Proceedings of ACEDC'96
PEDC, University of Plymouth, UK

USE OF GENETIC ALGORITHMS TO SOLVE OPTIMAL REGIONAL WATER QUALITY MANAGEMENT PROBLEMS

C A Coello Coello† & J A Figueroa Gallegos‡
†Department of Computer Science, Tulane University, New Orleans, LA 70118 (USA)
‡Escuela de Ingeniería Civil, UN. A. CH., Apdo. Postal 61, Tuxtla Gutiérrez, Chiapas (México)

ABSTRACT

In this paper we present an approach to optimize the design of regional branched wastewater systems using genetic algorithms (GAs). An ad-hoc floating point representation is used, and a sequential parameter adjusment is proposed as an alternative to get an optimal (or at least sub-optimal) answer in a reasonable amount of time, instead of the traditional trial and error process used to fine tune the GA parameters. We also experimented with Gray codes and different mutation operators for the floating point representation. Our results are compared to a Random Polyhedron Search (RPS) technique in an example taken from the literature. Since this work is intended to be applied in real-world problems in México, we had to consider additional constraints on the computer resources available to develop our software.

INTRODUCTION

Wastewater treatment represents an important segment of the public budget of any nation concerned with water quality management. It is therefore desirable that the planning of such wastewater treatment is done on a regional basis so as to maximize cost and efficiency while meeting all the desired objectives of the project. This implies that we should be able to devise the proper size and location of the treatment and conveyance facilities of any wastewater treatment system. However, the problem of regional water quality planning is normally very complex, mainly due to the vast number of alternative designs which are capable to meet the desired water quality objectives. Since the number of alternative designs increases rapidly with the size of the region, the normal trend is to come up with a design that satisfies the objectives, but that is not optimal, and in most cases not even sub-optimal. Several approaches have taken in the past to deal with this problem. Recently, several mathematical programming techniques have been used (1):

1) Linearizing the objective function about a guessed solution (2, 14): It is very straightforward, but not very satisfactory as it is usually unable to identify good candidate solutions, even after several attempts. This makes this technique not very attractive to solve real-world problems.

2) Ranking the extreme points of the model's feasible region (3) : In this case, each extreme point represents a potential optimal solution to the model, which is solved using Murty's extreme points ranking algorithm (3). This technique is able to find the optimal, but it is not very suitable for large-scale problems, as the number of extreme points (each one representing a potential optimal solution) is very large.

3) Dynamic programming (4) : This technique is normally applicable only to unbranched systems (i.e., sources of wastewater and potential treatment plan sites must lie in a linear configuration along the receiving water such that a single line can be drawn to link the first with the last source as well as those in between). Moreover, the computational resources required for this technique are very high, and it turns out to be generally impractical for realistic models because of the "curse of dimensionality" (5).

4) Mixed integer programming (6) : As with dynamic programming, this technique can be unsuitable for solving moderate to large size regional treatment plant location problems.

5) Heuristic procedures (7, 8) : These techniques are normally very easy to implement and very simple to understand. However, it has been argued that they are not very good to accommodate restrictions on the capacities of the conveyances and treatment plants and restrictions on in-stream water quality.

6) Use of search techniques : Ong and Adams (1) formulated the regional treatment plant location problem such that it could be effectively solved by a simple search technique. Then, they used the Random Polyhedron Search (RPS) algorithm (9) to choose a good solution (presumably, at least suboptimal). This method is also another heuristic, but powerful enough as to easily incorporate all the constraints imposed by the design, and to converge in a very short period of time.

DESCRIPTION OF THE PROBLEM

We will consider only the first case mentioned in Ong and Adams' work (1), in which only the constraints on plant capacities and inter-plant transfer are considered, mainly because of space limitations. In the second case,

water quality constraints in the receiving waters are included (1,10).

The regional wastewater treatment plant location problem

Given a region with *n* communities (water sources), we want to decide what is going to be the number and location of regional treatment plants and the assignment of waste sources to the plants such that the overall cost for the system is minimized. Since, in real-world applications, the number of possible treatment sites is finite and is known in advance due to physical and institutional constraints, there is only a limited (but normally very large) number of possible alternatives to choose from.

In real-world situations, there are two types of regional configurations: branched and unbranched. However, since the unbranched system happens to be a special case of the branched system, only the last will be considered in this work. Consider the single branched system shown in Figure 1. Since the actual direction of flow (i.e., upstream or downstream) is not known in advance, it is necessary to include in the model the two possible conveyances, but keeping in mind that the solution will have only one (the non-negative value).

Mathematical Formulation. The objective of this problem is to minimize:

$$\sum_{i=1}^{6} CP_i(Y_i) + \sum_{i=1}^{6} CP_i(Z_i) + \sum_{i=1}^{7} CT_i(X_i) \qquad (1)$$

where *CT* is the cost of treatment and *CP* is the cost of conveyance, X_i is the amount of wastewater to be treated at plant *i*, Y_i is the amount of wastewater carried by the *i*th section of conveyance (in the direction from upstream to downstream) and Z_i is the amount of wastewater carried by the *i*th section of conveyance in the opposite direction of Y_i. This objective is subject to:

1) Flow balance at each source node:

$$X_1 + Y_1 - Z_1 = S_1 \qquad (2)$$
$$X_2 + Z_1 - Y_1 + Y_2 - Z_2 = S_2 \qquad (3)$$
$$X_3 + Z_2 - Y_2 + Y_3 - Z_3 = S_3 \qquad (4)$$
$$X_4 + Y_4 - Z_4 = S_4 \qquad (5)$$
$$X_5 + Z_4 - Y_4 + Y_5 - Z_5 = S_5 \qquad (6)$$
$$X_6 + Z_3 - Y_3 + Z_5 - Y_5 + Y_6 = S_6 \qquad (7)$$
$$X_7 + Z_6 - Y_6 = S_7 \qquad (8)$$

2) Upper and lower bound constraints:

$$0 \le X_i \le XU_i \qquad i = 1,\ldots,7 \qquad (9)$$

$$0 \le Y_i \le YU_i \qquad i = 1,\ldots,6 \qquad (10)$$

$$0 \le Z_i \le ZU_i \qquad i = 1,\ldots,6 \qquad (11)$$

Random Polyhedron Search

Ong and Adams (1) use the so-called Random Polyhedron Search (RPS) algorithm to obtain the optimal (or at least suboptimal) solution. This technique is a variant of the Nelder-Mead algorithm (11), which was adapted to deal with constraints imposed on the decision variables. What the algorithm does is to generate a polyhedron of *N*+1 vertices (where *N*=number of independent variables) located within the feasible region of the model. The decision maker is required to provide a set of initial trial values and an initial search range for each of the independent variables, and the algorithm generates the corresponding vertices of the polyhedron. Nonfeasible points are modified with the aid of the constraints imposed by the system. In its second phase, this algorithm searches for the optimal solution of the problem using the method proposed by Nelder and Mead (11). This algorithm has been successfully applied to analyze chemical equilibrium processes at constant temperature and pressure, optimize the operation of production processes, estimate the parameters of nonlinear differential equations, optimize the stream assimilative capacity management, etc.

Use of the Genetic Algorithm

The genetic algorithm (GA) is a heuristic technique based on the mechanics of natural selection (12) which has been used successfully in the past for solving many optimization problems in different domains (13). We experimented with the Simple Genetic Algorithm (SGA) proposed by Goldberg (13), using both binary and floating point representation.

The traditional representation used by the genetic algorithms community is the binary scheme according to which a chromosome is a string of the form $\langle b_1, b_2, \ldots, b_m \rangle$, where $b_1, b_2, \ldots, b_m$ are called allele (either zeros or ones). Since the binary alphabet offers the maximum number of schemata per bit of information of any coding (13), its use has became very popular among scientists. This coding also facilitates theoretical analysis of the technique and allows elegant genetic operators. However, since the "implicit parallelism" property of GAs does not depend on using bit strings (15) it may be worthwhile to experiment with larger alphabets and even with new genetic operators. In particular, for optimization problems in which the parameters to be adjusted are continuous, a floating point representation scheme seems a logical choice. According to this representation scheme, a

chromosome is a string of the form $\langle d_1, d_2, \ldots, d_m \rangle$, where $d_1, d_2, \ldots, d_m$ are digits (numbers between zero and nine).

One of the advantages of floating point representation is that it has the property that two points close to each other in the representation space must also be close in the problem space, and vice versa (15). This is not generally true in the binary approach, where the distance in a representation is normally defined by the number of different bit positions. However, it is possible to reduce such discrepancy by using Gray coding (14,15).

To convert a binary number **b** = $\rangle b_1, b_2, \ldots, b_m \langle$ into a Gray code number **g** = $\rangle g_1, g_2, \ldots, g_m \langle$, where m denotes the number of bits, we can use the two following procedures (15):

procedure Binary-to-Gray
begin
    $g_1 = b_1$
    for k = 2 **to** m **do**
        $g_k = b_{k-1}$ *XOR* b_k
end

procedure Gray-to-Binary
begin
    $value = g_1$
    $b_1 = value$
    for k = 2 **to** m **do**
        begin
        if $g_k = 1$ **then** *value = NOT value*
        $b_k = value$
        end
    end

The Gray code representation has the property that any two points next to each other in the problem space differ by only one bit (15). In other words, an increase of one step in the parameter value corresponds to a change of a single bit in the code. This is a well known technique used to reduce the distance of two points in the problem space, and it is argued to bring some benefits because of its adjacency property, and the small perturbation caused by many single mutations. However, the use of Gray codes didn' t help much in this particular application, as we will see in the results.

It is interesting to notice that even when the mathematical formulation of this problem seems very easy to incorporate into the GA, it turns out that in practice, the GA has problems handling too many constraints using a simple penalty function of the form:

$$fitness = \frac{1}{(Cost \times (penalty \times viol + 1))}$$

where *Cost* is given by equation (1), *penalty* is a value to be determined empirically, and *viol* is the maximum error accumulated from equations (2) to (8).

To help the GA to improve its performance, we decided to add the following part of the solution strategy proposed by Ong and Adams (1) into the GA' s evaluation function:

if $S_1 - X_1 < 0$ $\quad Y_1 = 0$ and $Z_1 = |S_1 - X_1|$
$S_1 - X_1 = 0$ $\quad Y_1 = 0$ and $Z_1 = 0$
$S_1 - X_1 > 0$ $\quad Y_1 = S_1 - X_1$ and $Z_1 = 0$

if $S_2 + Y_1 - Z_1 - X_2 < 0$ $\quad Y_2 = 0$ and
$Z_2 = |S_2 + Y_1 - Z_1 - X_2|$
$S_2 + Y_1 - Z_1 - X_2 = 0$ $\quad Y_2 = 0$ and $Z_2 = 0$
$S_2 + Y_1 - Z_1 - X_2 > 0$ $\quad Y_2 = S_2 + Y_1 - Z_1 - X_2$
and $Z_2 = 0$

The values of Y_i and Z_i for i=3 to 6 are calculated in a similar manner.

This incorporates certain knowledge about the domain, because those equations correspond to equilibrium conditions that we know must be satisfied in order to generate a valid solution.

We worked with a simple genetic algorithm (14) implemented in Turbo Pascal 7, using Kent Porter' s technique (16) for dynamic memory management. Our implementation included two-point crossover, traditional mutation, binary tournament selection and a population size of 450 chromosomes. The length of each chromosome was 44 when using binary representation, and only 14 when using floating point representation. The mutation was the only operator that had to be redefined for using floating point representation. Instead of negating the corresponding bit (as when using binary representation), the gene selected to mutate was replaced by a random digit (a number between 0 and 9). Each GA ran for 200 generations using the methodology described next to adjust its parameters.

Selecting the Parameters of the GA

One of the main problems when using GAs is how to choose the most appropriate parameters values (i.e., the population size, maximum number of generations, mutation and crossover rate). This is normally a trial and error process which takes a considerable amount of time. Based on our experience, we can say that it turns out to be harder to fine tune the parameters of the GA when a floating point representation is used.

Nodes		Cost function
From	To	($\$10^6$)
1	2	$(0.026830 - 0.0017890Y_1)Y_1$
2	1	$(0.032196 - 0.0021468Z_1)Z_1$
2	3	$(0.080490 - 0.0053670Y_2)Y_2$
3	2	$(0.096588 - 0.0064404Z_2)Z_2$
3	6	$(0.067075 - 0.0044725Y_3)Y_3$
6	3	$(0.080490 - 0.0053670Z_3)Z_3$
4	5	$(0.053660 - 0.0035780Y_4)Y_4$
5	4	$(0.064392 - 0.0042936Z_4)Z_4$
5	6	$(0.080490 - 0.0053670Y_5)Y_5$
6	5	$(0.096588 - 0.0064404Z_5)Z_5$
6	7	$(0.134150 - 0.0089450Y_6)Y_6$
7	6	$(0.160980 - 0.0107340Z_6)Z_6$

Table 1 - cost function for wastewater conveyance

This is a very problematic situation since, as we will see in the comparison of results, floating point representation produces better results than binary representation. Obviously, any optimization system will not be very useful if its outcomes are completely unpredictable. After a lot of experimentation, we came out with a systematic empirical process that seems to be able to generate optimal (or at least suboptimal) solutions in a reasonable amount of time. However, we don' t have yet any theoretical support of its reliability, even though the empirical evidence is quite solid, since we have used several times in the past (17, 18, 19). The method is the following:

- Choose a certain value for the random numbers seed and make it a constant.
- Make constants also the population size and the maximum number of generations (we used 450 chromosomes and 200 generations, respectively).
- Loop the mutation and crossover rates from 0.1 to 0.9 at increments of 0.1 (this is actually a nested loop. This implies that 81 runs are necessary.
- For each run, update two files. One contains only the final costs, and the other has a summary that includes, besides the cost, the corresponding values of the design parameters and the mutation and crossover rates used.
- When the whole process ends up, the file with the costs is sorted in ascending order, and the smallest value is searched in the other file (or the greatest, if we are maximizing), returning the corresponding design parameters as the final answer.

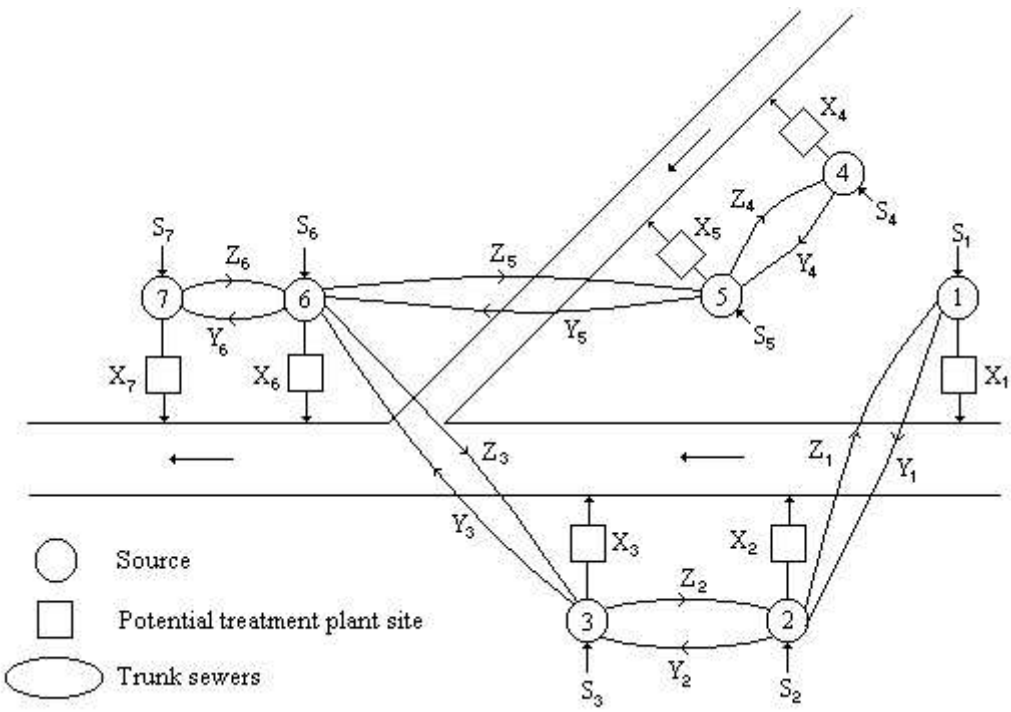

Figure 1 - Schematic diagram of a single branched system.

i	1	2	3	4	5	6	7
S_i	0.3	0.2	2.1	0.7	0.1	2.5	0.7

Table 2 - Wastewater generated at each source

Notice that in this process, the population is not erased or reinitialized, but instead is re-used with new parameters. This is some sort of dynamic adjustment of

parameters that turns out to work very well with both representation schemes, but that due to the size of the increments, seems to favor the floating point representation.

Since each run is completely independent from the others, we can run all these processes in parallel, so that the total execution time will be practically the same required for a single run (about 58 seconds in an IBM PC DX/2 running at 66 MHz and with a mathematical coprocessor).

An Example

Consider the hypothetical problem formulated by Deininger and Su (14), and solved by Ong and Adams (1,10). Briefly, they considered a region which contains seven communities located along a river and its tributary (Figure 1). There are 7 potential sites available for the construction of wastewater treatment plants. The objective is to find the size and location of wastewater treatment plants and the interconnecting conveyance system which will minimize the total cost. The required information to solve this problem is the following (1,5):

(a) The cost function for wastewater conveyance is shown in Table 1.

(b) Cost function for treatment plants:

$$CT_i(X_i) = (0.5036 - 0.0227X_i)X_i \qquad i = 1,\ldots,7$$

(c) The amount of wastewater generated at each source is given in Table 2.

(d) The upper bounds for the decision variables are shown in Table 3. XU_i is the maximum amount of wastewater that is permitted to be treated at plant *i* (due to physical and/or water quality constraints), YU_i is the maximum amount of wastewater that can be carried by the *i*th section of conveyance in the direction of Y_i (due to physical constraints, if any), and ZU_i is the maximum amount of wastewater that can be carried by the *i*th section of conveyance in the opposite direction of YU_i.

COMPARISON OF RESULTS

The results obtained by the RPS algorithm and the GA are compared in Table 4. The GA was used both with binary (B) and floating point representation (FP). We also tried binary Gray coding (G) and floating point representation with a normalized mutation operator (FP-N), in which the current value of the gene selected would be altered by an integer between -1 and 1, passing by zero, depending on the random number generated. Also, following the advise of Michalewicz (15), we implemented a non-uniform mutation operator. This new operator is defined as follows (15) : if $s_v^t = \rangle v_1, \ldots, v_m \langle$ is a chromosome (*t* is the generation number) and the element v_k was selected for this mutation, the result is a vector $s_v^{t+1} = \rangle v_1, \ldots, v_k^{'}, \ldots, v_m \langle$, where

$$v_k^{'} = \begin{cases} v_k + \Delta(t, UB - v_k) & \text{if a random digit is 0,} \\ v_k - \Delta(t, v_k - LB) & \text{if a random digit is 1,} \end{cases}$$

and *LB* and *UB* are lower and upper domain bounds of the variable v_k (0 and 9, in our case). The function $\Delta(t,y)$ returns a value in the range [0,*y*] such that the probability of $\Delta(t,y)$ being close to 0 increases as *t* increases. This property causes this operator to search the space uniformly initially (when *t* is small), and very locally at later stages; thus increasing the probability of generating the new number closer to its successor than a random choice. We used the same function suggested by Michalewicz (15):

$$\Delta(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{T}\right)^b} \right)$$

where *r* is a random number from [0..1], T is the maximal generation number, and *b* is a system parameter determining the degree of dependency on iteration number (we used *b*=5, as suggested by Michalewicz (15)). Our results using floating point representation with non-uniform mutation (FP-NU) are also shown in Table 4, and as you will see they turn out to be the same than when using the normalized mutation operator mentioned before.

As you can see, the use of floating point representation with a mutation operator that generated digits provided the best results, only compared to those produced with the RPS algorithm.

The optimal configuration generated by the GA (and the RPS algorithm) is shown in Figure 2.

i	1	2	3	4	5	6	7
XU_i	4.8	4.8	5.9	4.0	4.0	6.6	6.6
YU_i	0.3	0.5	2.6	0.7	0.8	5.9	—
ZU_i	6.3	6.1	4.0	5.9	5.8	0.7	—

Table 3 - Upper bounds for the decision variables

The convergence graph of the GA that produced the best results is shown in Figure 3. As you can see, it is not until the last twenty generations that we were able to obtain the best solution, which by the way, is the global optimum, according to Ong and Adams (1).

FUTURE WORK

We are still running a lot of different experiments to try to minimize the parameters of the GA. We are running different strategies to help the GA converge in a shorter amount of time with smaller populations. For example, we are trying to eliminate duplicates in our population, and to use elitism (i.e., to keep the best individual in each generation without crossing it with any other chromosome), but the results found so far have not been very encouraging. We have successfully solved several examples found in the literature and we are looking ahead to solve real-word problems within the next few months.

We are interested in using other techniques for adjusting the parameters of the GA, such as fuzzy logic. Also, we would like to experiment with other GA operators, such as different types of mutation operators, and some forms of restricted crossover. Furthermore, we want to try some other alternative representation schemes. For example, we are interested in using a binary representation of the IEEE formats of 8, 16 and 32 bits, to represent floating point numbers.

We are also working in the second case of this problem (1), in which more variables are introduced and the search space and the CPU time required considerably increase.

Finally, we are interested in using other powerful heuristic techniques, such as Tabu Search (20, 21), probably in combination with the GA.

Method	**RPS**	**GA (B)**	**GA (FP)**	**GA (G)**	**GA (FP-N)**	**GA (FP-NU)**
x_1	0.00	0.10	0.00	0.40	0.00	0.00
x_2	0.00	0.00	0.00	0.00	0.00	0.00
x_3	0.00	0.00	0.00	0.40	0.00	0.00
x_4	0.00	0.30	0.00	0.00	0.00	0.00
x_5	0.00	0.00	0.00	0.10	0.00	0.00
x_6	6.60	6.20	6.60	5.70	5.90	5.90
x_7	0.00	0.00	0.00	0.00	0.70	0.70
y_1	0.30	0.20	0.30	0.00	0.30	0.30
y_2	0.50	0.40	0.50	0.10	0.50	0.50
y_3	2.60	2.50	2.60	1.80	2.60	2.60
y_4	0.70	0.40	0.70	0.70	0.70	0.70
y_5	0.80	0.50	0.80	0.70	0.80	0.80
y_6	0.00	0.00	0.00	0.00	0.00	0.00
z_1	0.00	0.00	0.00	0.10	0.00	0.00
z_2	0.00	0.00	0.00	0.00	0.00	0.00
z_3	0.00	0.00	0.00	0.00	0.00	0.00
z_4	0.00	0.00	0.00	0.00	0.00	0.00
z_5	0.00	0.00	0.00	0.00	0.00	0.00
z_6	0.70	0.70	0.70	0.70	0.00	0.00
Cost ($\$10^6$)	2.73009	2.792489	2.73009	2.893132	2.810168	2.810168

Table 4 - Comparison of results

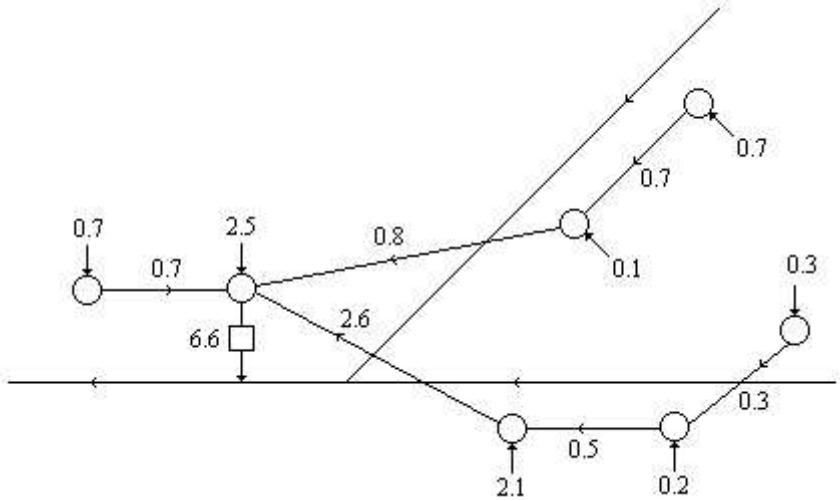

Figure 2 - The optimal configuration for the given hypothetical region (case 1).

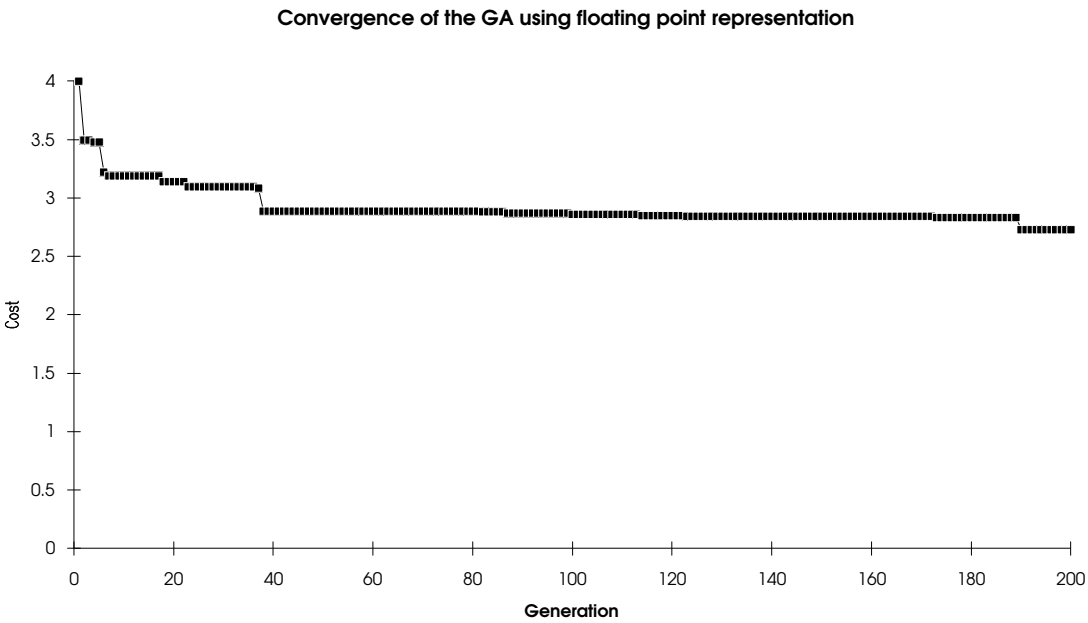

Figure 3 - Convergence of the GA using floating point representation (FP) with a mutation operator that generates random digits.

CONCLUSIONS

We have shown a successful application of a genetic algorithm to a numerical optimization problem which is highly non-linear. We have seen how, even when the specifications of this problem seem very simple and straightforward, the shape of its search space is very difficult for the GA to track. We have proposed the use of a very simple floating point representation, together with a systematic parameter fine tuning procedure, to deal with this kind of problems. Our preliminary results seem to indicate that floating point representation is not only faster, but also better in terms of the quality of solutions that it may find. This representation is particularly important for large domains where binary representation would require excessively long strings. Also, this representation is intuitively closer to the problem space, and is easier to design for it specific operators that incorporate knowledge about the domain, as we did in this application. This is very important

when we have a lot of complex constraints, like in this case.

The main goal of this on-going research is to produce a computer program able to generate optimal (or at least sub-optimal) designs of regional branched wastewater treatment systems in a reasonable amount of time. Such tool would be very useful in real-world applications in the state of Chiapas, in which there is a great need of such wastewater treatment plants. So far, our prototype seems to work well, but there is still a lot of room for improvement, and issues such as convergence time and limitations on the computer resources available have to be taken into account, considering that this work has been developed in México.

We still have a lot of work to do, before we may claim that the GA is a highly-reliable numerical optimization technique, and several problems remain open for research in future. However, we hope that in the next few years we can be able to control this technique in such a way that we may produce designs and to solve complex real-world optimization problems using this old mechanism that Nature has been using in our planet for millions of years.

REFERENCES

1. Ong, S. L., and Adams, B. J., 1987, ' Application of an efficient search technique for solving optimal regional water quality management problems' , Civil Engineering Systems, Vol. 4, September, 131-141.
2. Converse, A. O., 1972, "Optimum number and location of treatment plants", Journal of Water Pollution Contr. Fed., 44, 1629-1636.
3. Murty, K. G., 1971, ' Solving the fixed charge problem by ranking the extreme points' Operations Research, 3, 479-484.
4. Sniedovich, M., 1992, ' Dynamic Programming' , M. Dekker, New York, U.S.A.
5. Bellman, R., 1957, ' Dynamic Programming' Princeton University Press, Princeton, New Jersey, U.S.A.
6. Adams, B. J., 1981, ' Discussion of least-cost optimization for areawide wastewater management using mixed integer programming' , in Jenkings, S. H. (Editor) Water Pollution Research and Development, Part 4, IAWPRC.
7. Jarvis, J. J., Rardin, R. L., Urger, V. E., Moore, R. W. and Schimpeler, C. C., 1981, ' Optimal design of regional wastewater system: a fixed-charge network flow model' Operations Research, 26, 538-550.
8. Voutchkov, N., 1993, ' Heuristic screening methodology for regional wastewater-treatment planning' ,Journal of Environmental Engineering, 119, July/August, 603-614.
9. Ong, S. L., and Adams, B. J., 1984, ' Random polyhedron search = a nonlinear optimization algorithm' ,Journal of Advanced Management Studies, 161-179.
10. Ong, S. L., Adams, Barry J., 1990, ' Capacity Expansion for Regional Wastewater Systems' Journal of Environmental Engineering, Vol. 116, No. 3, May/June, 542-560.
11. Nelder, J. A., and Mead, R. A., 1964, ' Simplex method for function minimization' ,Computer Journal, 7, 308-313.
12. Holland, J. H., 1975, ' Adaptation in Natural and Artificial Systems' , University of Michigan Press.
13. Goldberg, D. E., 1989, ' Genetic Algorithms in Search, Optimization and Machine Learning' Addison-Wesley Publishing Co.
14. Deininger, R. A., 1965, ' Water quality management: the planning of economically optimum pollution control systems' , First Annual Water Resources Conference, Chicago, American Water Resources Association.
15. Michalewicz, Z., 1992, ' Genetic Algorithms + Data Structures = Evolution Programs' Springer-Verlag, second edition.
16. Porter, K., 1988, ' Handling Huge Arrays' Dr. Dobb' s Journal of Software Tools for the Professional Programmer, Vol. 13, No. 3, 60-63.
17. Coello, Carlos, Alonso Farrera, F., 1995, ' Use of Genetic Algorithms for the Optimal Design of Reinforced Concrete Beams' , Computer Aided Optimum Design of Structures IV. Structural Optimization. Edited by S. Hernández, M. El-Sayed & C. A. Brebbia, Computational Mechanics Pub., Southampton, UK, 209-216.
18. Coello, Carlos A., Alonso Farrera, F., 1995, ' Optimal Design of Axially Loaded Non-prismatic Columns via Genetic Algorithms' Sixth International Conference on Computing in Civil and Building Engineering, Berlin, Germany, Edited by Peter Jan Pahl and Heinrich Werner, Vol. 1, A. A. Balkema, Rotterdam, Netherlands, 691-696.
19. Coello, Carlos A., Christiansen, Alan D., Alonso Farrera, F., 1995, ' Use of Genetic Algorithms for Multiobjective Optimization of of Counterweight Balancing of Robot Arms' TAI' 95,IEEE Computer Society Press, USA.
20. Glover, Fred, 1989, ' Tabu Search - Part I' ORSA Journal on Computing, 1 (3): 190-206.
21. Glover, Fred, 1990, ' Tabu Search - Part II' ORSA Journal on Computing, 2:4-32.