

A Hybrid GA-Heuristic Search Strategy

This paper by Akeel Al-Attar (PhD), the Technical Director of Attar Software, appeared in the September 1994 issue of AI EXPERT USA.

The need for a hybrid GA approach

There is currently wide interest in genetic algorithms among the AI community world-wide. This interest is driven mainly by the inadequacy of linear programming and rule based (heuristic) systems in solving complex resource planning and scheduling problems, and more generally by the desire to have an effective search algorithm which is independent of the nature of the solution domain. The GA concepts are very appealing since they appear to offer the ultimate 'free lunch' scenario of good solutions evolving without a problem solving strategy, once the requirements and constraints are defined.

However, the reality for developers attempting to put GAs into practical use can be quite different from the hype surrounding the technology. Developers are often faced with having to use a very large number of 'genes' to represent real problems and, as a result, hours or days to evolve solutions which may not be of acceptable quality. So how does one reconcile the hype and the reality. The answer is that developers take the 'free lunch' promise too literally. They expect a GA system to discover solutions from zero beginnings; i.e. the GA is expected to search for good solutions in a massive search space without being armed with even primitive common sense heuristics. The lack of such heuristics would account for the large number of genes required and the slowness of the evolution process. Over the last three years I have been involved in developing software tools and methodologies for applying GAs to complex search problems. The approach I have developed is a hybrid GA-Heuristic search strategy. The hybrid approach is reflected in both the implementation of the Genetic Algorithm and in the methodology of applying it to solve problems.

Implementing An Effective Genetic Algorithm

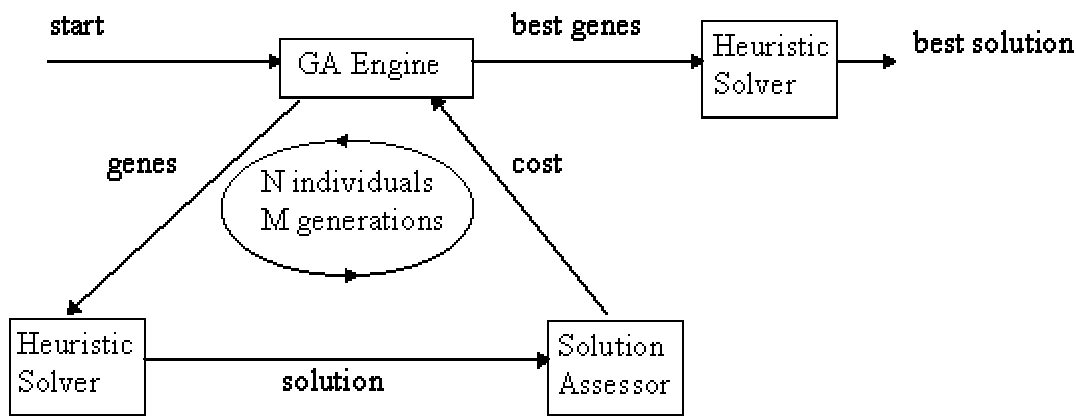
In addressing real world problems the following features are required from a genetic algorithm (beyond the standard GA functionality):

1. The developer should be able to represent the parameters of the problem as numeric variables (phenotype or feature genes). The mapping of these numeric variables into genotype genes (bits) should be carried out by the algorithm transparently to the developer. Genetic operators (crossover and mutation) function at bit level.
2. The algorithm should allow the number of genes to be a variable that is determined at runtime.
3. The algorithm should allow the genes to be structured into a number of chromosomes (groups of genes). The genetic operators (crossover and mutation) should work on each chromosome independently.
4. The algorithm should support different types of phenotype genes, numeric, index and Sequence. Numeric genes are used to represent numeric parameters, they can be real or integer numbers and the developer should be able to define the minimum and maximum allowable values for each gene. Index genes are used to represent selections from lists of options, each gene can have a defined maximum value. Sequence genes are used to return an ordered list of index genes.
5. The genetic operators (mutation and crossover) should ensure that only valid gene values are generated; i.e. numeric or index gene values falling within the minimum and maximum allowable values, and sequence genes having unique values within a chromosome. The algorithm should have a strategy for 'repairing' genes in order to meet the value constraints. Forcing the genetic operators to produce 'legal' individuals which do not violate constraints makes the evolution process a few orders of magnitude faster than it would have been if it was relying on natural selection to satisfy constraints.

6. An additional genetic operator is often needed to 'fine tune' the genetic search in the later stages of the evolution cycle. This operator we call 'adaptation' and is inspired by the principles of Lamarckian adaptation. The operator is very similar to mutation, except that it would only retain mutations which 'improve' the individual.
7. It is beneficial for the algorithm to support an optional crossover operator for sequence genes which functions at phenotype gene level.

A hybrid methodology for Applying Genetic Algorithms

The basic concept behind the hybrid methodology is not to use the GA to directly optimise the parameters of the solution, but rather to use the GA to optimise the parameters of a simple heuristic problem solving strategy. The new approach is shown in the following diagram:



I will demonstrate the application and the effectiveness of this methodology by using a simple resource planning application. In this problem we have four resources A, B, C and D each being capable of carrying out a number of different operation types O1, O2 and O3. For each resource, the time it takes to carry out an operation, the cost of running, and the available capacity are given in the following table:

O1,O2,O3 = time per unit of operation

Resource	O1	O2	O3	cost per minute (\$)	available capacity (min)
A	2	5	1	30	1000
B	-	4	2	40	4720
C	1	-	-	50	400
D	1	-	2	20	1000

Given that we have three requirements (jobs) for 1400 units of O1, 1200 units of O2 and 900 units of O3, what is the optimal resource planning that meets the following objectives:

- Cheapest production cost
- Delivering the required units of each job
- Not exceeding the available capacity for each resource

This problem is quite simple and can readily be solved using linear programming or rule based techniques, however it is useful in illustrating our GA methodology.

The obvious way of using GA's to solve the above problem is to define a number of numeric (phenotype) genes to represent the number of units of each operation assigned to each resource. The GA would then be expected to minimise a cost function which consists of the total cost of running the resources plus the total penalty points for exceeding the available capacity of resources. The relative cost of exceeding the available capacity per minute has to be set to a value which reflects the severity of the constraint. This representation of the problem needed some 30 generations of 50 individuals to evolve a solution which had a cost of 99% of the known optimal cost. This is quite slow considering the simplicity of the problem.

Following the new methodology, we used genes to represent the sequence of jobs to be presented to a simple resource planner. For this strategy to work effectively, we needed first to break down the three large jobs (requirements) into smaller jobs. The jobs were broken down as follows:

- The job of 1400 O1 was broken down into five jobs of 280 O1 each
- The job of 1200 O2 was broken down into four jobs of 300 O2 each
- The job of 900 O3 was broken down into three jobs of 300 O2 each

The initial three requirements are broken down into 12 jobs in total. We can now use 12 sequence genes to represent the order in which these jobs are presented to a simple planner. The simple planner uses very simple heuristics; namely take the next job in the queue and allocate to the first resource (from A to D) which has available capacity. This new gene representation strategy needed only two generations of 50 individuals to evolve a solution with a cost of 100% of the known optimal cost. The evolution time was reduced from minutes in the initial strategy to seconds using the new methodology.

The methodology described above was used to deliver a very innovative application to Channel 4, the largest commercial TV station in the UK. Channel 4 had the requirement of optimising the sequencing of advertisements within a commercial break. They have to ensure that different copies (versions) of the same advert go out simultaneously to different regions of the UK whilst satisfying as many of advertisers requests for special positions as possible. Advertisers can specify a mix of regions, first or last position in a break or a 'top & tail' pair in the same break. A break sequencer has to ensure that all the requirements are met without overlaps or gaps in the break transmitted to each region. In a four minute break across six regions there are 144 ten second slots to schedule with up to 50 adverts. A heuristic knowledge based system solution to the problem was deemed too difficult to achieve and a GA optimisation approach was considered.

The direct way of solving the break scheduling problem is to use 144 index genes to represent each of the 10 second slots across the six regions. Each index gene would return a pointer to an advert (one of 50). This strategy could take hours to optimise. Given that there are over 70 breaks a day to optimise, the evolution time is unacceptably high. However by applying our methodology, we were able to use sequence genes to represent the sequence of adverts to be scheduled (up to 50) by a simple heuristic scheduler. This approach reduced the evolution time to seconds. The simple scheduler works by taking the next advert in the sequence and placing it in the first available position (across the required regions) in the break. The optimisation task becomes one of optimising the sequence in which the adverts are presented to the simple scheduler. A major advantage of using this methodology is that a simple scheduler would always produce a valid schedule which satisfies the hard (lethal) constraints while the soft constraints are represented in the cost function and are satisfied through natural selection.

Our experience with using GA's to develop resource planning and scheduling applications, is that as the complexity of the problem increases so does the number of genes needed. For complex problems, this would result in very long evolution times and may even require more genes than can be handled by the algorithm. Our strategy is that as the complexity of the application increases, we would keep the number of genes under control at the expense of increasing the complexity of the 'simple' heuristic schedule/planner

that is used in conjunction with the GA. Typical increase in complexity involves defining additional index genes to select resources from lists sorted according to domain heuristics.

Applying the Hybrid Optimisation Strategy to Learning Rules From Data

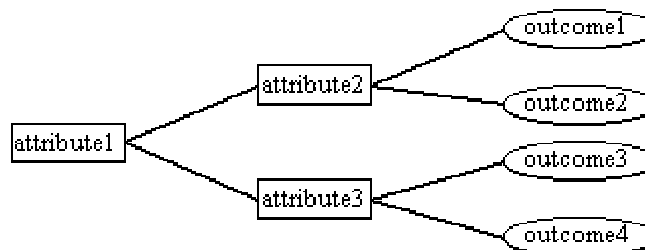
One area in which GA's can potentially offer an effective search strategy is that of symbolic learning of rules and patterns from data. Symbolic learning is a supervised learning paradigm which can derive rules from data. Such rules relate an outcome of interest to a number of attributes. These rules are typically of the format:

IF attribute1 = a AND attribute2 < b THEN outcome = 1 (Probability = .9)

Where attribute1 and attribute2 are discrete and numeric attributes respectively.

Rules can be used to understand relationships and patterns in a data file or as a data model to predict future outcomes given attribute values. The problem facing a symbolic learning algorithm is that of searching billions of possible rules for generic rules that can accurately predict the outcomes of a test data set. Several attempts have been made to use GA's to search for rules in data. Whilst some of these attempts have been successful, the evolution times required to derive a small number of rules (less than 6) from even small data sets (several hundred records) can typically be hours. The standard approach is to use a long string of genes to represent the set of rules. A maximum number of conditions is assumed, typically four or so. A condition involving a numeric attribute would need one gene to define the attribute, a second gene to define a comparator and a third gene for an attribute value. Conditions involving discrete attributes are more demanding needing one gene to define the attribute and a number of genes representing the discrete values in the condition. Assuming an average of four genes per condition and four conditions per rule, 16 genes are required to define a rule. It follows that even searching for 6 rules would require about 100 genes and therefore a lengthy evolution time. The data set is normally split between a training and a test sample. The outcomes of the rules represented by the genes are derived by scanning the training data set. The cost function to be minimised during evolution represents the number of test data records wrongly classified by the rules.

The above gene representation does not exploit any heuristic strategies which accounts for the lengthy evolution time. On the other hand, symbolic learning algorithms, such as ID3, are based completely on heuristics. Following our methodology of a hybrid GA-heuristic approach we applied this to the symbolic learning problem. Before explaining the way in which this was achieved, let's first consider how an ID3 type of algorithm works. ID3 builds a symbolic model in the form of a classification tree, an example of which is shown below:



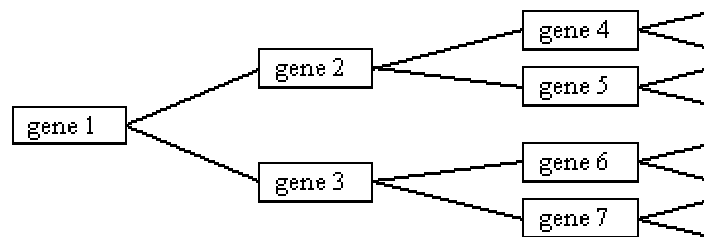
Note that each path from the root of the tree to a leaf (outcome point) represents one rule of the type discussed above. The branching strategy used in the above tree is called binary whereby a numeric attribute would have a threshold branch value, and a discrete attribute would have its values split between the two branches.

ID3 uses entropy as a heuristic measure to determine the best attribute to use at every node in the tree. Entropy is also used as a heuristic to determine the threshold value or the grouping of the attribute values between the branches at a node. When applied to learning from real data, classical ID3 is complemented by a heuristic tree pruning strategy to prevent the tree growing to fit noise and irrelevant

attributes in the data. Statistical significance of a new node, or the errors introduced by adding a new node are often used as pruning heuristic. The limitations of ID3 with pruning extensions are:

- The entropy is a local search heuristic which, although gives good results, may not be globally optimal.
- The size of the tree, and therefore the number of rules, can be small if only a small data set is used (few hundred records). This is because repeated branching reduces the number of data records feeding to each leaf. This reduces the number of useful rules that can be derived and the accuracy of the resulting data model.
- A tree normally only uses a sub-set of the attributes available. This is particularly true with small data sets or when attributes are correlated. This can make the data model unstable and highly dependent on the correlation of the attributes holding in future data.

The basic concept behind a hybrid GA approach is to use GAs to evolve multiple classification trees. The genes are used to define the attributes used in the nodes of the tree as shown below for a tree 3 attributes deep:



While the attributes at every node in the tree are defined by genes, entropy heuristics are used to define the numeric threshold or the grouping of discrete values at every branch. Pruning heuristics are also used to prune the tree defined by genes. This is a similar approach to the resource planning problem discussed above. The genes represents a set of attributes or parameters which are used to drive a simple heuristic tree builder. The hybrid approach is very efficient on genes, it uses N-1 genes to generate (potentially before pruning) N rules. By using sets of genes, we can evolve multiple trees simultaneously. The efficiency of the gene representation and the use of heuristics reduces the evolution time by a few orders of magnitude.

This hybrid symbolic learning approach was applied to Loans data supplied by a Bank. Some 430 data records were available. This was split 50:50 between a test and training test. The resulting training set (215) is on the small side for ID3 with pruning extensions which induced a five leaf tree (5 rules) with a classification accuracy of 73%. Only three attributes out of thirteen were used by the tree. In contrast, from the same data, the hybrid GA system evolved three trees which had a total of 11 leafs (rules), a classification accuracy of 78% and used 6 of the available attributes. The evolution time was only a few minutes on a 33MHz 486.

Conclusions

In this article, I have discussed a practical approach to harnessing the potential of GA's by combining their search power with heuristic knowledge that exists in an application domain. I hope that I have demonstrated that even simple domain heuristics can have a major effect on the performance of a GA in a particular domain. We at Attar Software have helped corporate clients develop complex systems using this methodology. These systems would have been almost impossible to develop using either a pure heuristic rule based approach, or a pure GA optimisation approach.

Dr Akeel Al-Attar, England, July 1994.

The user case stories from Channel 4 TV and United Distillers show practical implementations of this technology.