Transactions on Information and Communications Technologies vol 8, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

The kinematic synthesis of path generating mechanisms using genetic algorithms

A.M. Connor, S.S. Douglas, M.J. Gilmartin
Mechanisms & Machines Research Group, Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, UK

Abstract

This paper presents a methodology for the synthesis of path generating mechanisms using Genetic Algorithms (GAs). GAs are a novel search and optimisation technique inspired by the principles of natural evolution and 'survival of the fittest'. The problem used to illustrate the use of GAs in this way is the synthesis of a four bar mechanism to provide a desired output path.

1.0 Introduction

A *mechanism* is a device which transforms a given input motion to provide a desirable output motion. The study of mechanisms is used to understand the relationships between geometry, motions and the forces that provide those motions and provides the theoretical foundation for machine design.

Mechanisms can be divided into distinct groups depending upon the purpose for which they have been designed. *Function generating* mechanisms are designed so that a given output member rotates, oscillates or reciprocates according to a specified function of time. *Path generating* mechanisms are designed so that a given coupler point traces out the path of a desired trajectory. *Motion generating*, or *body guidance*, mechanisms are designed to transfer a rigid object from place to place whilst maintaining a given orientation.

The design of mechanisms, of all types, involves two distinct phases. *Kinematic synthesis* is concerned with the action of the mechanism without consideration of the forces involved. Forces, torques and other kinetostatic properties are calculated in the *dynamic analysis* of the mechanism. Kinematic synthesis is the process of designing a mechanism of the correct configuration and geometry to provide the desired output motion. In general, synthesis can be divided into two stages. *Type* synthesis is the selection of the mechanism configuration, which is usually based on the experience of the machine designer. A selection is made by considering the input motion and the desired output

motion and how it is best to transform one into the other.

Dimensional synthesis is essentially the process of choosing the geometry and link lengths of the mechanism so that the output motion exactly matches the specified motion. Traditionally, dimensional synthesis was carried out using graphical techniques where the motion could be specified at up to five precision points. However, dimensional synthesis can be achieved using numerical optimisation methods, which have no limit on the number of precision points. Despite this, machine designers are reluctant to use these methods as they often require complex definitions of synthesis requirements and lack the robustness needed for an efficient synthesis technique.

The method [1] presented in this paper utilises a GA to synthesise a mechanism to generate a desired trajectory specified by twelve precision points. GAs have several advantages over gradient based or 'point to point' search methods. Gradient methods are very effective on smooth, uni-modal functions, but on multi-modal or noisy functions they become trapped in local sub-optimal solutions. GAs avoid this by using a population of solutions. For each iteration, individuals from the population reproduce with each other to ensure that desirable characteristics are passed on to the child solutions. By modelling real genetic operators, and using a 'survival of the fittest' approach, the method will tend to converge towards globally optimum solutions without the use of penalty functions, or other techniques, required to ensure that 'point to point' methods remain in the desired solution space.

2.0 Genetic Algorithms

GAs differ from most other search methods in many ways, and these differences are what make them as robust and as widely applicable as they are. Several differences may be identified;

- GAs work with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not a single point.
- GAs use objective function information directly, not secondary information such as gradients.
- GAs utilise probabilistic transition rules, not deterministic rules.

The aim of this section is to give a brief introduction to GAs and try to answer the questions 'how' and 'why' they work. In a GA the parameter set of the problem is represented as a binary string, called *chromosomes*, where sub-sections of the string represent each parameter. These subsections are known as *genes*. Each gene may take a variety of values or *alleles*.

In natural genetics, genes from the chromosomes of two parents are recombined through a variety of genetic operators to form the chromosomes of the children. These operators are crossover and mutation. However, there is also an implicit operator of selection, or choice of mate. The simple GA used in this study models all three of these operators.

2.1 Reproduction (Selection)

Reproduction is the selection of parents from a generation for the creation of children for the next generation of solutions. In the GA used in this work, selection was carried out using a 'roulette wheel' method [2]. This is a randomised approach, where the probability of selection is weighted by the objective function value of each solution. The effect of this is that the most fit solutions are more likely to propagated into the successive generation. This embodies the 'survival of the fittest' concept.

2.2 Crossover

Crossover occurs between two parental chromosomes to form two children. In accordance with the randomised nature of GAs, there is a probability of crossover occurring between two parents. If crossover does not occur, the two parent strings are resubstituted into the new population. If crossover does occur, a random crossing site is chosen and data is transferred between strings around this point. As an example, consider the two parent strings below;

1 2 3 4 5 6 7 8	Position
1 1 1 1 1 1 1 1	Parent 1
0 0 0 0 0 0 0 0	Parent 2

If crossover occurs between positions 4 & 5, the two child strings produced are;

1 2 3 4 5 6 7 8	Position
1 1 1 1 0 0 0 0	Child 1
0 0 0 0 1 1 1 1	Child 2

2.3 Mutation

Crossover is the main process by which data is transferred between successive generations. However, mutation also plays an important role in the convergence of a population to a globally optimum solution. A GA utilising crossover alone can only converge to a solution represented by the *schemata* (see section 2.4) of the initial population. The effect of mutation is to broaden the search space by creating new schemata, thus forcing the search to explore new hyperplanes. The mutation operator is very simple. Each bit of a child string is tested against the probability of mutation. In a binary alphabet, if mutation occurs, the bit undergoes Boolean inversion. That is, a one becomes a zero and vice-versa. Because the effect of mutation in natural systems is less than that of crossover, the probability of mutation occurring is normally quite low.

2.4 The Schema Theorem

The previous sections have outlined the mechanics of how a GA works, but do not answer the question of 'why' a GA works. The answer to this is found in Holland's schema theorem [2,3] and the principle of implicit parallelism.

One of the major reasons for using a binary representation for the GA coding of the parameter set is that similarities between highly fit solutions

become more apparent. A *schema* (pl.schemata) or similarity template describes a subset of solutions at certain positions. For example, in five bit binary coding a typical schema could be;

1 0 # # #

where the # symbol represents an irrelevant value. The order of a schema, denoted by o(H) is simply the number of fixed positions. The defining length, denoted by $\delta(H)$ is the distance between the first and last fixed positions. The schema theorem is often described in terms of a growth equation;

$$m(H,t+1) \geq m(H,t) \times \left(\frac{f(H)}{\bar{f}} \right) \times \left[1 - P_c \times \left(\frac{\delta(H)}{(L-1)} \right) - o(H) P_m \right] \qquad \ldots(1)$$

where P_c and P_m are the crossover and mutation probabilities, f(H) is the 'fitness' of schema H and $\bar{f}$ is the average schema fitness. The term m(H,t) represents the number of examples of schema H at time t.

The growth equation shows that highly fit, low order schema with short defining lengths are rapidly propagated through the population. In real terms, this implies that the population will tend to progress or 'evolve' towards highly fit solutions.

Whilst the schema theorem explains why a GA works, the principle of implicit parallelism explains why they work so effectively. This effectiveness derives from the simultaneous allocation of search effort to different regions of the solution space. Essentially, a schema may be viewed as a representation of a hyperplane of the solution space. By testing a single schema the GA is effectively testing all solutions on that hyperplane.

3.0 Mechanism Synthesis and Four-Bar Mechanisms

Figure 1 is a schematic diagram of a four bar mechanism. Each 'stick' represents a link between revolute joints. This study was limited to an important class of mechanism where the input link rotates with constant velocity, and throughout the cycle the path generating point traces out a closed curve and the output link oscillates through a fixed angle. This type of mechanism is known as a crank-rocker, and occurs when the shortest link is the crank and lies adjacent to the fixed frame. This is known as the Grashof mobility criterion and is expressed by the inequality $l + s < p + q$, where l & s are the longest and shortest lengths and p & q are the intermediate lengths.

The four bar mechanism is a single degree of freedom mechanism and as such requires only a single input to completely specify the motion of the mechanism. It is a very useful mechanism as the constant velocity input provides desirable motion transmission characteristics.

The coupler curve is the locus of point P as the input link rotates through one complete revolution. In mechanism design, the shaping of the coupler curve

to satisfy design needs is regularly enountered and therefore an automatic method which deterimines the mechanism proportions for a specified coupler curve is of great use.

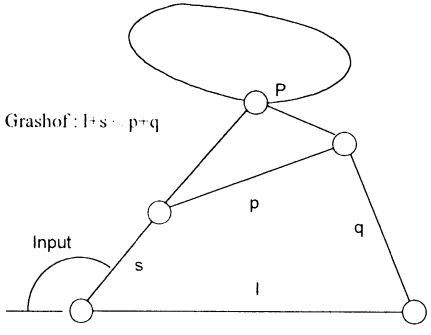

Figure 1 : Four Bar Mechanism

3.1 The Objective Function

The objective function is evaluated by calculating the error at the path generating point between the actual position and the desired position, for a given input angle. This is illustrated in Figure 2, where (X_d,Y_d) are the co-ordinates of the desired position and (X_a,Y_a) are the co-ordinates of the actual position.

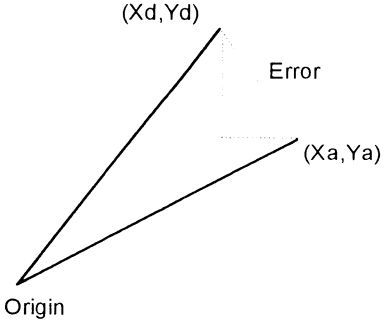

Figure 2 : Calculation of Error

The total error around the cycle of the mechanism is calculated and used as the 'fitness' score for the mechanism. The objective function includes a heuristic filter as a penalty function for solutions that fail the Grashof mobility criterion. Any solutions which fail this criterion are automatically assigned the worst possible objective function value. The GA is used to find a mechanism which has the smallest possible error between the actual curve it generates and the desired design curve.

4.0 Results

The GA was tested on two coupler curve problems which were generated by known mechanisms. Figure 3 and Figure 4 are graphs which show typical performances of the GA on each problem. The graphs plot best of generation fitness against generation number. The GA was run for a total of one hundred generations.

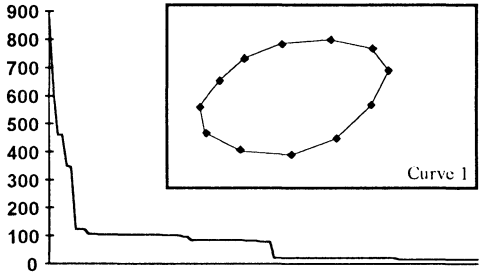

Figure 3 : Graph of Convergence for Test Curve 1

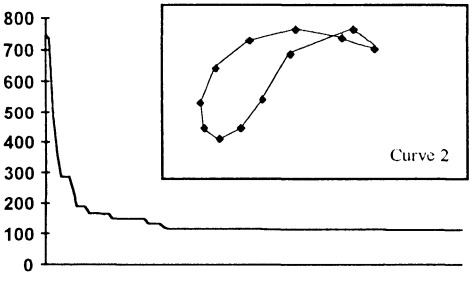

Figure 4 : Graph of Convergence for Test Curve 2

For the first test coupler curve the error between the curve generated by the final trial mechanism and the desired curve is 15.96 units. When the parameters of this mechanism were passed into a simple hill climbing search, the error dropped to a value of 0.40 units, indicating that the GA had located the optimal peak on the objective function surface and the coupler curve generated by the trial mechanism is identical to the desired curve.

The second coupler curve is more complex as it contains a 'double point'. The shape of the curve resembled a 'figure of eight'. After 100 generations the error of the trial mechanisms is 113.79 units. After the hill climbing search was performed, the error is still 109.79 units. The GA has located a sub-optimal peak. The coupler curve generated by this trial mechanism is shown in Figure 5 along with a diagram of the mechanism. It can be seen that this curve does not resemble the desired curve as it does not contain a 'double point'.

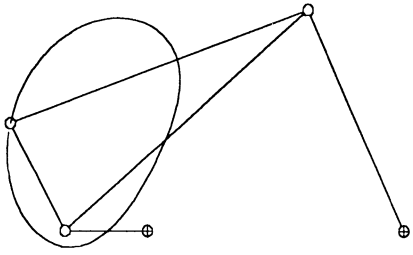

Figure 5 : Trial Mechanism and Generated Coupler Curve

5.0 Discussion

For all of the synthesis problems the controlling parameters of the GA were set at the following values;

Probability of crossover = 0.75
Probability of mutation = 0.04
Population size = 80

Whilst Goldberg [2] and DeJong [4] have suggested a crossover rate of 0.6 and a mutation rate that is inversely proportional to the population size, Grefenstette [5] has shown that a higher crossover rate is beneficial for improved performance. The relatively high mutation rate was chosen to offset any biases brought into the GA by the rejection of solutions that fail to meet the Grashof mobility criterion.

The simple GA has performed quite well on the first coupler curve and has located the global optimum solution. An examination of the population distribution at the end of the run showed that the entire population of valid solutions had started to converge to a solution in the region of the global optimum. However, for the more complex second curve the GA could only locate a sub-optimal solution. This difference in performance may be attributed to the fact that the objective function is actually defined by the specification of the desired coupler curve. Each of the problems, therefore, has a different objective function. By examining the curve shown in Figure 5, it is possible to see that the use of only twelve precision points is making it difficult for the GA to differentiate between curves with double points and normal closed curves.

For the second curve, the GA is experiencing premature convergence. This may be due to the poor definition of the coupler curve or may be due to biases in the GA. To ensure that the GA performs well on a variety of synthesis problems, the nature of the GA must be changed to prevent this. There are a variety of methods which can be introduced to this end. This include sharing [2], crowding [4] and a variety of other breeding schemes.

There are several other methods which can also be used. These include deflation and identical string elimination [6] and incest prevention [7]. Further work will investigate the use of these methods and illustrate any improvements

in effectiveness. It is also important to investigate the possibility of biases in the GA from either the selection method or the 'heuristic filter'. One alternative to this filter is to use a mobility penalty function, which penalises the objective function by a varying amount depending on how far 'away' the trial mechanism is from the Grashof criterion.

6.0 Conclusions

The results of this study have shown that GAs may be successfully applied to the problem of mechanism synthesis, but further work is required to ensure that the GA is effective across a wide range of synthesis problems. The results have also shown that the GA is not a particularly quick optimisation method and that difficulties where encountered due to poor problem definition and constraint violations. However, adequate performance can be achieved by combining the GA with a local search method. It may be possible to use the GA, in conjunction with local search methods, to develop a robust 'double pass' synthesis technique.

Once an effective GA is developed for use in the synthesis of four bar mechanisms, it can also be applied to a variety of other mechanism synthesis problems. These include the synthesis of multi-degree of freedom mechanisms with both constant velocity inputs and programmable servo motor inputs.

Future papers will deal with the refinement of the GA, the synthesis of multi-degree of freedom mechanisms using an objective function based on structural error only and the development of more complex 'multiple objective' functions.

7.0 References

1. Connor, A.M., The Use of Genetic Algorithms in Optimisation, M.Sc. Dissertation, Liverpool John Moores University, 1994.
2. Goldberg, D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley, 1989.
3. Holland, J.H., Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.
4. DeJong, K.A., An Analysis of the Behaviour of a Class of Genetic Adaptive Systems, Ph.D. Thesis, University of Michigan, 1975.
5. Grefenstette, J.J., Optimisation of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, Man and Cybernetics, 1988, SMC-16/1, pp122-128.
6. Pham, D.T. & Yang,Y., Optimisation of Multi-Modal Discrete Functions Using Genetic Algorithms, Proceedings of the Institution of Mechanical Engineers (Part D), 1993, vol 207, pp 53-59.
7. Eshelman, L.J. & Schaffer, J.D., Preventing Premature Convergence by Preventing Incest, pp 115-122, Proceedings of the Fourth International Conference on Genetic Algorithms, 1991.