thought leads one to conclude that a single-peak-finding algorithm for steady-state signals may well be sufficient even for time-varying, multiple-peak signal sources as long as sampling rates are high relative to peak movement.

### B. Field Test: Coupling the Genetic Algorithm with Sandia’s RATLERS

The rovers used in the field tests were Sandia’s Robotic All-Terrain Lunar Exploration Rover (RATLER) vehicles. Typical RATLER vehicles are shown in Fig. 7. The largest are approximately the size of two shoeboxes placed side by side. RATLER vehicles were

developed by Sandia as a prototype vehicle for a lunar mission. Each vehicle is typically equipped with an Intel 486 computer, differential GPS receiver, spread spectrum two-way packet radio, electronic compass and tilt sensors, video camera, and RF video transmitter. Three RATLERs of the type shown in Fig. 7a were used during the tests. This was the minimum number needed for vehicle-to-vehicle communications as provided for in the genetic algorithm.

The base station equipment with which the RATLERs stay in constant communication consists of a Pentium laptop computer, spread spectrum two-way packet radio, differential GPS base receiver, RF video receiver, and a battery power source. The equipment is contained within a small trailer for mobility. The base station sends commands and queries to the RATLERs over the packet radio. The communication network is configured as a token ring. Hence, if the base station becomes non-functional, the vehicles will continue to communicate. Also, if either the vehicle or base station misses its turn to communicate, communications can be re-established after a specified delay.

During field tests, the operator places the RATLERs in autonomous navigation mode. A live video image from one of the vehicles can be displayed on the laptop along with the current position of the vehicle on a Geographic Information System (GIS) map. Multiple RATLERs are driven to operator-specified set points using differential GPS and a magnetic compass, where they are allowed to navigate on their own to the source using the genetic algorithm controlling program. The positioning accuracy of the vehicles is typically 1 meter.

As a result of its success in finding the peaks of all signal models, the algorithm for Model 0 was implemented on robotic rovers for field tests. The signal source was a loud speaker placed in a large field so as to closely simulate the  $1/r^2$  Model 0 source. The RATLERs were placed in a random position about the source. The genetic algorithm previously loaded into the RATLERs onboard memory was then executed and the vehicles were allowed to move about as directed by the algorithm. No obstructions were placed between the rovers and signal source.

Direct observations of the ensuing test were that the vehicles found the source but wandered significantly beforehand. The wandering was attributed to signal noise that may have been caused by nearby vehicle traffic, wind, and possibly electronic component tolerances. Signal noise was not due to vehicle movement since signals were generated only after each RATLER stopped momentarily. Base station equipment recorded the noise to be as high as 10% of the signal source. Noise was not modeled in the initial algorithm for Model 0.

It was also discovered that a RATLER acting alone showed nearly identical behavior to that when all three were attempting to locate the target. This apparently indicated that the vehicles were not communicating with each other even though provisions such as registers were made available with the functions given in Table 1. Computer simulations using only one simulated robot reinforced the conclusion that the vehicles were not communicating as originally believed. Implications are that the vehicles were acting

autonomously and not collectively, as should be in the case of a swarm of vehicles. It is unknown why this occurred, but it is certainly an area for future research.

### C. Simulation: Non-Noisy Genetic Algorithm Applied to Noisy Signals

Noise was introduced into the signal source used in Model 0's simulator. The original genetic algorithm was then used to perform a post-mortem simulation and analysis of the field tests. A random number generator was used to perturb the original signal within a user-specified percentage at each time step. This would hopefully reveal the effects of noise on the robot convergence path.

Results of the robot convergence path are illustrated in Fig. 8. Shown in Fig. 8a is the convergence path for a 5% noise signal. The simulated rover finds the peak even though the signal strength is slightly perturbed. However, an 8% perturbation causes the simulated rover to never converge, as illustrated in Fig. 8b. Thus, the genetic algorithm for Model 0 is apparently robust enough to handle small perturbations to around 5%. However, higher perturbations cause the robot to wander as observed in the field test.

At this point, two alternatives became obvious to alleviate the wandering. The first approach was to lower the signal noise in some way. This was quickly abandoned, since the factors causing the noise levels were out of the operators' control. It is also possible that higher noise levels may exist during actual future applications and that these levels could not be predicted beforehand.

The second approach was to introduce noise in the code that generates the genetic algorithm.

### D. Simulation: Genetic Algorithm for Noisy Signal Sources

It was thought that the genetic algorithm for Model 0 could be regenerated with the ability to process noisy signals. The new algorithm, if successful, would allow the rovers to find valid signal peaks even through a 'dirty' signal. Once again, so as not to add more complexity to the problem, walls were not modeled.

Significant computer time was spent on this approach, but without much success. Project deadlines prevented a thorough attack on this problem, but preliminary analyses indicated simulated rovers would find their way to a fairly large distance from the peak, and no closer. It is not entirely understood why this happened. However, better convergence might be achieved if the algorithm ensured each rover communicated with some of its nearest neighbors, thereby triangulating the signal source. As has been discussed, rover-to-rover communications were apparently not occurring in the original Model 0 genetic algorithm. Hence, another area of research should include a study of the ability of genetic algorithms to intelligently process noisy signals.

## Summary and Conclusions

This report documents a research effort in which a genetic algorithm code was developed and ported to Sandia's parallel compute clusters. The code was modified to use the MPI message passing protocol. Efficiency was improved by reducing excessive message passing between the master node and slave nodes. The ability to investigate time-varying signal sources was added to the original code. Visualization schemes were developed and implemented for investigating simulated robot behavior before running field tests with actual hardware.

The result of this effort, a genetic algorithm, has been implemented in hardware as a robot controlling program. Field tests were conducted using Sandia's RATLER robotic vehicles attempting to locate a low humming stereo speaker. Tests were successful, though significant wandering was observed that was not evident during computer simulations. This behavior is believed to be due to signal noise. Project deadlines prevented generating a genetic algorithm that could filter noise and locate the peak efficiently. It was also noticed that the algorithm resulted in autonomous, rather than collective, robot behavior. The factors that govern this behavior should be a topic of future research.

An interesting finding of this research was the fact that a genetic algorithm developed for a simple test case proved very robust for more complex applications and signals. Computer simulations showed that the algorithm developed for a simple  $1/r^2$  case proved sufficient for much more complicated applications. This should be kept in mind in any future research involving applying genetic algorithms to complicated applications: keep it as simple as possible. Extensions of simple algorithms may be possible for much more complicated applications.

In conclusion, the authors believe genetic algorithms have a strong future at Sandia, especially when applied to problems that have no definitive analytical answers, but where a 'good' solution will do. Future areas of research should include an approach that ensures rover-to-rover communications and the study of the effects of noisy signals on obtaining acceptable rover behavior. It is hoped that this report gives impetus to additional research in these areas so that more robust genetic algorithms may be developed.

## References

1. Koza, J. R., Genetic Programming, On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
2. Pryor, R. J., "Developing Robotic Behavior Using a Genetic Programming Model," SAND98-0074, Sandia National Laboratories, Albuquerque, New Mexico, January 1998.
3. Wolfram, S., The Mathematica Book, 3<sup>rd</sup> edition, Wolfram Media & Cambridge University Press, 1996.

[Intentionally Left Blank]

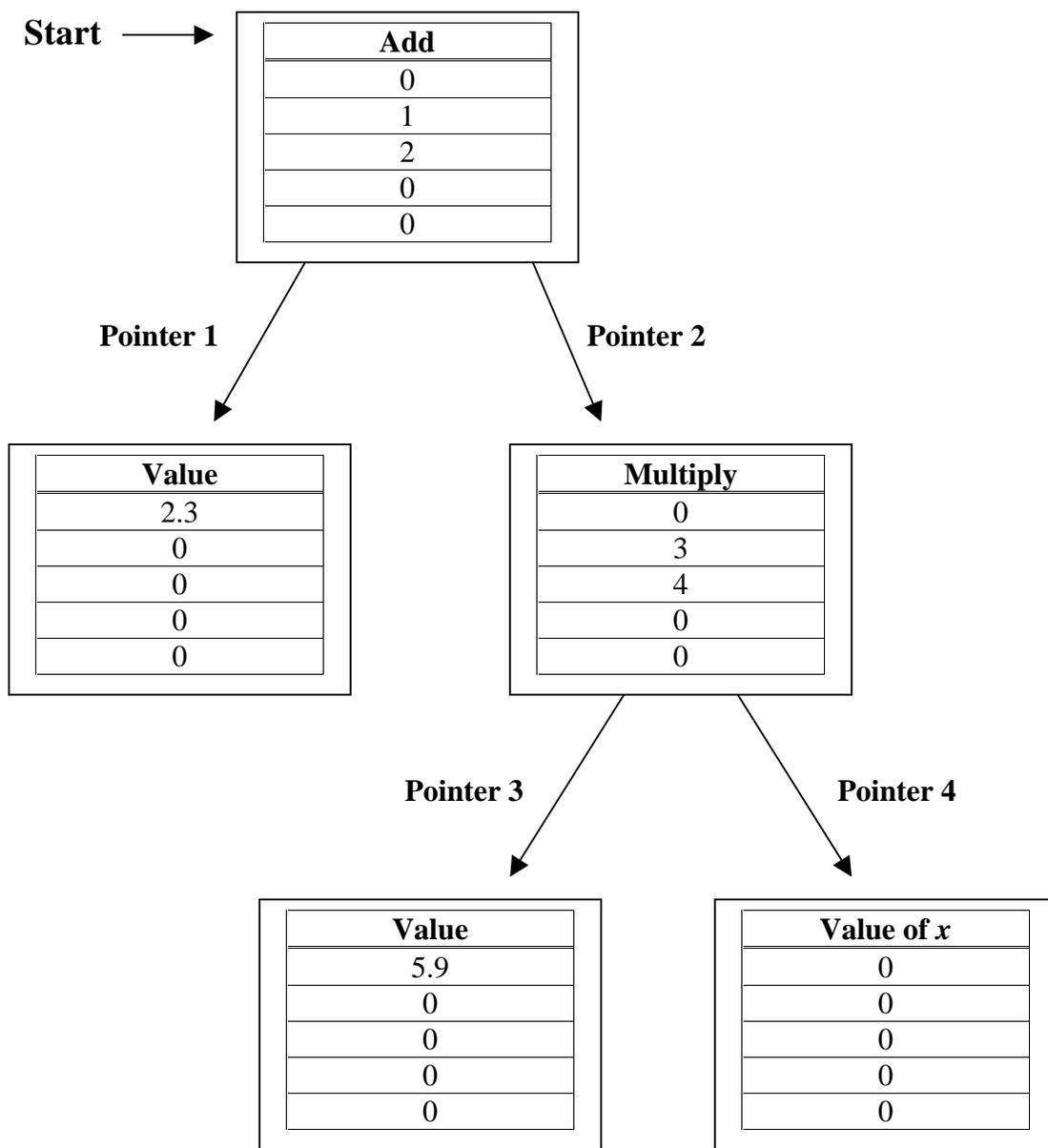
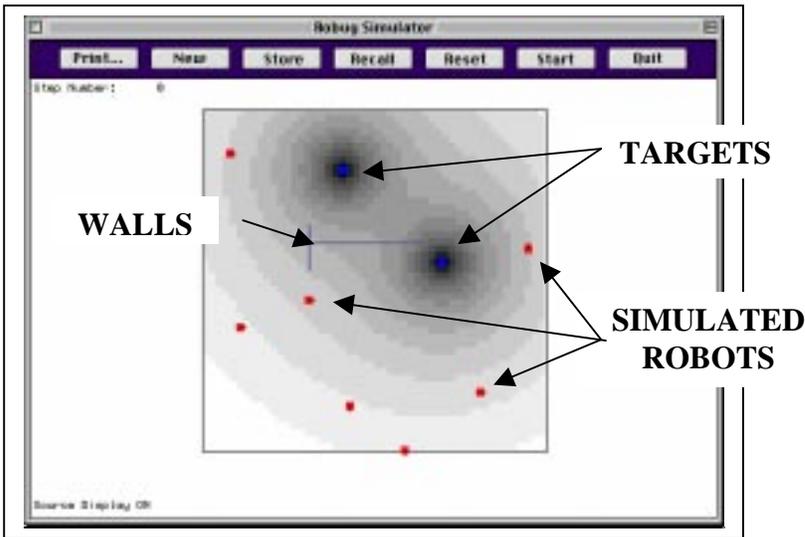
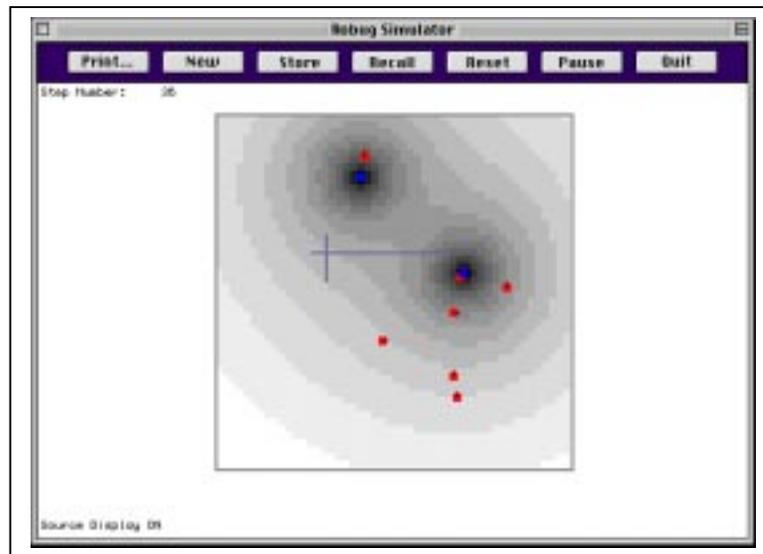


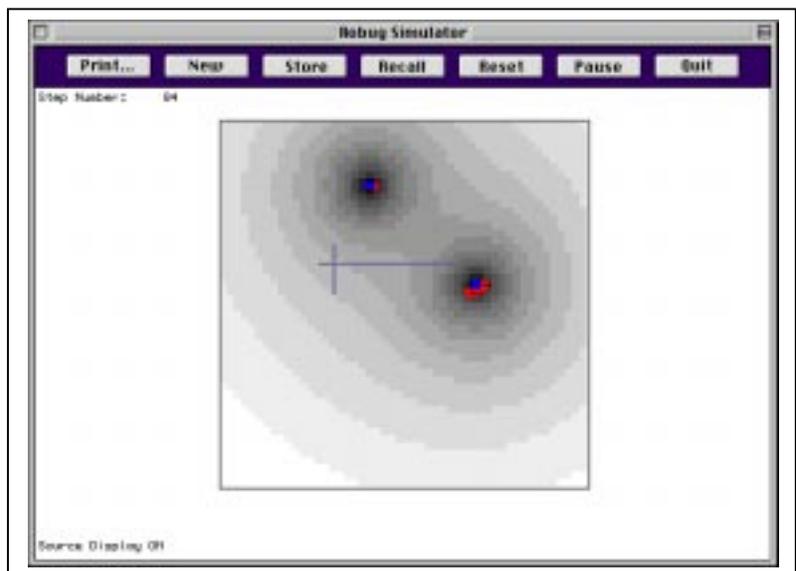
Figure 1. Example of a 5-node, 3-level decision tree representing  $y = 2.3 + 5.9x$ .



a) Step 0



b) Step 36

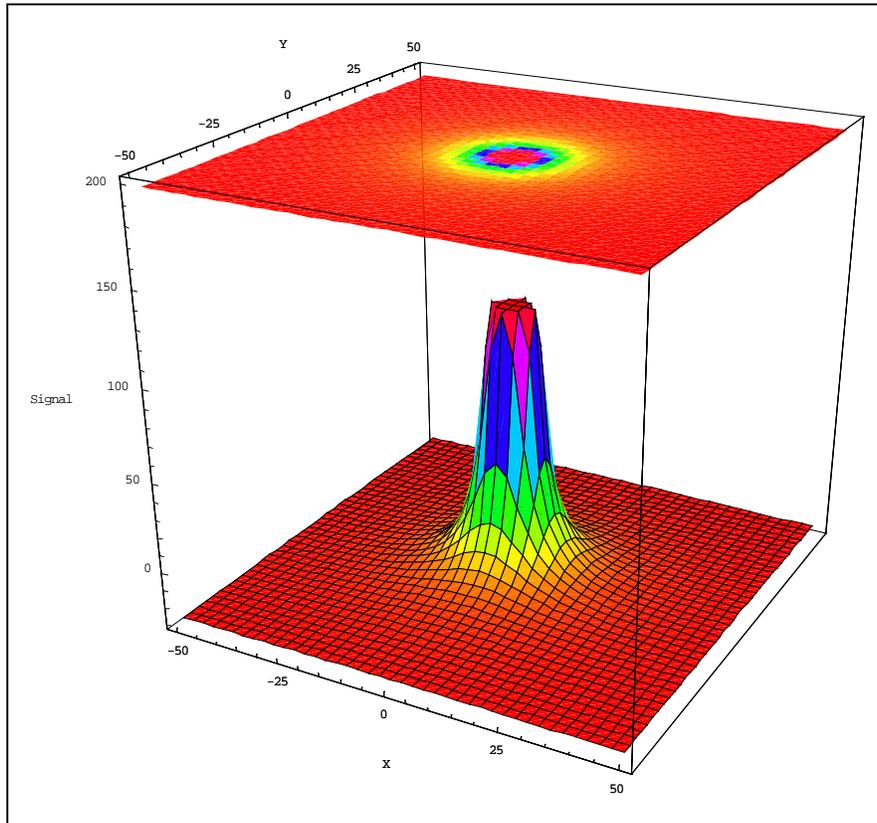


c) Step 84

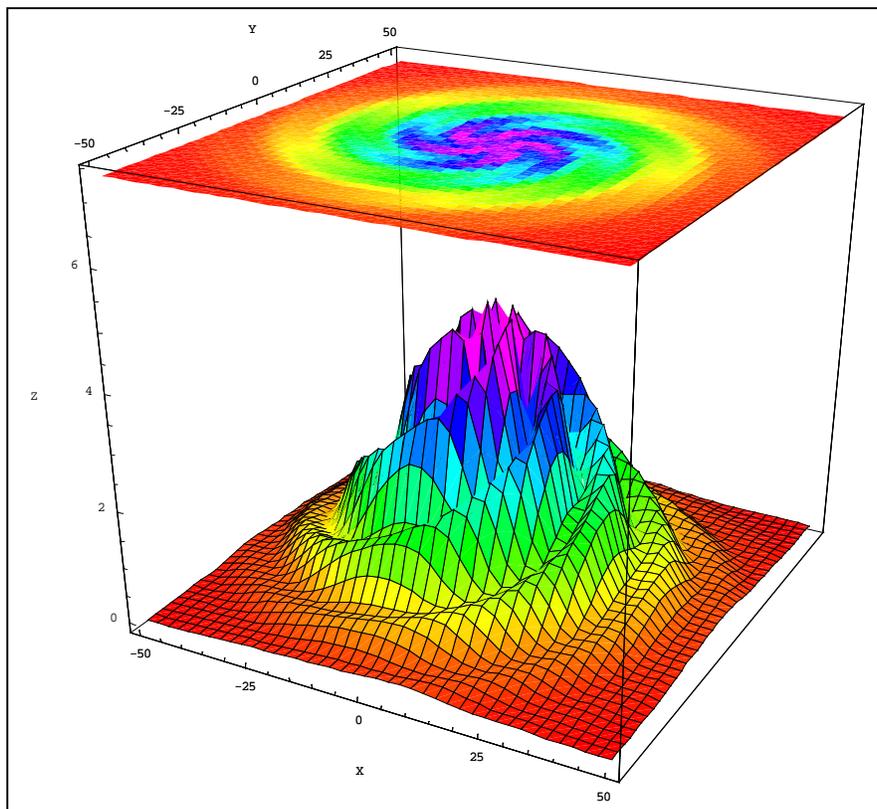
Figure 2. Select, top-view convergence sequences of a representative genetic algorithm.



Figure 3. Sandia's parallel compute cluster, ALASKA/SIBERIA. For more details, go to web site <http://www.cs.sandia.gov/cplant>.

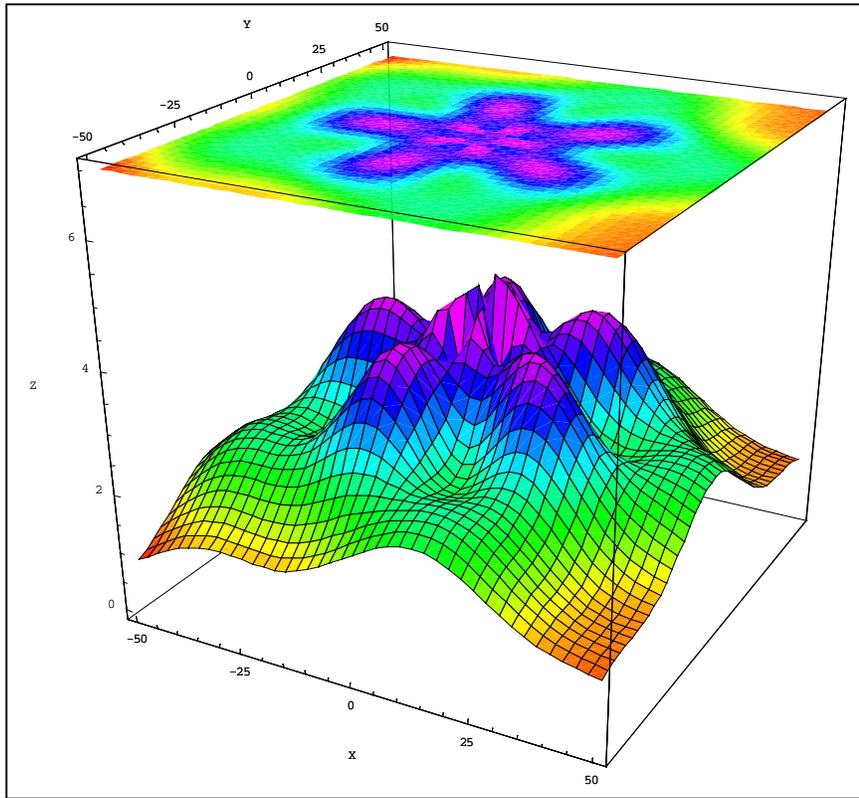


a) Model 0

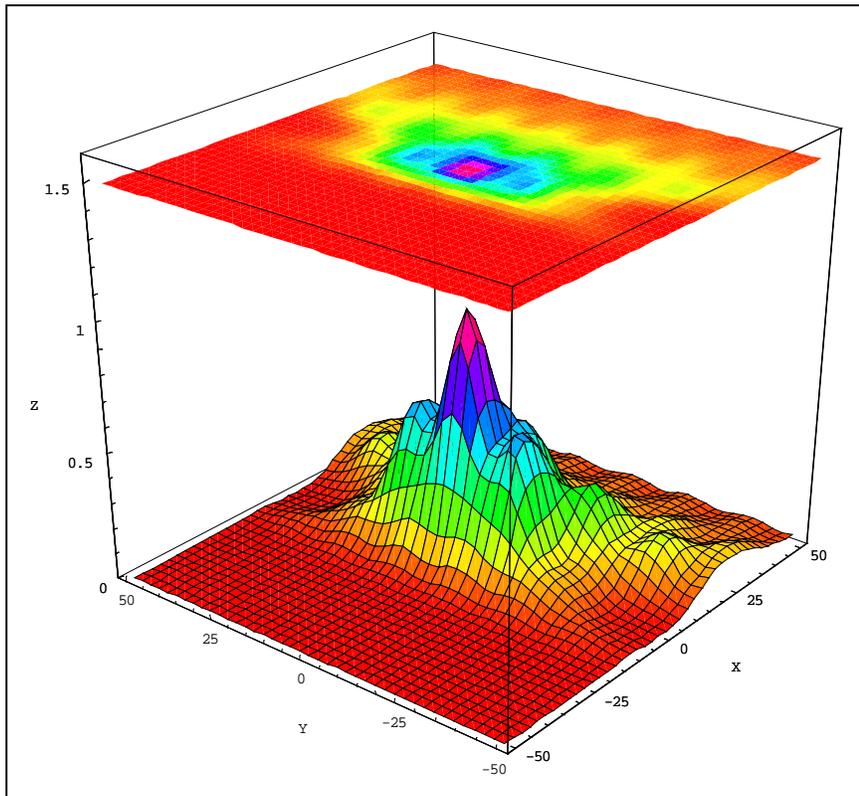


b) Model 1

Figure 4. Various source signal models used.

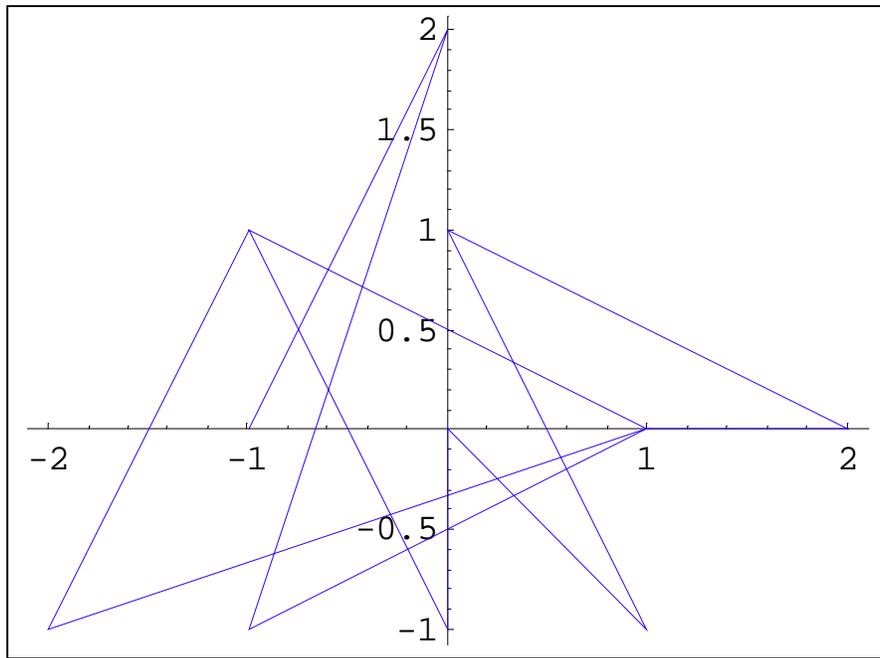


c) Model 2

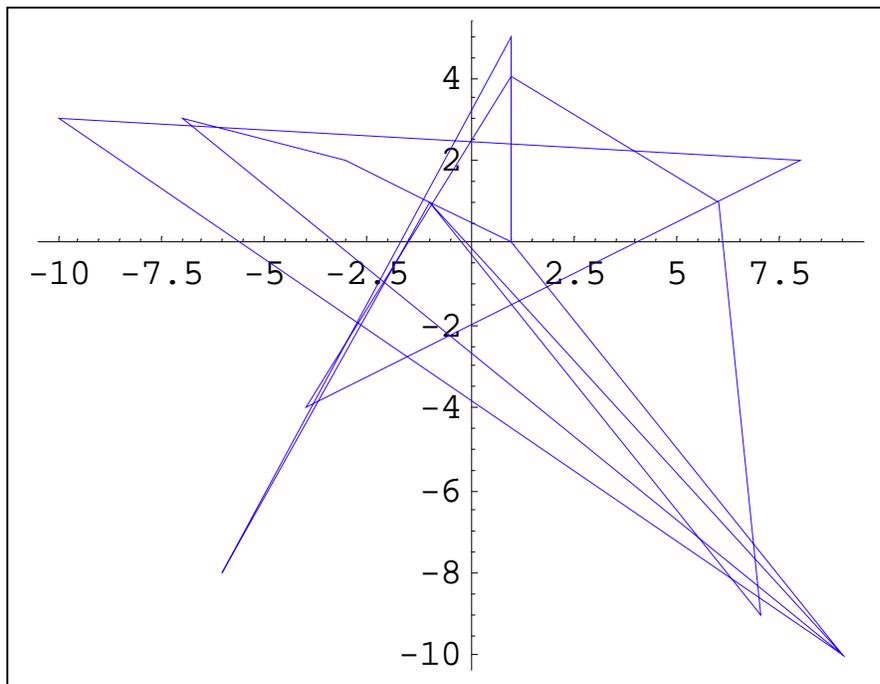


d) Model 3

Figure 4. Concluded

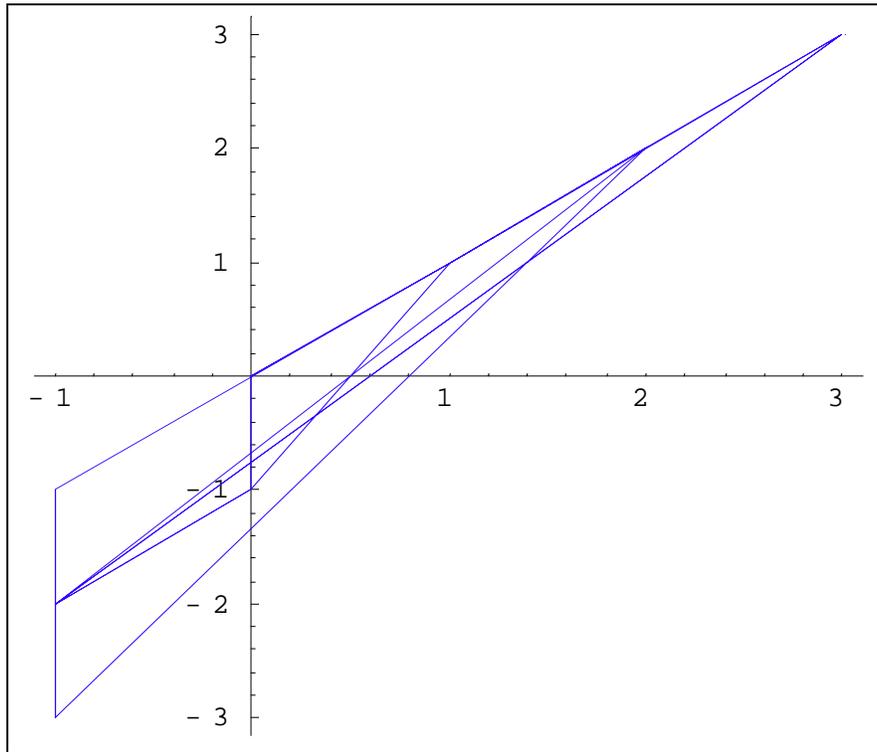


a) Model 1,  $0 < t < 10 \text{ sec}$



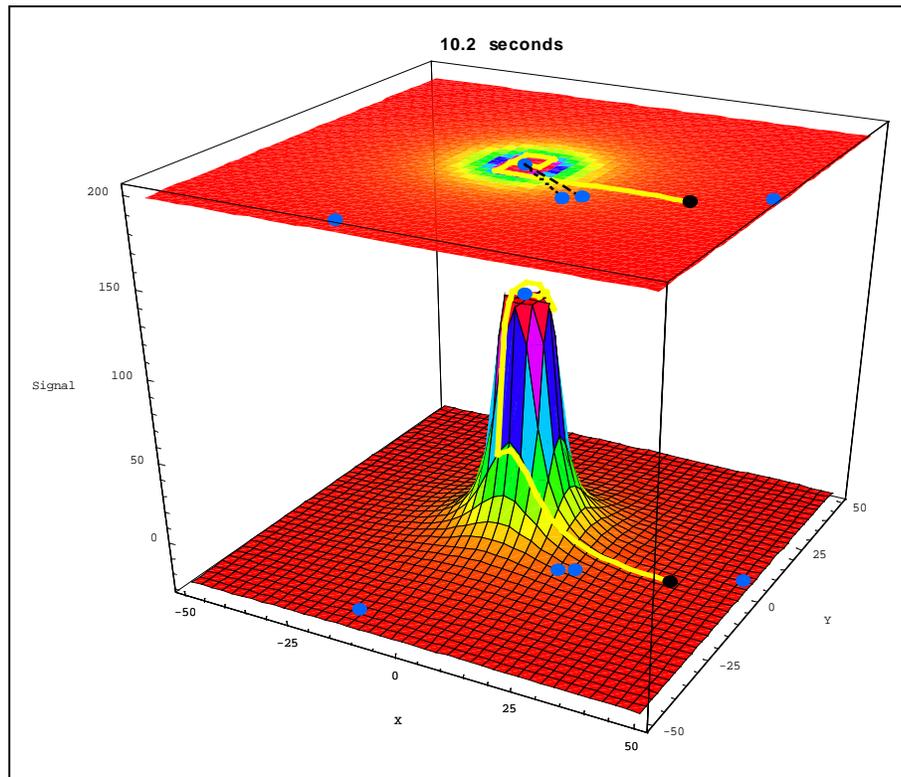
b) Model 2,  $0 < t < 10 \text{ sec}$

Figure 5. Peak movement for the various signal source models.

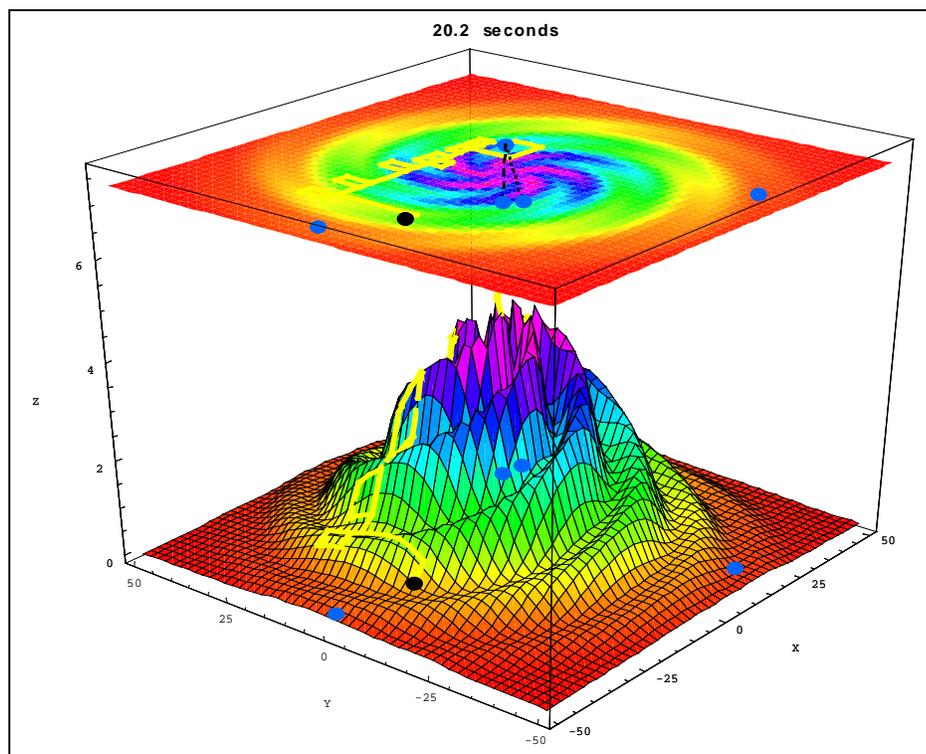


c) Model 3,  $0 < t < 10 \text{ sec}$

Figure 5. Concluded.

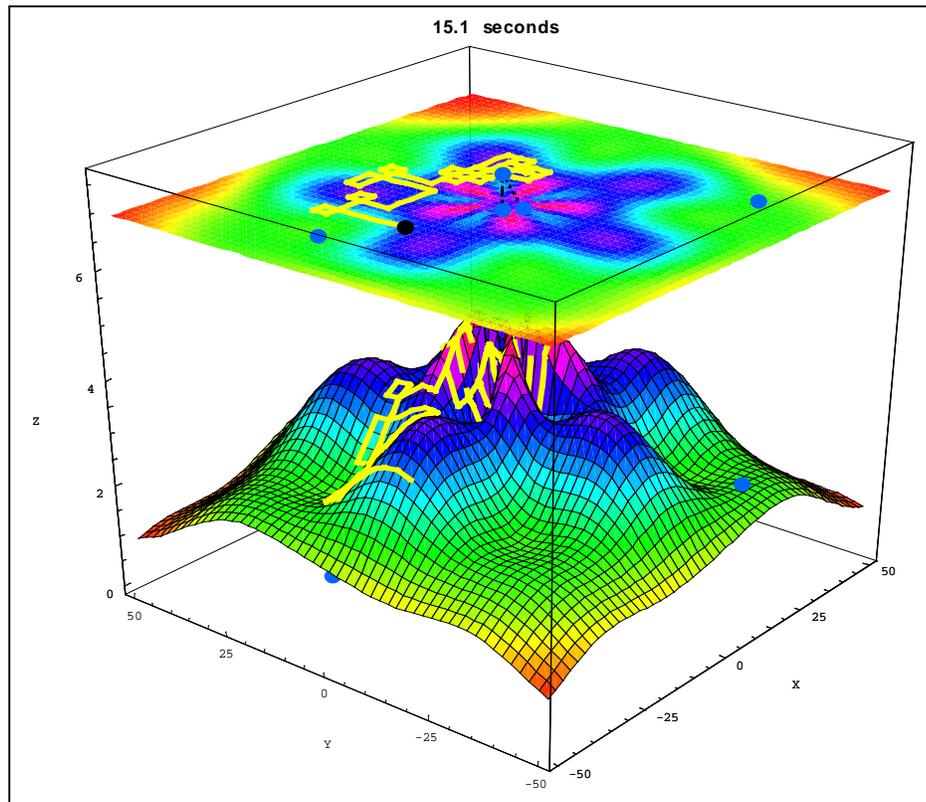


a) Model 0,  $t=10.2 \text{ sec}$

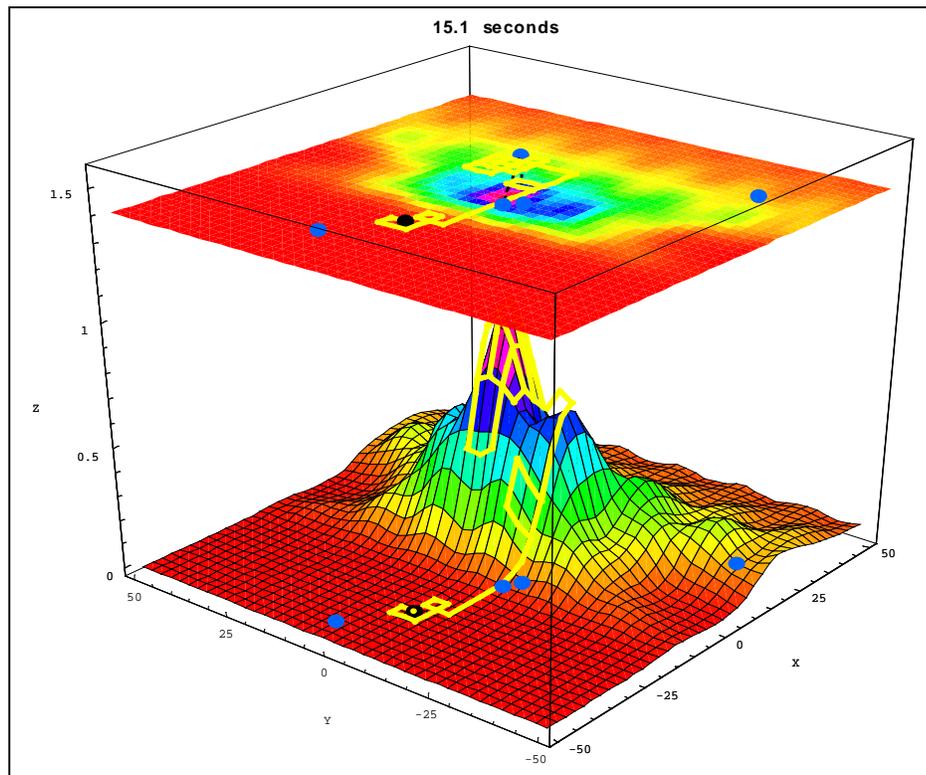


b) Model 1,  $t=20.2 \text{ sec}$

Figure 6. Simulated vehicle movement for the signal source models.



c) Model 2,  $t=16.1 \text{ sec}$



d) Model 3,  $t=15.1 \text{ sec}$

Figure 6. Concluded.

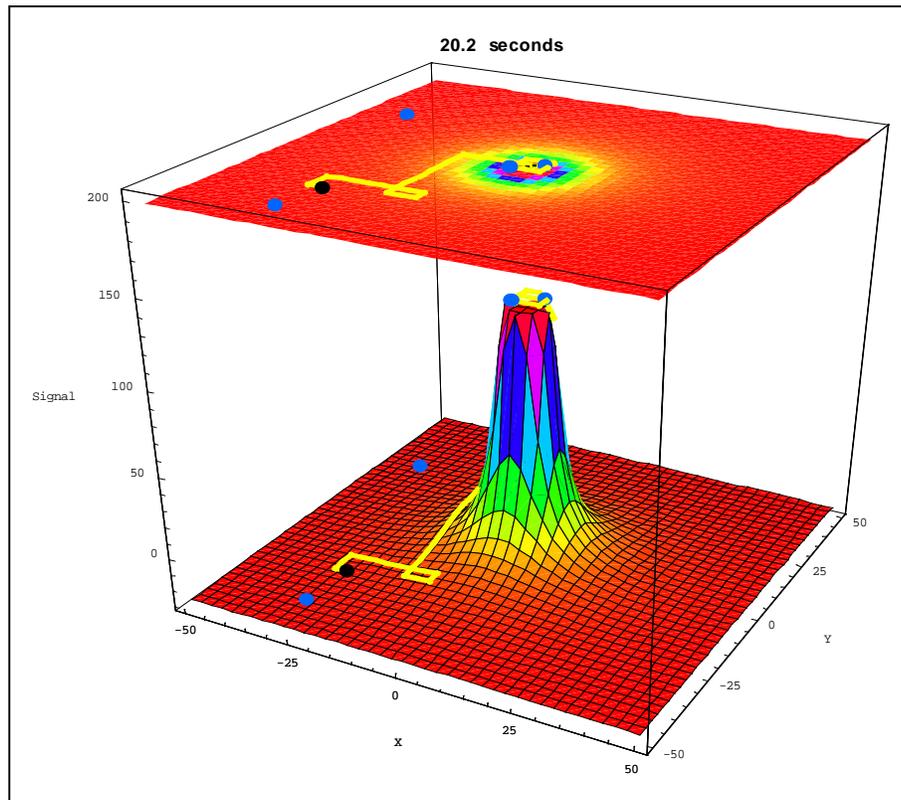


a)

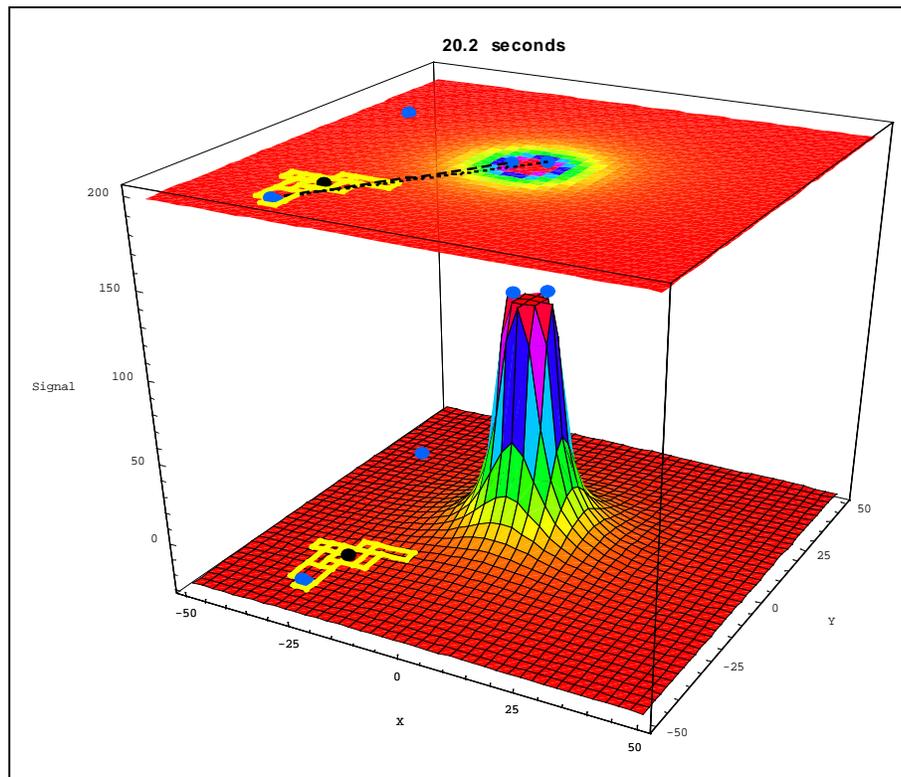


b)

Figure 7. RATLER vehicles developed at Sandia National Laboratories.



a) 5% signal noise



b) 8% signal noise

Figure 8. Model 0 source illustrating algorithmic convergence for 5% signal noise and non-convergence for 8% signal noise.

[Intentionally Left Blank]

EXTERNAL DISTRIBUTION:

Copies:      Name/Entity:

4      Santa Fe Institute  
Attn: Melanie Mitchell (2 copies)  
        James Crutchfield (2 copies)  
1399 Hyde Park Road  
Santa Fe, New Mexico 87501

INTERNAL DISTRIBUTION:

Copies:	Mail Stop:	Name/Org:
1	0316	S. S. Dosanjh, 9233
1	0318	G. S. Davidson, 9212
5	0318	R. J. Pryor, 9212
1	0318	M. Boslough, 9212
1	0318	K. Boyack, 9212
1	0318	R. Hightower, 9212
1	0825	W. H. Rutledge, 9115
1	0321	W. J. Camp, 9200
10	0316	D. W. Barnette, 9233
1	1110	D. E. Womble, 9214
1	0321	A. L. Hale, 9220
1	0316	G. S. Heffelfinger, 9235
1	0847	R. W. Leland, 9226
1	0819	E. Boucheron, 9231
1	0310	P. Yarrington, 9230
1	0847	D. R. Martinez, 9124
1	1002	P. Eicker, 15200
1	1003	R. Robinett, 15211
5	1003	J. Hurtado, 15211
1	1003	J. Feddema, 15211
1	1003	C. Lewis, 15211
1	1004	D. Schoenwald, 15221
1	1010	G. R. Eisler, 15222
1	0839	G. Yonas, 16000
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9616
1	0612	Review & Approval Desk, 9612 For DOE/OSTI