AIAA JOURNAL
Vol. 33, No. 11, November 1995

Variable-Complexity Genetic Algorithm for Topological Design

P. J. Gage,* I. M. Kroo,† and I. P. Sobieski*
Stanford University, Stanford, California 94305-4035

Genetic algorithms are known to be appropriate for optimization in nonsmooth domains. A variable-complexity algorithm has been developed for use in topological design problems. The system starts with simple descriptions and adds degrees of freedom to allow development of more complex designs as optimization proceeds. The algorithm is applied in two different domains: design of low-speed nonplanar wings to minimize induced drag or total drag and design of structural trusses to minimize weight. In each domain, special constraint-handling schemes are introduced to improve performance. Calculus-based optimization in smooth subspaces is also employed to improve efficiency. These successful applications demonstrate that the variable-complexity algorithm is an important new tool for topological design.

Introduction

GENETIC algorithms have been applied to a wide range of optimization problems and have been shown to be effective in multimodal and nonsmooth domains.[1–4] Calculus-based methods are ineffective for these tasks because they locate only a local minimum and require domains with continuous first derivatives for gradient calculation. A standard genetic algorithm optimizes the values of a fixed set of parameters, but for many optimization tasks the best parameter set is not known a priori. Hence, an algorithm that can vary the parameter set during optimization has been developed. This variable-complexity algorithm is particularly appropriate for design studies where it is common to start with a simple representation and progress to more detailed descriptions that use more design variables.

Although genetic algorithms can be applied to a wide range of problems, domain-specific implementation details are often critical to their success. These algorithms rely on the existence of building blocks from which the global solution can be assembled, and the topology of the design space influences how readily these building blocks can be identified. Constraint-handling schemes affect the topology of the design space, so that it is important to design them carefully. The use of calculus-based optimizers in smooth subspaces can help when tight convergence in the smooth regions is essential for correct identification of the global optimum.

A brief description of genetic algorithms is provided in the next section of this paper, with particular attention given to the unusual features of the variable-complexity algorithm. The application of this algorithm to the aerodynamic design of nonplanar wings and the structural design of trusses is then described. Evaluation of these applications leads to conclusions regarding the general utility of this optimization method.

Genetic Optimization

Genetic algorithms are designed to mimic evolutionary selection.[1] A population of candidate designs is evaluated at each iteration, and the candidates compete to contribute to the production of new designs. Each individual is represented by a string, which is a coded listing of the values of the design variables. The entire string is analogous to a chromosome, with genes for the different features (or variables). When individuals are selected to be parents for offspring designs, their genetic strings are recombined in a crossover operation, so that the new designs have elements of two earlier designs. A mutation operation also allows modification of elements of the new individual so that it may include new features that were not present in either parent.

Genetic algorithms do not use gradient information to guide their search. Their effectiveness depends on useful correlations between parts of the genetic string (genotype) and the performance of the individual it represents (phenotype). Substrings, or building blocks, which appear in the description of above-average phenotypes are likely to survive into the next generation, even if the genotype is broken up by the action of crossover and mutation. Short, low-order building blocks are retained and combined to form higher order building blocks, with the process repeating over many generations until the best design is found. Promising features of different candidates can be recombined to produce improvements in complete designs.

Efforts should be made to ensure that the design space is shaped to allow ready identification of building blocks. At the simplest level, the arrangement of variables in the string can affect the likelihood of building blocks being disrupted by crossover.[5] When optimal values of different variables are closely linked, as is the case for wing sweep and wing thickness to chord in compressible flow, they should lie close together in the genetic string. This arrangement reduces the likelihood of the crossover point falling between them, and the offspring will inherit values for both variables from a single parent. It is more common for the constraint-handling scheme to influence performance. The standard method for imposing constraints is to append a penalty function to the objective,[1] but it is important to grade these penalties to reflect the extent of constraint violation.[6–8] Where possible, a repair scheme should be used in preference to a penalty function.[6,9] These repairs typically involve a modification in the decoding of a string to ensure that it describes a viable candidate design. Consequently, they are domain specific, because each decoding scheme is domain specific.

A standard genetic algorithm operates on a fixed set of design variables. This limitation requires that a description of all possible elements must be explicitly included in the genetic string throughout the optimization process. A modifed genetic algorithm has been developed to permit the number of design variables to change during optimization.[6] This variation is achieved by modification of the crossover operator, so that the crossover point can be in a different position for each parent and the offspring have genetic strings of different length. This new operator is similar to that used in genetic programming.[10] It is illustrated in Fig. 1, for a wing design problem, with the dihedral of each wing element concatenated to form the genetic string. This approach allows the identification of

Presented as Paper 94-4413 at the AIAA/NASA/USAF/ISSMO 5th Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, Sept. 7–9, 1994; received Dec. 2, 1994; revision received April 26, 1995; accepted for publication April 26, 1995. Copyright © 1995 by P. J. Gage. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Research Assistant, Department of Aeronautics and Astronautics. Student Member AIAA.

†Associate Professor, Department of Aeronautics and Astronautics. Member AIAA.

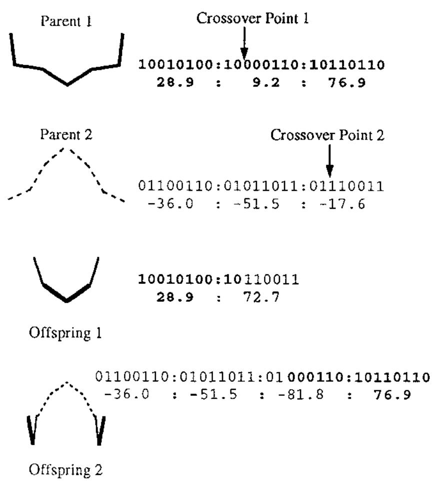

Fig. 1 Modified crossover operator.

successful features in simple designs described by few variables, which can then be included in designs of greater complexity.

One general limitation of genetic algorithms is the lack of simple and reliable convergence criteria, analogous to the Karush–Kuhn–Tucker conditions for gradient-based optimization. Genetic algorithms typically get close to an optimum point quickly but can take many generations to locate it exactly. Often it is more effective to run several cases to near optimality rather than run a single case to exact optimality. With fewer total function evaluations, several near-optimal solutions can suggest quite different designs, or elements from different candidates can be combined by the designer to produce a superior final product. In the region of an optimum, final refinement can be achieved by fixing the discontinuous variables and using a gradient-based method in the subspace that is continuous. It is also possible to create a true hybrid method, optimizing the smooth subspace at every function evaluation of the genetic algorithm. Both uses of gradient-based optimization are explored here.

The variable-complexity capability is particularly attractive for synthesis studies because it is similar to the process used by designers. In wing design, for example, parameters such as wing area and span are generally determined before the airfoil sections are chosen. The variable-complexity genetic algorithm should be suitable for design tasks in which the best solutions are complex combinations of simple elements. Two such tasks are explored in the remainder of this paper.

Topological Wing Design for Minimum Drag

The objective in the low-subsonic speed wing design tasks considered here is to minimize drag with fixed lift. Without the imposition of geometric constraints, the optimal wing has very large span and an elliptic chord distribution that produces the optimal local lift coefficient everywhere. Therefore, span and height constraints are introduced to force the wing to lie within a box, as shown in Fig. 2, so that results will be of practical interest.

Candidate wing geometries are represented by a number of planar lifting elements. The design variables include element dihedral, span, and chord. Each element may have a span of between 1 and 8 panels, each of unit span. Each panel is aerodynamically modeled by a horseshoe vortex, with the bound vortices oriented at the correct dihedral and the trailing vortices lying in the freestream direction. The discrete values for element span (which is an integer number of panels) make the problem intractable for calculus-based optimizers. An additional variable, the number of elements, is available when a variable-complexity representation is used. Between 1 and 10 elements can be used to describe a wing, in contrast with the standard algorithm which uses 10 elements for all wings. All design variables are indicated in Fig. 2. This figure also shows the hierarchy of wing components, with panels grouped in elements, which are combined to form complete designs.

Performance is evaluated using the MULTOP program,[11] in which a system of linear equations representing the relation between vortex strength, lift, induced drag, and parasite drag is constructed. Solution of this system yields the vortex strengths that produce minimum

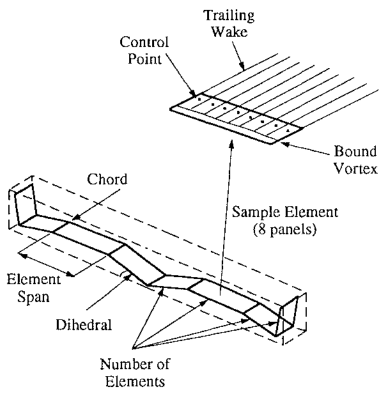

Fig. 2 Wing topology design.

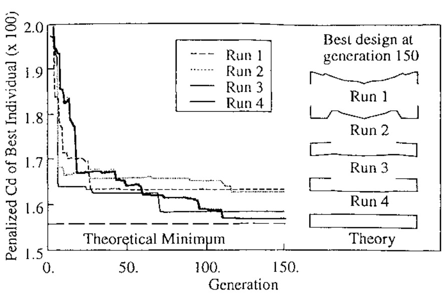

Fig. 3 Optimization histories.

total drag subject to the constraint of fixed lift. (This automatic satisfaction of the lift constraint is an attractive feature of the solution method: earlier efforts using a vortex-lattice method required the use of two incidence variables for each element, and a repair scheme to satisfy the lift constraint.[6]) Geometric constraints are enforced by the genetic algorithm through quadratic exterior penalties.

Minimum Induced Drag

The best geometry for minimum induced drag given fixed lift is known to be a box wing.[12] Hence, for purposes of comparison, only induced drag is considered in the initial investigation of optimizer performance. For the chosen height-to-span ratio of 0.1, the induced drag for the optimal wing is 21% lower than the drag for an elliptically loaded planar wing. Chord does not affect induced drag and so it is removed from the set of design variables for this case.

Rather than concentrating resources in a single long run, computer time is split between several shorter runs. Results for four different randomly generated starting populations are included. The best designs produced after 150 generations, from an initial population of 500 candidates, are shown in Fig. 3. The final geometries are quite different in each case, although performance is quite similar (and always far superior to the performance of the best planar wing). The first two runs produce good designs that differ markedly from the true optimum. These indicate design opportunities that might be preferred for reasons not modeled in the optimization problem, reasons that might be missed by focusing too early on global optimality.

In the last two runs, the final result closely approaches the theoretically optimal box shape. Only very slight potential for performance improvement remains, and the selection pressure to further extend the upper wing is very low. Although the exact optimum is not located, the nature of the best solution is clearly identified.

A standard genetic algorithm achieves results similar to those for the first two runs. All wings are described using 10 elements, each using between 1 and 8 panels. The average design in the randomly generated initial population has a total span of 40 panel widths (10 elements with average span of 4 panel widths). The geometric constraints limit total span to 20 panel widths and height to 4 panel widths. These constraints are satisfied by using elements of low span and moderate dihedral, so that total span is reduced and the wing folds to fit within the specified box. Any subsequent attempt to reduce drag by extending the wing can only be achieved by increasing the span of an existing element, and this is likely to produce constraint violation.

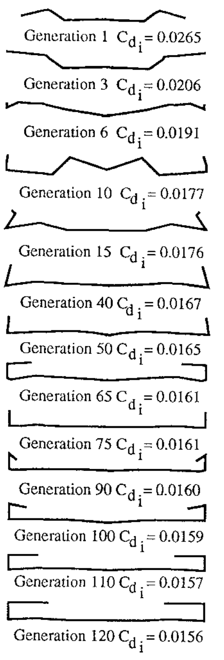
Generation 1 C_{d_i} = 0.0265
Generation 3 C_{d_i} = 0.0206
Generation 6 C_{d_i} = 0.0191
Generation 10 C_{d_i} = 0.0177
Generation 15 C_{d_i} = 0.0176
Generation 40 C_{d_i} = 0.0167
Generation 50 C_{d_i} = 0.0165
Generation 65 C_{d_i} = 0.0161
Generation 75 C_{d_i} = 0.0161
Generation 90 C_{d_i} = 0.0160
Generation 100 C_{d_i} = 0.0159
Generation 110 C_{d_i} = 0.0157
Generation 120 C_{d_i} = 0.0156

Fig. 4 History of best individual in population.

For the variable-complexity algorithm, wings may be described using 1–10 elements. The average wing in the initial population has only 5 elements, with an average element span of 4 panel widths and, hence, a total span of 20 panel widths. It is much easier to satisfy the geometric constraints and, consequently, there is more freedom to work on drag minimization. When these wings are refined, it is possible to add new elements and discover more complex designs. This process is illustrated in Fig. 4, which gives a history of the best individual design in the population at various stages in the optimization run.

In the original population, the best design has reasonably high span and is relatively flat. Almost immediately, wings of the maximum span are developed, and vertical elements are present at the tips. The main wing is then refined to be closer to horizontal, and the first nearly vertical winglet extensions are seen. Finally, when the main wing and winglet are close to fully refined, further improvement is achieved by increasing the span of the horizontal winglet extensions.

These general trends are also reflected in the whole population, as shown by the superposition of the right half-wings of the entire population, in Fig. 5. In the first generation, which is randomly generated, there are designs with extreme dihedral but no individual with the maximum possible span. By generation 50, almost all designs lie within the box and have large span. At generation 100, most main wings are close to flat, there are many vertical winglets and a couple of nearly horizontal winglet extensions. In the final generation, the main wings are even closer to horizontal, and many more C wings are present.

There is significant diversity in the population even after 150 generations. The premature convergence that often limits the performance of standard genetic algorithms is not evident here. This is due to the novel crossover operation, which can produce new designs as offspring even if both parents are identical, by acting at a different location in the genetic string of each parent.

Minimum Total Drag

Consideration of parasite drag complicates the optimization task for nonplanar topologies. The best load distribution for induced drag is not optimal for total drag. The total vertical load is constant (fixed total lift) but the best distribution of vertical load depends on the loading of nonplanar elements. Load on these nonplanar elements can reduce the induced drag of the planar section by redistributing the load in that region but at the cost of increasing parasite drag locally. Nonplanar elements are expected to be smaller and carry less load as parasite drag becomes an increasingly important component of total drag, but a theoretical solution for the optimal topology

Table 1 Levels of parasite drag

Case	C_{d_0}	C_{d_1}	C_{d_2}
1	0.002	0	0.002
2	0.005	0	0.005

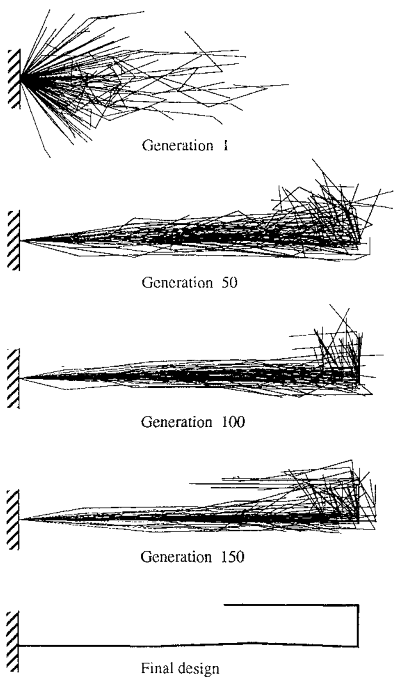

Fig. 5 Superposition of population members; front view of right half-wings.

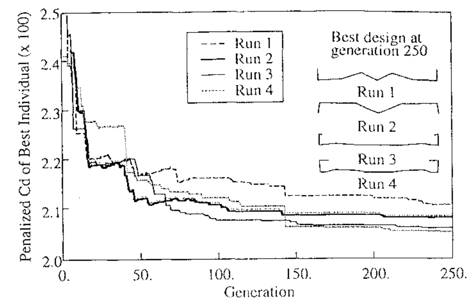

Fig. 6 Optimization histories, case 1.

is not available. The parasite drag coefficient is assumed to vary quadratically with section lift coefficient C_l,

$$C_{d_p} = C_{d_0} + C_{d_1} C_l + C_{d_2} C_l^2$$

The relative importance of parasite drag can be modified by varying the coefficients C_{d_0}, C_{d_1}, and C_{d_2}. The zero parasite drag case has already been discussed. In this section, two further cases are considered, as indicated in Table 1.

The genetic algorithm now uses three variables to describe each wing element, with chord being introduced to reflect the influence of element area on parasite drag. Population size is increased to 800 and number of generations to 250 to handle this larger design space.

The convergence histories and elevations of final designs for case 1, shown in Fig. 6, are similar to those for the case of zero parasite drag. A significant difference can be observed in Fig. 7, which shows that the chord of the winglet extension is very small. For the prescribed total lift coefficient of 1.0, parasite drag is only 20% of the total, but the benefit of the extensions for induced drag is already almost completely offset by their contribution to parasite drag. The combined area of the winglet and extension is 9.6% of the planar area.

Figure 7 also indicates that the dihedral and chord distributions of the best design at generation 250 are not precisely optimal. In this case, fixing the topology and performing gradient-based optimiza-

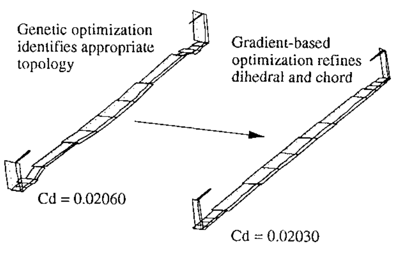
Genetic optimization identifies appropriate topology
Gradient-based optimization refines dihedral and chord
Cd = 0.02060
Cd = 0.02030

Fig. 7 Final refinement by gradient-based optimizer.

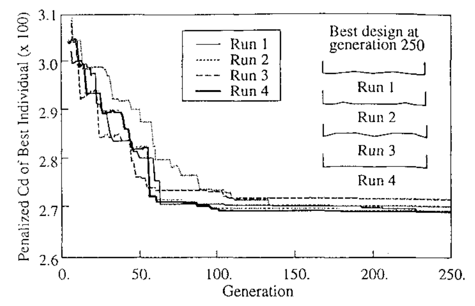

Fig. 8 Optimization histories, case 2.

tion of dihedral and chord yields a 1.5% improvement in total drag. A hybrid method, which would optimize the smooth subspace at every function evaluation of the genetic algorithm, is not used here because it increases the cost of each function evaluation by three orders of magnitude. It is more efficient to repeatedly run the genetic algorithm alone, to generate several near-optimal designs. The gradient-based method is used only to refine the dihedral and chord distributions of these final topologies to establish a true optimum.

The runs for case 2, shown in Fig. 8, indicate that a single solution becomes clearly optimal as the relative importance of parasite drag increases. Horizontal winglet extensions are not beneficial, but the winglets are all of maximum height (10% of span). Winglet area is 6.6% of planar area, so that nonplanar area is, indeed, reduced as the importance of parasite drag increases. With parasite drag contributing 40% of total drag, the genetic algorithm gets very close to the correct chord and dihedral distribution. Final refinement with a gradient-based optimizer produces less than 0.25% improvement in total drag.

These results indicate that the optimal topology is sensitive to the level of parasite drag. The genetic algorithm is able to successfully identify near-optimal wing shapes for several parasite drag levels. Gradient-based methods can be used for final refinement but are not needed when parasite drag is responsible for a large percentage of the total drag.

Topological Truss Design for Minimum Weight

In the wing design problems just considered, the complexity of the wings was modified by changing their overall size, and this was usually achieved by adding extra elements at the tip of the wing. Complexity can also increase by splitting existing elements into smaller pieces, so that the genetic string carries more detailed information about a design of similar total size. A structural example of this mode of complexity increase is desirable because it also serves to illustrate that the algorithm is not discipline dependent. There is extensive literature on topological design of trusses and so results produced by the variable-complexity genetic algorithm in this domain can be readily compared with the designs found by other methods.

A plane truss structure is typically described by a set of nodes connected by structural members loaded only in tension or compression. Some nodes are prescribed to have zero displacement whereas others are acted upon by specified external loads. An optimum truss has minimum total weight, while satisfying stress and buckling constraints on the members and displacement constraints on the nodes.

The design variables used in truss optimization may include properties of the members and locations of the nodes. There are three kinds of standard truss design problems. At the simplest level, a sizing problem varies only the cross-sectional areas of a fixed number of members. A shape problem may add variables describing the location of the nodes. A topological problem must also allow trusses which use different sets of members.

The earliest application of genetic techniques to truss design was performed by Goldberg and Samtani.[13] They optimized the areas of a 10-member truss, with stress constraints for each member applied as quadratic exterior penalty functions. Their solutions on three separate runs, each performing 6400 function evaluations and starting from a different randomly generated initial population, were within 2% of the optimum attained by a gradient method. Although the genetic algorithm was effective for this sizing task, it was not efficient, and the gradient method is really more appropriate.

More recently, Sakamoto and Oda[14] introduced a hybrid technique for truss design with a genetic algorithm used for layout design and a simple gradient method used for sizing the cross-sectional areas. They found that the hybrid method had greater practical reliability than a member elimination strategy, where they defined practical reliability as the likelihood of finding a design with performance within 20% of the global optimum. The member elimination strategy often became stuck at local optima that were unnecessarily complex.

The genetic string used by Sakamoto and Oda[14] for topological design is a concatenation of single bits, each representing the existence or absence of a possible element. Several other researchers have used similar schemes for genetic solution of structural optimization tasks.[15–18] In these schemes the string length grows very rapidly as the number of nodes (and hence the number of possible elements) is increased. The population size required to avoid a deceptive sample of the design space can become prohibitively large.

The variable-complexity genetic algorithm provides an alternative approach for topological design. The genetic string need only carry information about members that actually appear in the design. They are described as general truss members with particular endpoints, rather than particular truss members with known endpoints. This approach is more flexible and efficient for designs where exhaustive representation of all possible members becomes unwieldy.

Candidate designs generated by a genetic algorithm may be incomplete structures or mechanisms. Sakamoto and Oda[14] identified these candidates and gave them fitness equal to the minimum in the population. This makes it difficult to identify useful building blocks, because it creates large regions with no gradation in fitness.[6–9] It is preferable to avoid description of mechanisms by employing an encoding scheme that generates only feasible candidates. Such a scheme is clearly biased, because it precludes description of many instances in the feature space. However, Mitchell[19] claims that biases that include factual knowledge of the domain (that candidates should not be mechanisms) and biases that favor simplicity are useful aids to the learning process.

In the encoding scheme chosen for truss design tasks, illustrated in Fig. 9, candidate solutions start from a baseline design, and the

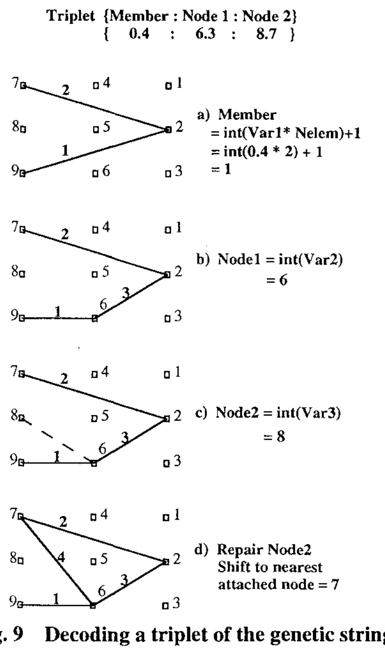

Fig. 9 Decoding a triplet of the genetic string.

genetic string describes modifications to be made. The baseline is a nonmechanism that uses the minimum possible number of members to attach all loaded nodes. Where several alternatives use the same number of members, the description with minimum total length of members is selected.

The genetic string is a concatenation of sets of three variables, with each triplet describing a potential modification of an existing element. The first variable identifies the existing member to be modified. The second identifies the node to which the endpoints of the old member are to be connected. The third variable identifies a second node, and a new member is introduced between the two described nodes. This member must ensure that the new structure is not a mechanism, and so the second node must already be connected to the structure. A repair scheme shifts it to the nearest attached point if necessary. Complexity increases as the length of the genetic string is increased, because each triplet splits an existing member into two and adds an additional member.

When the topology of the candidate structure has been determined by decoding the string, a gradient-based optimizer is used to size the members. In a hybrid scheme, the cost of each function evaluation for the genetic algorithm depends directly on the time spent on optimization in the smooth subspace. The work required for sizing increases linearly with the number of members (quadratically if finite differences are used to produce gradient information). The variable-complexity algorithm typically generates candidates with a small number of members, so that the time spent sizing them is greatly reduced.

Two applications of this genetic scheme to truss design tasks are reported here. The first provides direct comparison with the hybrid method employed by Sakamoto and Oda.[14] The second allows comparison with a theoretically determined optimum for a Michell[20] truss.

Nine Nodes, Two Load Points

The first optimization task is to find a minimum weight truss to support two loads of 1000 N each, with a vertical deflection of 0.015 mm at the load points. A 3 × 3 grid of nodes is provided, with horizontal spacing of 100 mm and vertical spacing of 50 mm (Fig. 10). The number of possible elements is 36 (for n nodes, the number of possible elements is $n[n-1]/2$), and Young's Modulus is prescribed to be 200 GPa. The density used to calculate truss weight is 0.0079 g/mm^3.

A population size of 100 is used, and the genetic algorithm is allowed to run for 30 generations. Figure 11 indicates that the true optimum is found within 2500 function evaluations in each of five trials. The hybrid method of Sakamoto and Oda has difficulty in this problem, finding a practical optimum in only 80% of trials. They

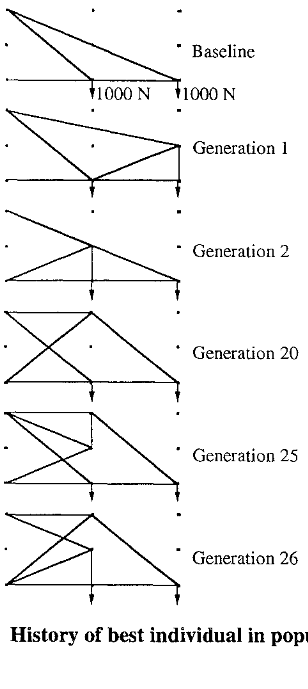

Fig. 10 History of best individual in population.

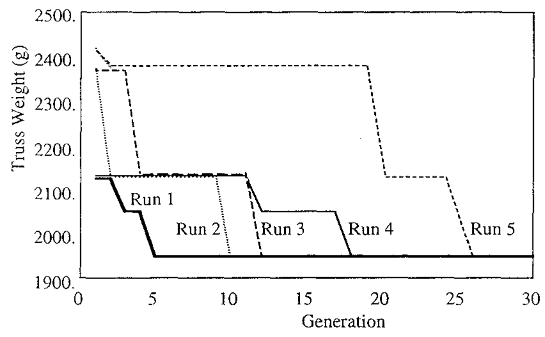

Fig. 11 Optimization histories for 9-node truss with two load points.

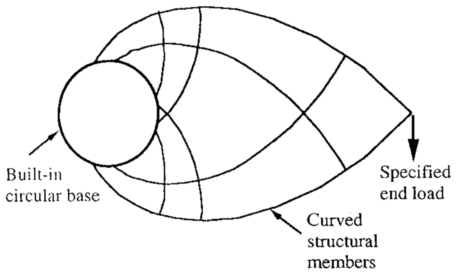

Fig. 12 Michell truss for single end load.

are essentially reporting a success rate for finding a feasible design, whereas the encoding used here reduces the search space by guaranteeing feasibility. The average candidate design in their starting population has 18 members (half the bits in an average random string of length 36 will be 1). Consequently, each function evaluation for their genetic algorithm is more expensive, because there are more members to be sized. The variable-complexity algorithm provides superior performance at lower computational expense.

Figure 10, a history for the best population member for a sequence of generations shows the variety of designs encountered during optimization. The trusses up to generation 20 are built with a single modification of the baseline structure, whereas the trusses in later generations are built using two triplets in the genetic string. The optimum combines the modifications of generation 2 and generation 20. The variable-complexity algorithm combines building blocks from simple parents to produce more complex offspring with higher performance.

Michell Truss

For a single vertical end load, which produces a force and a couple resisted by a supporting circle, the optimal truss topology is known. First described by Michell,[20] the members lie along lines of principal strain, forming a series of spirals that intersect orthogonally (Fig. 12). The optimal members are curved, although straight bars are used in practice.

The optimal solution can be approached by providing a grid of reasonable density, so that several straight bars approximate the theoretical spirals. A 7 × 5 grid of nodes is used in this example, as shown in Fig. 13. The base circle is approximated by prescribing zero displacement for 3 nodes. (The base circle is represented in Fig. 13 for completeness, but it is not directly modeled in the analysis).

The variable-complexity algorithm starts with a baseline structure that has two members, connecting the upper and lower supports to the loadpoint. Stress constraints are now imposed on all members, instead of displacement constraints on the load point, to allow direct comparison with Michell's theoretical solution.

The outer shape of the truss exerts a strong influence on the total weight of the truss, and a history of the best member in the population, displayed in Fig. 13, shows that trusses of maximum height are quickly found. Details of the internal structure are not completely determined when the algorithm is stopped after 150 generations. The final designs from five different runs are shown in Fig. 14. The result from run 5 is a simple, near-optimal truss that uses only 10 bars. It is observed during every run. The first four runs produce designs that include up to four additional elements that improve

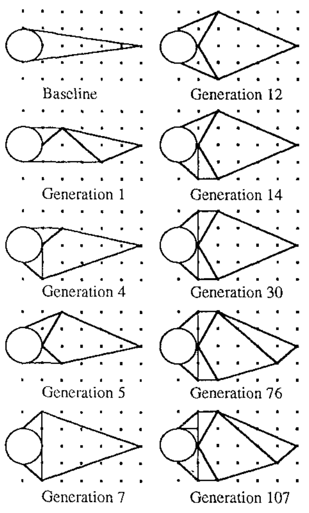

Fig. 13 History of best individual in population.

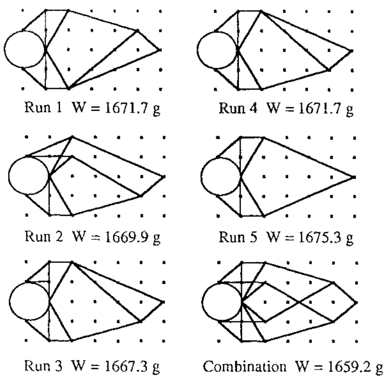

Fig. 14 Final designs from different runs.

performance by up to 0.5%. No single design produced by the genetic algorithm includes all of the features that appear in different runs. The designer is able to combine these features to seek superior performance. The last truss shown in Fig. 14 is generated in this fashion, and it yields a further 0.5% performance improvement. It uses 18 bars, and closely approximates the shape of the curved optimal truss. Its weight is within 15% of Michell's solution.

A problem of this magnitude has not been solved using a standard encoding with a fixed length genetic algorithm. With 595 bits required to represent all possible members, a population of several thousand candidates would be needed to provide an adequate sample of the search space. Average candidates in the first generation would include 297 bars, an order of magnitude more complex than anything considered by the variable-complexity algorithm in the entire optimization procedure. The algorithm described here is far better suited to tasks of genuine engineering interest.

Conclusions

The variable-complexity genetic algorithm typically moves from simple designs to more complex topologies, a progression that is familiar to human designers. (The opposite trend is common in strategies that require exhaustive description of all possible design components prior to the start of optimization.) The increase in complexity may arise from addition of extra components or from more detailed description of existing components. Applications in aerodynamics and structures illustrate both modes of refinement.

Genetic algorithms are slow to converge to exact optima. Solutions generated by the variable-complexity algorithm are often slightly simpler than the true optimum, but they capture its essential nature. Results of genetic optimization are most useful when several near-optimal designs are produced. The designer can combine features from different designs, or may prefer a slightly suboptimal design due to considerations not modeled in the problem description. Gradient-based optimizers can be used in conjunction with a genetic algorithm, to achieve tighter convergence in smooth subspaces.

The encoding language used by a genetic algorithm can strongly influence its performance. Biases that reflect factual knowledge of the domain can prevent description of infeasible designs and, thereby, restrict the search space to manageable size. Encodings of variable length, which are available to the variable-complexity algorithm, allow a bias toward simplicity. This generally reduces the cost of each function evaluation.

The variable-complexity algorithm described in this paper provides a flexible and efficient method for performing topological design. The success achieved in two different domains suggests that it will be an important new tool for multidisciplinary investigations.

Acknowledgments

This work was funded in part by NASA Langley Research Center under Grant NAG 1-1494. Their support is gratefully acknowledged.

References

[1]Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading, MA, 1989.

[2]Greene, D. P., and Smith, S. F., "A Genetic System for Learning Models of Consumer Choice," *Proceedings of the Second International Conference on Genetic Algorithms*, edited by J. J. Grefenstette, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987, pp. 217–223.

[3]De Jong, K. A., and Spears, W. M., "Using Genetic Algorithms to Solve NP-Complete Problems," *Proceedings of the Third International Conference on Genetic Algorithms*, edited by J. D. Schaffer, Morgan Kaufmann, San Mateo, CA, 1989, pp. 124–132.

[4]Caldwell, C., and Johnston, V. S., "Tracking a Criminal Suspect Through Face-Space with a Genetic Algorithm," *Proceedings of the Fourth International Conference on Genetic Algorithms*, edited by R. Belew and L. Booker, Morgan Kaufmann, San Mateo, CA, 1991, pp. 416–421.

[5]Liepins, G., and Vose, M., "Deceptiveness and Genetic Algorithm Dynamics," *Foundations of Genetic Algorithms*, edited by G. Rawlins, Morgan Kaufmann, San Mateo, CA, 1991, pp. 36–50.

[6]Gage, P., and Kroo, I., "A Role for Genetic Algorithms in a Preliminary Design Environment," AIAA Paper 93-3933, Aug. 1993.

[7]Gage, P., Braun, R., and Kroo, I., "Interplanetary Trajectory Optimization Using a Genetic Algorithm," AIAA Paper 94-3773, Aug. 1994.

[8]Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M., "Some Guidelines for Genetic Algorithms with Penalty Functions," *Proceedings of the Third International Conference on Genetic Algorithms*, edited by J. D. Schaffer, Morgan Kaufmann, San Mateo, CA, 1989, pp. 191–197.

[9]Michalewicz, Z., and Janikow, C., "Handling Constraints in Genetic Algorithms," *Proceedings of the Fourth International Conference on Genetic Algorithms*, edited by R. Belew and L. Booker, Morgan Kaufmann, San Mateo, CA, 1991, pp. 151–157.

[10]Koza, J. R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[11]Kroo, I., "A General Approach to Multiple Lifting Surface Design and Analysis," AIAA-84-2507, Oct. 1984.

[12]von Kármán, T., and Burgers, J., "Airfoils and Airfoil Systems of Finite Span," *Aerodynamic Theory*, edited by W. F. Durand, Vol. II, Springer, Berlin, 1935, Div. E, Chap. IV, Sec. 17.

[13]Goldberg, D., and Samtani, M., "Engineering Optimization Via Genetic Algorithm," *Electronic Computation: Proceedings of the Ninth Conference on Electronic Computation*, edited by K. W. Will, American Society of Civil Engineers, New York, 1986, pp. 471–482.

[14]Sakamoto, J., and Oda, J., "A Technique of Optimal Layout Design for Truss Structures Using Genetic Algorithm," AIAA Paper 93-1582, April 1993.

[15]Grierson, D., and Pak, W., "Optimal Sizing, Geometrical and Topological Design Using a Genetic Algorithm," *Structural Optimization*, Vol. 6, No. 3, 1993, pp. 151–159.

[16]Grierson, D., and Pak, W., "Discrete Optimal Design Using a Genetic Algorithm," *Topology Design of Structures*, edited by M. P. Bendsoe and C. A. Mota Soares, Kluwer Academic, Norwell, MA, 1993, pp. 89–102.

[17]Koumousis, V., "Layout and Sizing Design of Civil Engineering Structures in Accordance with the Eurocodes," *Topology Design of Structures*, edited by M. P. Bendsoe and C. A. Mota Soares, Kluwer Academic, Norwell, MA, 1993, pp. 103–116.

[18]Hajela, P., Lee, E., and Lin, C.-Y., "Genetic Algorithms in Structural Topology Optimization," *Topology Design of Structures*, edited by M. P. Bendsoe and C. A. Mota Soares, Kluwer Academic, Norwell, MA, 1993, pp. 117–134.

[19]Mitchell, T. M., "The Need for Biases in Learning Generalizations," *Readings in Machine Learning*, edited by J. W. Shavlik and T. G. Dietterich, Morgan Kaufmann, San Mateo, CA, 1990, pp. 184–191.

[20]Michell, A. G. M., "The Limits of Economy of Material in Frame-Structures," *Philosophical Magazine*, Vol. 6, No. 8, 1904, p. 589.