USING A GENETIC ALGORITHM IN DESIGNING HIGH-VELOCITY DRAINAGE CHANNELS

Clarence O. E. Burg[1]

ABSTRACT

Designing storm water management systems using high-velocity channels is a complex optimization process, with current techniques using scale models and computer simulations in a trial-and-error approach. Gradient-based optimization techniques are currently being investigated but may require alterations to the existing computational flow solvers. In this paper, a genetic algorithm technique is applied to an existing simulation (HIVEL2D) and produces good design improvements, using approximately the same amount of computation time as gradient-based algorithms without the need for any modifications to the flow solver.

Introduction

As urban centers have grown in recent years, redesign of many public facilities, such as transportation and sewage systems, has been necessary. Because of the increase in roads and buildings, less precipitation is absorbed by the ground, and the increased run-off reaches the storm water management systems more quickly. These hydrological changes serve to increase the system design load, which tends to reduce public safety. Thus, these systems must be modified to handle the increased load. Engineers use elaborate scale models and computationally intensive simulations to evaluate various channel designs. By analyzing these designs and using their knowledge of hydraulics, engineers modify the design and run new simulations; these changes should result in better designs. Clearly, this process is not deterministic and succeeds only because of the ingenuity of the designers.

In order to develop a deterministic design methodology, the fitness or effectiveness of a particular design must be measured. Also, the parameters that determine the design of the channel must be chosen. Using a computational simulation code, the derivatives of the fitness function with respect to the various design variables are calculated. Then, the design variables are modified, using this information, with the goal of obtaining the most effective channel design. Some difficulties with gradient-based approaches to design optimization are the errors and the computational costs involved in a finite difference approximation to the gradient, the necessary modifications of the computer simulation code in order to calculate these gradients efficiently, the convergence and stability characteristics of the optimization algorithm and the difficulties arising from convergence to a sub-optimal local minimum.

[1]Graduate Student, Computational Engineering, Mississippi State University.

Since the second derivative matrix, the Hessian matrix, can not be calculated accurately or efficiently, the method of steepest descent is commonly used to update the design variables. This method only demonstrates linear convergence and often uses an additional simulation in a linear search step. Furthermore, this design process is inherently sequential, requiring the evaluation of the fitness of a design and the determination of the gradient, before the flow for the next design can be calculated.

As a non-deterministic search algorithm, the genetic algorithm approach uses probabilistic techniques to locate regions of the search space that contain highly fit designs. A code based on genetic algorithms does not require the computation of derivatives or the effective use of these derivatives, and the search process is highly parallelizable, because multiple flow simulations for various sets of design variables can be processed at any given time. In this paper, the parallel nature of the genetic algorithm is not exploited.

The basic concept of a genetic algorithm, based on the work of Holland[1] and Goldberg[2], is to evolve a population of design parameters using the biological concept of survival of the fittest. As in nature, the genetic operators are selection of parents, crossover and mutation of the genetic material and replacement into the population. Since each member's fitness value can be calculated independently, this evaluation process can be conducted in parallel. Another advantage of this process is that the existing computer simulation codes can be easily adapted to a genetic algorithm interface.

For the genetic algorithm used in this paper, the members were ranked from best fit to least fit and randomly selected as parents based on their ranking. The genetic material was a string of real numbers, which were the values of the design variables. Uniform crossover was used, with the values of the design variables in the children based upon those in the parents via a random distribution. Mutation of individual design variables also used a random distribution. These two children were evaluated and placed into the population, removing the two least fit members. These operators were designed so that the population would move quite rapidly in the direction of the best fit members.

The computational fluid simulation code, HIVEL2D[3], used the finite element method to solve the two-dimensional, depth-averaged, shallow water equations. This code was used to generate the steady-state solutions for the sets of design parameters generated by the genetic algorithm. The fitness of a particular design was determined from the steady-state solution.

Three examples of high-velocity channel contractions are presented. In the first case, the wall of the channel contraction was defined by a polynomial which passed through three points. The fitness function was a measure of the difference between the flow depths for the current design and the flow depths for the target design. Hence, the optimal design was already known. The second test case used the same channel contraction design, with three design variables, and had the goal of obtaining uniform downstream flow. In the third test case, the wall of the channel contraction was given by a Bezier curve, defined by ten design variables. As in the second test case, the fitness function was a measure of the non-uniformity of flow and would be minimized.

In all three cases, the genetic algorithm identified a region of the search space that contained much improved designs. The best set for the three design variable test case yielded a 92.5% improvement over a straight wall contraction. For the 10 design variable test case, the best set produced a 99.7% improvement over a straight wall contraction.

Genetic Operators

Because of the computational cost for evaluating a particular member of the population, the genetic algorithm was designed to converge quickly. By converging quickly, the population will lose some of its genetic diversity and not be able to search the whole design space, making the algorithm more prone to finding a sub-optimal, local minimum. In particular, the selection/replacement procedures were designed to put pressure on the population to converge. The outline for the genetic algorithm is

```
Initialize, Evaluate and Rank the Population of size POP
Output Statistics for Population
Do i=1, GEN
      do j=1, POP/2
            Choose 2 Parents
            Generate 2 Children
            Evaluate Children
            Insert Children into Population
            Rank Population
      end
      Output Statistics
end
```

The number of evaluations for one run of the genetic algorithm is POP*(GEN+1). Since the population is ranked each time two children are added to the population, no special process occurs at the end of a generation; rather, the term generation for this genetic algorithm is used for comparison purposes against other selection/replacement techniques that rank the population only at the end of a generation.

Selection and Replacement. As the optimal solution was the minimum of a fitness function, the lower the fitness score for a member, the more desirable the member of the population was. Prior to the selection process, the population was ranked from lowest fitness value to highest. Each member was given an evaluation value, based on their ranking and the parameter, rank size, which determined the number of members that received the same evaluation value. For instance, if the population size was 20 and the rank size was 4, then the best 4 members received an evaluation value of 5, the next 4 members received an evaluation value of 4, with this process continuing to the worst members receiving a value of 1. The advantages of this ranking and evaluation process is that the best members are given a much greater chance of reproducing, hence placing pressure on the population to move towards these members. An observed difficulty is that if the genetic algorithm finds a set of design variables that produces a fitness value substantially better than the other values, the genetic algorithm will drift towards this member, losing diversity, and thus probably not identifying the global minimum.

Once the members were given an evaluation value, two parents were selected via the roulette wheel technique where the chance of a particular member begin chosen was based on its evaluation value. The genes of these two parents were combined via crossover and were mutated, if instructed to do so. The parents remained in the population, and the two children replaced the worst two members of the population. This replacement process guaranteed

that the best members stayed in the population and that the population's average fitness value would almost always improve.

Crossover. The genes of each member of the population were real numbers, instead of integers, with each gene representing a design variable. Because there are only a few genes in each member, uniform crossover was used, so that each gene was modified during the crossover process. The value of each gene in the children was determined by the values of each gene in the two parents via a random process. Given two random numbers, $rand_1$ and $rand_2$, between 0 and 1, and the two values for a gene from the two parents, $parent_1$ and $parent_2$, the crossover functions for generating the gene values for the two children are

$$\begin{aligned} child_1 &= parent_1 + \frac{|parent_1 - parent_2|}{2} |2rand_1 - 1|^{\gamma} sign(2rand_1 - 1) \\ child_2 &= parent_2 + \frac{|parent_1 - parent_2|}{2} |2rand_2 - 1|^{\gamma} sign(2rand_2 - 1) \end{aligned} \tag{1}$$

These crossover functions pick a value randomly in the range that is centered on the value of a parent and whose length is the distance between the two parents. The exponent γ determines the distribution of random numbers within this range. If γ is less than 1.0, then the distribution favors the values away from the parent; whereas, if γ is greater than 1.0, then the distribution favors the values near the parent. If γ is 1.0, then the distribution should be uniform.

Mutation. Creep mutation was used by centering the mutation range about the value to be mutated and choosing a new value randomly from within this range, or

$$New = Old + Range * (Random - 0.5) \tag{2}$$

The mutation rate and mutation range were parameters that could be adjusted.

The Design Problem

In the design of high-velocity channels, the main types of channel transitions are contractions, confluences of two different channels and obstacles within the flow field, such as bridge supports. For all test cases given in this paper, a channel contraction was studied, where the channel contracted from 40 ft to 20 ft over a distance of 100 ft. The flow into the channel was travelling at 28.375 ft/sec with a depth of 1.0 ft, or a Froude number of 5.0.

A channel contraction for supercritical flow often produces an oblique hydraulic jump in which the water level jumps dramatically. This change in water level propagates down the channel and may result in flooding if the water level over tops the channel wall. By adjusting the shape of the wall in the region of channel contraction, the waves produced by the contraction can hopefully be made to negate each other, resulting in nearly uniform downstream flow.

Flow Simulation. To simulate the flow in the channel, a computational flow solver HIVEL2D[3] was used. This code used a Petrov-Galerkin finite element approach to solve the two-dimensional, depth-averaged, viscous, shallow water equations on unstructured grids. By writing a structured grid generation program based a set of design parameters and using the steady-state output of HIVEL2D in the fitness function, a genetic algorithm version of HIVEL2D has been developed.

Occasionally, a set of design parameters defines a grid that produces unstable results in HIVEL2D, such as channels whose walls change too rapidly. These invalid cases were disregarded, and new design variables were chosen via the aforementioned genetic operators. Also, a highly converged steady-state solution is not necessary. Rather, an approximate value for the fitness function is satisfactory. When the fitness function stabilizes and does not change by more than 0.5% for 4 iterations, the simulation terminates early, and this approximate value is used as the value of the fitness function. Because of this early termination, each simulation for the genetic algorithm requires about 50% of the number of iterations as required for a gradient-based algorithm.

Test Case 1. For the first test case, a three design variable problem was chosen where the wall of the channel contraction smoothly matches the walls before and after the contraction and passes through the three design variables via a polynomial fit as shown in Figure #1. The grid was 9 by 45, with 5 columns before the contraction, 20 columns in the contraction and 20 columns after the contraction.

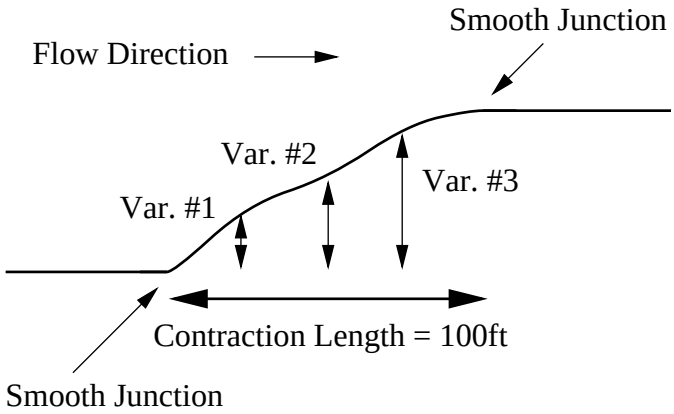

Figure #1. Location of the Three Design Variables on the Contraction Wall.

The flow was generated for a design set of (2.0, 5.0, 7.0), and the flow depths across the channel at a location downstream of the contraction were recorded. These 9 depths determined the fitness function, which was

$$F(\beta_1, \beta_2, \beta_3) = \sum_{node=262}^{270} (target_{node} - actual_{node})^2 \tag{3}$$

The global minimum for this objective function is zero, which occurs for a design set of (2.0, 5.0, 7.0).

Test Case 2. The same three design variable channel contraction problem was used in the second test case, with the goal of producing uniform flow downstream of the contraction. Uniform flow means the depth and velocities are the same at every location in the flow field. Hence, a function that measures the non-uniformity of the flow might sum the squares of the differences of the depths between adjacent locations, or

$$F(\beta_1, \beta_2, \beta_3) = \sum_{column=31}^{40} \left(\sum_{row=2}^{9} (depth(column, row) - depth(column, row-1))^2 \right) \tag{4}$$

which measures the difference in the depths of adjacent nodes along 10 columns across the channel, downstream of the contraction. For a straight wall channel contraction of these dimensions and for this grid (9 by 45), the fitness function is 1.215. As the best shape for a channel contraction is not known, this value was used to determine the effectiveness of the genetic algorithm in locating a good set of design variables. This test case was used in order to compare the efficiency of a genetic algorithm with a gradient-based search routine. The gradient-based search routine used the adjoint variable formulation of direct differentiation[4] to approximate the derivatives and the method of steepest descent with one linear search step to update the design variables.

Test Case 2a. After analyzing the results of Test Case 2 for grid independence, it became obvious that the grid spacing was too large. Thus, a smaller grid spacing was used, where there were 21 nodes across the channel instead of 9. The grid spacing along the channel was adjusted accordingly. The objective function was also modified to measure the flow non-uniformity over the same region of the channel. For this grid spacing, a straight wall channel contraction gave a function value of 2.2345.

Test Case 3. For the final test case, a ten design variable channel contraction problem was chosen, with the goal of obtaining uniform downstream flow. As in test case 2a, the refined grid with 21 nodes across the channel was used. The wall of the channel contraction is determined by a Bezier curve of 14 control points. The first two control points are 0.0 and the last two are 10.0, with the interior 10 control points to be determined. A Bezier curve is defined as

$$b(t) = \sum_{i=0}^{n} B_i^n(t) p_i \tag{5}$$

where t ranges from 0 to 1, p_i are the bezier points and B_i^n are the nth degree Bernstein polynomials defined as

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i \tag{6}$$

In this case, the 51 grid points that define the wall of the channel contraction are located at $(t * 100, b(t))$ for $\{t_i = \frac{i}{50} | i = 0, 50\}$. Since the goal is uniform downstream flow, the fitness function used for test case 2a was also used for this test case.

Results

For the first two test cases, there were 20 members in the population, and the genetic algorithm was run for 10 generations. The mutation rate was 30%, which meant that there was a mutation in almost every child, since there were three design variables. The mutation range was 1.0. The crossover exponent γ was 1.5, causing the distribution to favor locations near the parents. The rank size was 2, meaning that the best two members were assigned the same evaluation number. The parameters for test case 2a and test case 3 are given below.

Test Case 1. For the first test case, the optimal solution was known, (2.0, 5.0, 7.0). Table #1 shows the results for 7 different simulations and the fitness values for the best member in each population. From these results, it is clear that the genetic algorithm was able in all cases to find a solution that was close to the optimal solution. Also, for a solution to be good, variable 3 must be quite close to 7.0, variable 2 must be less close to 5.0 and variable 1 can vary widely from 2.0. Since there is just one optimal solution, the genetic algorithm works well even with a small population size and a few number of generations.

Variable 1	Variable 2	Variable 3	Fitness Value
1.9758	5.1476	7.0515	0.00054728
1.4074	5.5358	7.0946	0.00307624
1.8758	5.2348	7.0393	0.00060012
1.8524	5.2437	7.0681	0.00048117
1.9997	4.9996	6.9879	0.00025440
1.6562	5.3463	7.0653	0.00093108
1.9777	5.0322	7.0188	0.00020238

Table #1. Best Member of Population for 7 Runs.

Test Case 2. In this test case, the grid structure and chromosome structure was the same as in the first test case, but the optimal solution is not known, because the objective function was different. Table #2 shows the value for the best member in the population for 7 different executions of the genetic algorithm.

Variable 1	Variable 2	Variable 3	Fitness Value
4.526926	7.023662	9.929692	0.1199642
4.466168	7.055058	9.811562	0.1895257
4.479874	6.997278	9.90318	0.1498446
3.214928	5.745738	8.067184	0.3563117
3.169038	5.630676	7.96757	0.3636747
3.141586	5.635396	7.94228	0.3698275
3.556536	5.758932	8.35379	0.3242247

Table #2. Best Member of Population for 7 Runs.

From these executions, it appears that there may be at least two local minima for this problem, near (4.5, 7.0, 9.9) and near (3.2, 5.7, 8.0), with the first region possibly being the global minimum. For a straight wall channel contraction, the fitness value was 1.215. Thus, in

all executions, the genetic algorithm was able to find much improved solutions. By modifying the design of the genetic algorithm and adjusting the genetic algorithm parameters, it is hopeful that the genetic algorithm will be able to locate the global minimum more frequently.

For comparison, the gradient-based algorithm was run 30 times starting each simulation with a different set of design variables. Only 10 times did the gradient-based algorithm find the assumed global minimum near (4.5, 7.0, 9.9). To compare the computational costs of the two methods, a test run using the genetic algorithm had a population of 20, each member needing a steady-state simulation initially and then for 10 more generations, or 220 steady-state simulations for one execution. As noted before, the genetic algorithm could terminate early and did not require a linear search step as did the gradient-based algorithm, so each steady-state simulation for the gradient-based algorithm cost about 3 times as much as for the genetic algorithm. For the gradient-based algorithm, each test run made 40 steady-state simulations, or about the cost of 120 steady-state simulations of the genetic algorithm. When considering the observation that the gradient-based algorithm locates the optimal solution only approximately 33% of the time, the computational costs of the two methods are similar. It is expected that for cases with multiple, sub-optimal local minima, the gradient-based algorithm will succeed in locating the global minimum even less often. Further, the parallel nature of the genetic algorithm allows a parallel version of the genetic algorithm to identify the global minimum more quickly, even though the computational costs are similar.

Test Case 2a. For this test case, the population had 20 members, and there were 20 generations, needing 420 simulations. In all executions, the optimal solution was still improving slightly after 20 generations, showing that the genetic algorithm could produce better results with more generations. The crossover exponent γ was again equal to 1.5. The other parameter settings and the results of the executions are given in Table #3.

Simulation	Mutation Rate	Mutation Range	Rank Size	Fitness Value	Result
1	0.2	3.0	4	0.49875	Local
2	0.2	4.0	4	0.16760	Global
3	0.2	4.0	4	0.19244	Global
4	0.3	3.0	5	0.38487	Global
5	0.2	3.0	4	0.23571	Global
6	0.2	3.0	4	0.26204	Global
7	0.2	3.0	4	0.20767	Global
8	0.2	3.0	4	0.16690	Global
9	0.2	3.0	4	0.49502	Local
10	0.2	3.0	4	0.52765	Local
11	0.2	3.0	4	0.54686	Local

Table #3. Best Member of Population for Refined, 3-Design Variable Channel.

The objective function for the straight wall channel contraction where the grid has 21 nodes across the channel, instead of 9, was 2.2345. Again, each simulation found a good improvement over a straight wall contraction. By analyzing the design variables for the best member of each population, it can again be deduced that there is a sub-optimal, local minimum, near (3.2, 5.8, 8.0) and the probable global minimum near (4.6, 7.4, 9.9). These roughly corre-

spond to the minima found in Test Case 2. Four of the 11 simulations were trapped in the sub-optimal, local minimum, while the other 7 simulations were converging on the global minimum. Even though the genetic algorithm was able to locate the global minimum only seven out of 11 times (63.6%), it was able to find the global minimum twice as often as the gradient-based search algorithm used in Test Case 2 (10 out of 30 times, or 33.3%).

Test Case 3. For the final test case, the wall of the channel contraction was controlled by 10 design variables, with 21 nodes across the channel. The population size, number of generations and the rank size for each execution, as well as the results, are given in Table #4. For the other parameters, the mutation rate was 20%, the mutation range was 1.0, and the crossover exponent γ was 1.5, shifting the distribution to favor values near the parents. Even though there were 10 design variables, the genetic algorithm was able to find excellent solutions for each execution. These solutions were more than 96.8% better than a straight wall channel contraction, whose function value was 2.2345.

Population Size	Generations	Number of Simulations	Rank Size	Fitness Value
80	9	800	8	0.029974
40	18	760	4	0.005834
80	8	720	10	0.035675
80	8	720	10	0.031937

Table #4. Best Member of Population for 10 Design Variable Channel.

Conclusion

From this preliminary investigation into the use of genetic algorithms for designing high-velocity open channels, the results show that a genetic algorithm can find excellent improvements over the simple straight wall channel contraction. For a test case with 3 design variables, the optimal solution was over 90% better than a straight wall contraction, for both the unrefined and refined grids; for the case of 10 design variables, the optimal solution was over 99% better than a straight wall contraction. For this simple problem, the genetic algorithm worked quite well.

Computationally, the genetic algorithm obtained results similar to those found by using a gradient-based algorithm. More steady-state simulations were performed by the genetic algorithm than in a gradient-based algorithms, but since these simulations did not need to be as highly converged as for the gradient-based algorithm. Furthermore, the gradient-based algorithm must expend computation time to calculate the gradient, and if the algorithm uses a linear search, an additional simulation must be performed per design iteration. Also, an advantage of the genetic algorithm over any gradient-based algorithm is that these simulations can be performed in parallel for the genetic algorithm but must be performed sequentially for the gradient-based algorithm, allowing a good solution to be found much more quickly by using a genetic algorithm.

More work is needed to determine better population sizes and the number of generations as well as parameter values for the selection, crossover and mutation operators, as well as, to investigate other types of selection operators. Since the most time consuming portion of this genetic algorithm is the need to obtain a steady-state solution for a computational fluid

dynamics code and since each simulation can run independently, parallel computers should be used to run multiple simulations concurrently. Therefore, an investigation into good rules for evolving different populations and migrating between populations is also needed.

Because of its parallel nature and its ease of application to a variety of open-channel flow problems, a genetic algorithm approach holds much promise for efficient design optimization. For the case of channel contractions, it was shown that the genetic algorithm was able to find much improvement designs. For more complicated geometries, many local, suboptimal minima are likely to exist, which may cause much difficulty for gradient-based algorithms. But because it is a probabilistic search algorithm, a genetic algorithm may be able to find the global minimum or more fit local minima, more frequently than a gradient-based algorithm.

Acknowledgments

This work was supported by the National Science Foundation, through funding provided to the Engineering Research Center for Computational Field Simulation.

The author wishes to thank Dr. J. E. Boggess for introducing the author to the topic of genetic algorithms, Dr. R. C. Berger for allowing the author to modify HIVEL2D, and Dr. D. H. Huddleston for encouraging and aiding the author in this endeavor.

References

[1] Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

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

[3] Stockstill, R. L. and Berger, R. C., "HIVEL2D: A Two-Dimensional Flow Model for High-Velocity Channels", Technical Report REMR-HY-12, U.S. Army Corps of Engineers, Waterways Experiment Station, 1994.

[4] D. H. Huddleston, B. K. Soni, "Application of a Factored Newton-Relaxation Scheme to Calculation of Discrete Sensitivity Derivatives", AIAA Paper 94-1894, *AIAA 12th Applied Aerodynamics Conference*, Colorado Springs, June, 1994.