N94- 35052

A Genetic Technique for Planning a Control Sequence to Navigate the State Space with a Quasi-Minimum-Cost Output Trajectory for a Non-Linear Multi-Dimensional System

C. Hein,
Martin Marietta,
Moorestown NJ 08057

A. Meystel,
Drexel University,
Philadelphia, PA 19104

Abstract:

There are many multi-stage optimization problems that are not easily solved through any known direct method when the stages are coupled. For instance, we have investigated the problem of planning a vehicle's control sequence to negotiate obstacles and reach a goal in minimum time. The vehicle has a known mass, and the controlling forces have finite limits. We have developed a technique that finds admissible control trajectories which tend to minimize the vehicle's transit time through the obstacle field. The immediate application is that of a space robot which must rapidly traverse around 2-or-3 dimensional structures via application of a rotating thruster or non-rotating on-off thrusters. An air bearing floor test-bed for such vehicles is located at the Marshal Space Flight Center in Huntsville Alabama.

However, it appears that the developed method is applicable to a general set of optimization problems in which the cost function and the multi-dimensional multi-state system can be any non-linear functions, which are continuous in the operating regions.

Other applications include the planning of optimal navigation pathways through a traversability graph; The planning of control input for under-water maneuvering vehicles which have complex control state-space relationships; The planning of control sequences for milling and manufacturing robots; The planning of control and trajectories for automated delivery vehicles; And the optimization of control for automotive racing vehicles and athletic training in slalom sports.

Introduction:

Many optimization techniques have been developed such as: Bolza, Mayer, Lagrange, Green, Gradient Methods, Dynamic Programming, and Optimum Spacing of Corrective Thrusts[2]. Other methods that are based upon searching tend to be less efficient, exhausting much time searching the entire space at a given resolution level.

Techniques that are based on direct methods tend to be limited to specific types of cost functions and systems. In such methods, the system must be expressible as a single function. For example, a discrete maximum principle method (from Pontryagin), is based upon Calculus of Variations. The problem is formulated as finding the control function, u[k], which minimizes J=Sum of F[x,u,k] with constraint x(k+1) = f[x,u,k]. Where x are the system states, f is the system transition function, and F is the cost function. The solution is found by first, defining the costate vector, p(k), and the Hamiltonian, H[x,u,p(k+1),k], as:

H = F[x,u,k] - <p(k+1, f[x,u,k]>

Let p and x take variations, lambda. Then form 2*n first order difference equations / canonical state equations. Solve:

```
 partial dH
------------ = 0,
 partial dU
```

for optimum (minimum) relative to the boundary conditions. Then apply the condition of transversality to find u.

Background:

The purpose of our algorithm is to determine and plan an ideal path. Since it is a plan, unpredictable perturbation forces, and sensor, actuator errors cannot be considered. It is assumed that a final design must incorporate additional methods such as closed-loop feedback, periodic re-evaluation, or variable structure control methods in conjunction with this path-planning method. Planning is necessary since, none of the feedback methods could be used without a basic path plan such as that developed by this technique. In the absence of general rules for directly deriving an optimal control sequence, given arbitrary constraints such as vehicle dynamics, control limitations, and an arbitrary obstacle environment, a basic strategy is to search the control space for a control sequence which avoids obstacles while minimizing the time to reach the goal. We call such a control space search an "input-space" search.

An alternate method would be to search the "output-" space for an optimal trajectory, and then to find a corresponding admissible control sequence that produces it. However, there are many trajectories which cannot be achieved through any admissible control sequence due to limitations in vehicle dynamics. Namely, these limitations are: vehicle mass and the appliable acceleration, force, or power. The set of all trajectories includes those which are produced from admissible control, plus those which cannot be.

Likewise, the input-space corresponds to all trajectories that can be traversed, plus those which cannot be due to obstacles or other constraints. Consequently, it is not obvious which of the two candidate search spaces is more efficient to search. The decision then rests upon ease of implementation or computation.

Searching the input space requires posing various permutations of admissible control sequences and testing their outcomes for conformance to the output constraints, which in our case would be obstacle collisions. The output trajectory due to a control input sequence is easily computable as long as we have a system equation. Since the output trajectory is a function of the control input (and not vice-versa), this is a one-to-one mapping. In other words, there is exactly one output trajectory corresponding to a given control input sequence.

Searching the output-space requires posing various valid output trajectories, and working backwards for each one to derive an admissible control input sequence that would produce it, if any exist. The inverse to the system equation is required to compute the input sequence from the output trajectory. Therefore, since it appears more direct and tends to be more well-formed (since inversion is avoided), we have chosen to search in the control input space.

In general, the control space for this type of problem consists of a vector function, C(t), which is continuous in time. At any time instance, t, C(t) represents the control force vector on the vehicle. That is, it represents a force magnitude and direction (F,theta). Both quantities may change abruptly at any time. The magnitude is usually limited to some admissible range, Fmax, so it is continuous from 0 to Fmax, while the direction may vary continuously from 0 to 360-degrees.

The continuous quantities (F,theta) can be quantized into discrete quanta. As the quanta grow infinitesimally small, the quantization approaches the continuous function. Similarly, the functions can be discretized in time, so as to be approximated by discrete time series. Again, as the sample period grows infinitesimally small, the discrete approximation approaches the continuous function in time. Therefore, through quantization and discretization, the time/value continuous system can be approximated with arbitrary precision.

Discretization in time, and quantization of values of the control-input, allows a systematic search to be conducted over the possible sequences of control-input values. The control string, C(t), is analogous to a gene. We search for improved solutions by inserting random mutations into the control gene, and then evaluating the outcome as a whole. Mutations with favorable outcomes are retained, and further mutations are applied and evaluated recursively. This technique therefore belongs to the class of genetic algorithms.

As we attempt to approximate the continuous system with smaller force and time quanta (higher resolution), the number of terms, -and the values they may take-, grows infinite. Therefore, the search must be conducted at reduced resolution.

The best solution from a reduced resolution search will generally not be equivalent to the best solution of a search conducted at a higher, or infinite, resolution. Since, the higher resolution search space includes the lower resolution search space, if their best solutions differ, then the higher resolution solution must be better. Therefore, to approach a more optimal solution, we wish to conduct a search at the highest practical resolution.

It is desired to obtain the best solution that can be found in a given practical amount of computation time. A high resolution search consumes much computation time searching the entire space at high resolution. The maximum resolution is thus limited by the available compute time.

The required compute time is related to the number of permutations that must be examined in the search space. The number of potential permutations is:

$$(M * D)^N$$

Where M is the number of F-force magnitude quanta, D is the number of theta-direction quanta, and N is number of time samples or potential switching-points from the origin to the goal. From this, we can see that the computation time is exponentially related to the resolution. For instance, a doubling in force (magnitude or direction) resolution implies a 2^N increase. A doubling in time resolution implies a squared increase in compute time. Therefore, minimizing resolution is computationally expedient.

Note that we use the term "potential" permutations above. The actual number tends to be less-than the potential number since many potential permutations are obviated by spatial boundaries and obstacles in a depth-first search. Therefore, this is an upper-bound.

There is a time-space duality between the resolution of the switching times, and the resolution of the force magnitude/direction quantities. For instance, if the force resolution is halved while the time resolution is doubled, equivalent control trajectories can still be maintained by time averaging the now more rapid, but less precise force-switching quantizes.

In 2-D space, the minimum force resolution, without loosing any degrees of freedom in movement directions, is one magnitude quantity and four direction quantities, or two magnitude quantities and two direction quantities. In either case, the M*D product equals four (4). In the former case the magnitude would correspond to the maximum admissible force, and the directions could be 0, 90, 180, and 360-degrees. In the later case the magnitudes would correspond to (+) and (-) the maximum admissible force, and the directions could be 0 and 90-degrees. (Clearly this is just two ways of saying the same thing.)

We now describe an iterative refinement based method to achieve a solution in less time than would otherwise be required by performing a single search at the equivalent resolution level to produce a solution of similar quality. The strategy is to perform a course resolution search over all the control input space to determine the most promising region. Next a higher resolution search is performed only within this region to find a more optimal sub-region. Then continually higher resolution searches are recursively performed within the sub-region. The search complexity is roughly equal at each level.

For computational simplicity, the minimum force resolution of four (4) force magnitude/directions is used throughout the process, with increasing switch-point time resolutions. Time-resolution may later be traded for increased force-resolution, as a final integration step, once sufficient resolution has been obtained.

Construct Initial Coarse-Resolution Path:

The initial number of switch-points for the initial coarse resolution search depends upon the complexity of the feature space, the available computation time, and the search rate of the computer. Since a direct method for determining the initial number of switch-points is unknown, a variety of techniques can be used such as, experience, trial and error, and heuristics which take into account features of the obstacle space. For instance, fewer switch-points may be placed in areas which are far from obstacles and therefore require fewer course corrections.

The number of initial switch-points can impact the quality of the final solution, since the coarse resolution search determines the region of focused attention. Therefore, care should be taken to investigate the larger space before delving into a fine resolution search of a selected sub-region.

In general, the initial search should use the largest number of points possible to complete the search computation in the allotted time. This is estimated by measuring the rate at which switch-point nodes are processed by the computer. The maximum number of nodes to be processed is given by the $(M*D)^N = 4^N$ formula. It is recommended that at least half of the remaining computation time be devoted to the remaining resolution levels. For instance, a Sun Sparc-10 processes about 80,000-nodes/second. If the total allowable computation time is 60-seconds, then allowing 30-seconds for the initial search would imply:

$$N = \mathrm{Log}_4(T*R) = \mathrm{Log}_4(30*80{,}000) = 10\text{-nodes}$$

Actually, due to obstacles reducing the number of available paths, the search can be completed in much less time, which allows more time for refinement. Consequently, this estimate for N forms a conservative upper bound.

In general, there is no way of knowing what the distribution of the N switch points in time should be, nor of whether a solution can be found using N switch points. Therefore, it is useful to have an estimated upper bound on the solution cost. Then, the points can be distributed along this cost. This distribution and cost bound can be obtained through experience, trial and error, or a heuristic method.

For our navigation application, we determined the initial switch point set-size and distribution by the applying the following 2-step process:

1. Perform a Dijkstra search on the obstacle vertices, where cost is taken to be the Euclidean distance between vertices. This finds the minimum distance path from the origin to the goal, and it ignores the vehicle dynamics.
2. For each segment in the path from (1), compute the optimal switching times to move the vehicle along each segment ensuring the vehicle velocity is zero at each segment's end-points. This decouples the stages of the problem into a series of simple linear re-positioning sub-problems.

The result of this process is an initial control string, C(t), and a fixed upper bound cost.

The resulting initial pathway may not coincide with the optimal dynamic path, but it is usually a good one that is close. If it is close enough, then the refinement process can find the optimal path.

In the general case, if an intuitive heuristic approach is unknown for a given state-space, but a reasonable upper-bound cost can be estimated, then an initial control string can often be found by concatenating and evaluating N variable switching points that are distributed uniformly along the cost axis. If no solution can be found within the estimated upper bound cost with N-points, than either a higher resolution search (>N) or a higher upper-bound cost is required. If it is the former, an increase in N would exceed the allowable computation time, thus the problem would be partitioned into sub-problems which would be solved similarly but separately. The separate solutions would be combined to form the initial control string which would then be refined as described below.

Genetic Iterative Refinement or Increasing the time-resolution:

Once an initial trajectory is found, attempts are made to refine the switching points by inserting new variable switching points between the existing switching points. The switching point string, C(T), is considered to be a control-string or list of force-directions with time-deltas between them. A new switch-point B can be inserted between two elements A and C by relating their delta times as follows:

$$Tc - Ta = (Tc - Tb) + (Tb - Ta).$$

The time relationships of the remaining elements in the string remain unchanged.

The force-directions and time-deltas of the existing switch-points are held constant, while a search is performed over a newly inserted variable point, by simulating the trajectory determined by the new string. The trial is made four times, once with the new point set in each of the four switch directions. If a new trajectory reaches the goal in less time than the previous best trajectory, then the new trajectory becomes the best trajectory. At least one of the force-directions will equal the previous best trajectory, since one force-direction matches that of the previous switching-point in the string, and therefore represents no force-change.

Inserting the new switch nodes at random distances between the existing nodes seems to yield the best results. This is apparently due to the greater variation available through random insertion points.

To control the size of the trajectory change introduced by a varying-point, it is useful to insert a second switching-point, between the new varying point and the next switching-point, that resumes the force application to its value prior to the new varying point. In this way, the size of the trajectory change is controlled by the length of time the new force is applied. Inserting the second switch-point at a random distance between the new varying point and the next switch-point seems to be advantageous, especially if the distribution is weighed heavier close to the varying point.

Note that as the density of points increases in time, the time resolution is effectively increased, and the length of new force applications becomes shorter, which creates smaller trajectory variations. Therefore the search space automatically becomes more restricted as the time resolution increases. Consequently, the search complexity is maintained at a constant level by reducing the search space as the resolution increases.

It is usually beneficial to insert many new varying points into the control string at once, and then to test all their permutations together, since often multiple simultaneous trajectory changes are very beneficial to minimize the path yet avoid obstacles. Once again, the maximum number that can be inserted at any one time is N due to computational limitations.

The above process is simply iterated indefinitely, as more points are added. Points which do not change the switching direction from that of the previous point are simply removed from the control string, C(T). The result is that more optimal switching times and directions begin to emerge with greater precision.

An example switch-point string is shown below:

Time	Time-Delta	Control Force
0.0		1.0 /_ 0-degrees
0.25	0.25	1.0 /_ 90-degrees
0.5	0.25	1.0 /_ 0-degrees
0.75	0.25	1.0 /_ 90-degrees
1.0	0.25	1.0 /_ -90-degrees
1.5	0.5	1.0 /_ 90-degrees
1.75	0.25	1.0 /_ 180-degrees
2.0	0.25	1.0 /_ 0-degrees

A new switch-point is inserted at T=1.6 as follows:

	Time	Time-Delta	Control Force
	0.0		1.0 /_ 0-degrees
	0.25	0.25	1.0 /_ 90-degrees
	0.5	0.25	1.0 /_ 0-degrees
	0.75	0.25	1.0 /_ 90-degrees
	1.0	0.25	1.0 /_ -90-degrees
	1.5	0.5	1.0 /_ 90-degrees
--->	1.6	0.1	1.0 /_ 90-degrees
	1.75	0.15	1.0 /_ 180-degrees
	2.0	0.25	1.0 /_ 0-degrees

The new switch-point string can now be reduced to:

Time	Time-Delta	Control Force
0.0		1.0 /_ 0-degrees
0.25	0.25	1.0 /_ 90-degrees
0.5	0.25	1.0 /_ 0-degrees
0.75	0.25	1.0 /_ 90-degrees
1.0	0.25	1.0 /_ -90-degrees
1.6	0.6	1.0 /_ 90-degrees
1.75	0.15	1.0 /_ 180-degrees
2.0	0.25	1.0 /_ 0-degrees

Notice that the difference between the initial and final strings is that the sixth switch-point was placed with greater accuracy (ie. to within 0.1 versus 0.25 grid).

Diagram 1 shows the resulting trajectory from an arbitrary control string. Diagram 2 shows the resulting bifurcations when one new varying control point is added to the string. Diagram 3 shows the bifurcations resulting from the addition of another new varying point elsewhere in the string. Notice that there are now 16 potential trajectories. Diagram 4 shows the potential trajectories given 5 varying points. Note that 4^4 = 256 path permutations.

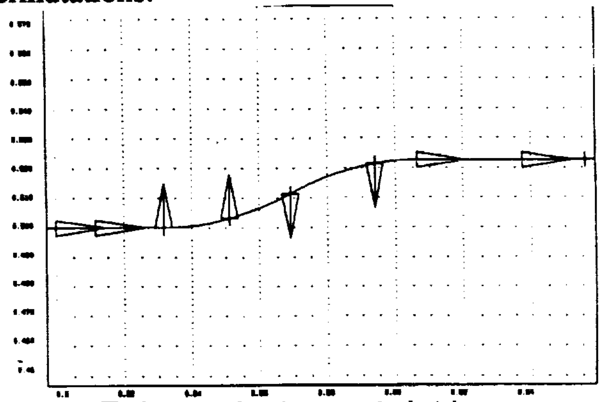
Figure 1 - Trajectory due to a control string.

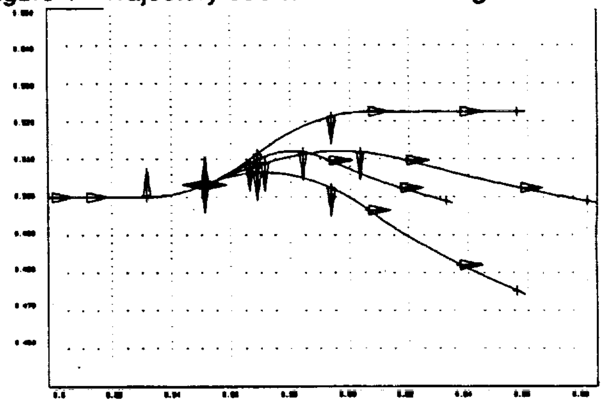
Figure 2 - Bifurcations due to 1-new varying point.

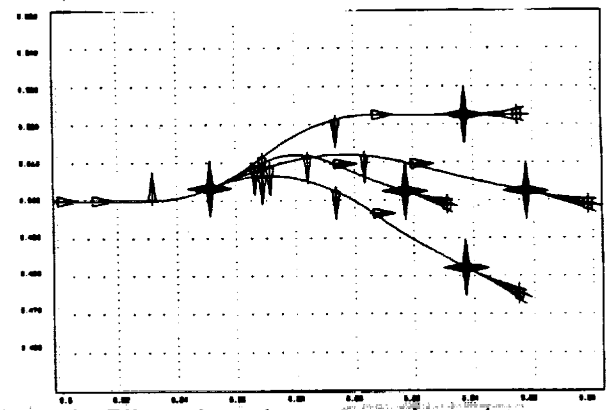
Figure 3 - Bifurcations due to 2-varying points.

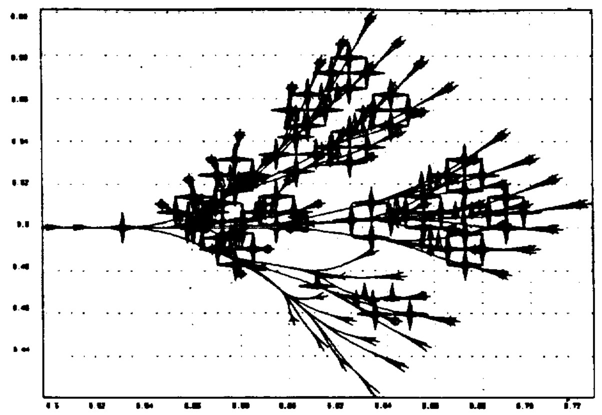

Figure 4 - Bifurcations due to 4-varying points.

Diagrams 5 through 8 show the resulting trajectories found for the respective obstacle fields. The velocity versus time graphs are also shown. The vehicle was assumed to have a 1-Kg mass, with a maximum admissible controlling force of 1-N. Diagram 9 is a typical convergence plot comparing the rate with which the solution was improved by the genetic algorithm to that of the baseline non-iterative full-resolution search. Note that the time required to reach a solution of equal quality was reduced through use of the genetic method.

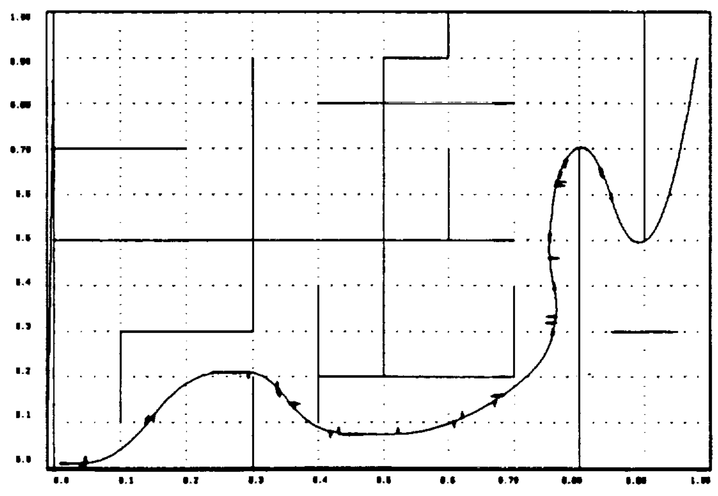

Figure 5 - Trajectory for obstacle field A in x,y.

Conclusion:

We have described a genetic iterative refinement method that is applicable for multi-stage coupled problems and that achieves a solution in less time than would otherwise be required by performing a single search at the equivalent resolution for a solution of similar quality. Further investigation will focus upon improving and characterizing the nature of the algorithm's convergence rate in terms of asymptotic, exponential, or other properties.

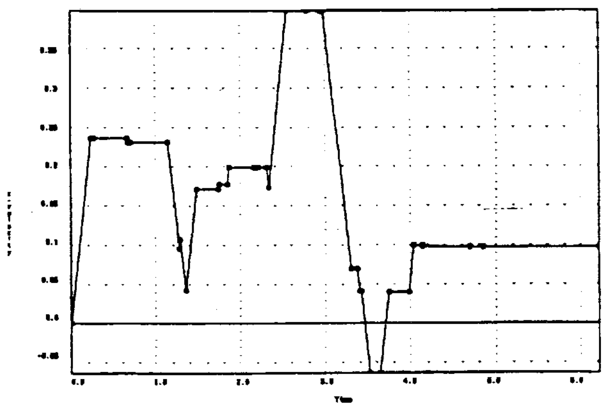

Figure 6 - X-Velocity vs. Time for obstacle field A.

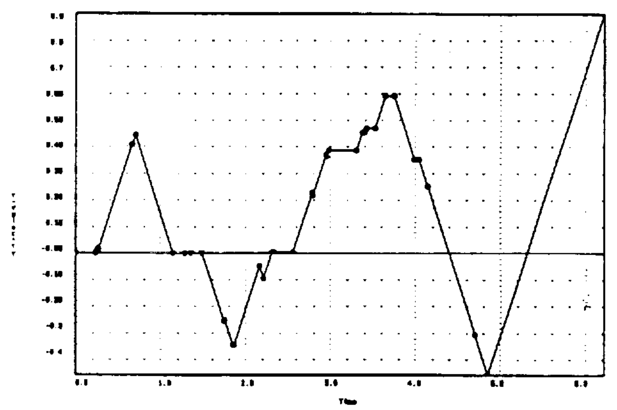

Figure 7 - Y-Velocity vs. Time for obstacle field A.

References:

[1] Anderson, B., Moore, J., "Linear Optimal Control", Prentice Hall.

[2] Leitmann, G., "Optimization Techniques with Applications to Aerospace Systems", Academic Press.

[3] Brayton, R., Spence, R., "Sensitivity and Optimization Simultaneous changes in M-components", Elsevier Scientific Pub.

[4] Kuo, B. "Digital Control Systems", Holt, Rienhart, and Winston.

[5] Slottine, Li, W., "Applied Non-linear Control", Prentice Hall, Inglewood Cliffs, NJ, 1991.

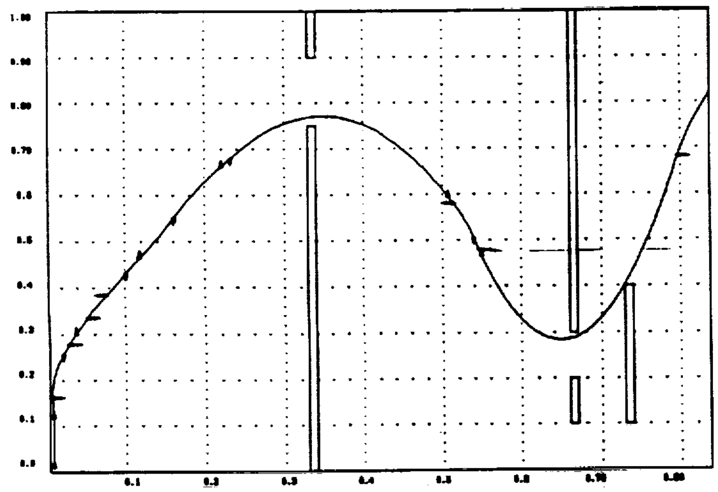

Figure 8 - Trajectory for obstacle field B in x,y.

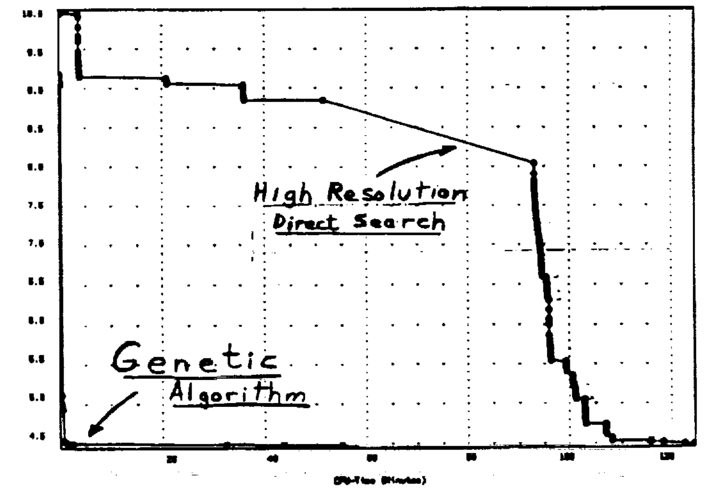

Figure 9 - Convergence rate comparison.