A Genetic Algorithm to solve an Integer Goal Programming Model for the Higher Education.

Rafael Caballero
Julián Molina
Trinidad Gómez
Mariano Luque
Angel Torrico
Department of Applied Economics (Mathematics). University of Málaga.
Campus El Ejido s/n
29071 Málaga (Spain)
e-mail: julian.molina@uma.es

ABSTRACT

In this work a genetic algorithm for the resolution of an Integer Goal Programming model is shown. In general, this type of problems produce serious difficulties for its resolution using traditional algorithms from Integer Programming, being in most of the real problem cases too expensive computationally to solve it. However, it will be shown how this new type of methods, the genetic algorithms, can solve real integer goal programming problems efficiently with a reduced computational cost. To this aim, we solve an Integer Goal Programming Model for the Higher Education applying a genetic algorithm and a traditional algorithm. For these resolutions, the computational cost is analysed to show the advantages of a genetic algorithm for the resolution of complex real problems.

Keywords: Integer Programming, Goal Programming, genetic algorithms, Multiobjective Optimization.

1. Introduction.

Evolutionary algorithms have become a very used tool for optimization, machine learning, design problems, neuronal network, search, and other fields on Mathematics and Engineering. The main point of these types of algorithms is simulating natural evolution to find solutions for complex problems. This way, the famous naturalist Charles Darwin defined, in his famous book in 1859, Natural Selection or Survival of the Fittest as the preservation of favourable individual variations, as well as the destruction of those injurious ones. In nature, individuals have to adapt to their environment to survive by means of evolution, in which those variations favouring their competitiveness are preserved, and those aspects that weaken their adaptation are eliminated. These characteristics, favourable or unfavourable, are stored and controlled in some units called genes, and these genes are grouped forming chromosomes.

In the late 60s, John Holland became interested in applying the principles of the natural evolution for the resolution of complex problems in the field of machine learning, developing a technique that will be the root of the genetic algorithms. In 1989 Goldberg published a book including a solid background for this type of strategies and where you could find more than 70 real applications of genetic algorithms. The main characteristic of a genetic algorithm is the use of a *recombination* operator or *crossover* as the search mechanism. This way, parents swap part of the information on their genes to create new offspring with some new features and with some of the parents' characteristics. This operator is based on the fact that different components of good solutions can be evolved independently and then combined to form better solutions. Additionally to the crossover, a *mutation* operator is used to maintain an appropriate diversity level among the population, by means of making the genetic search random. Each population member is represented by a set of chromosomes, that originally

were simply binary strings. This *representation* of the population's elements plays a main role in a genetic algorithm, since it must be able to represent all the range of feasible solutions for the problem, as well as the characteristics that we would like to obtain with the final solution. Given a representation, an initial population is generated and, by means of the *selection* operator, the more fit individuals (parents) are selected to produce new offspring. The quality of each individual, this is, the quality of each gene string, is settled by a *fitness* function. When a genetic algorithm is used to solve an optimization problem, the *fitness* function is the optimising function, $f(x)$, although it can be modified for example to penalise the constraint violation or to foment diversity among the population. A more detailed description of each element on a genetic algorithm can be found in Goldberg (1989).

Finally, we want to remark the capacity of a genetic algorithm to use and take advantage of some elements from other types of methodologies, like simulated annealing, taboo search or GRASP. As consequence of this great adaptability to solve different classes of complex problems, the number of real applications of genetic algorithm in the literature is also enormous, as you can observe in Gandibleux and Ehrgott (2000).

2. Multiobjective Programming with Genetic Algorithms.

In comparison with single objective optimisation problems, little research has been reported on the field of Multiobjective optimisation with genetic algorithms. Goldberg (1989) mentions the first attempts of using genetic search in a multicriteria problem in the late 60s. These first attempts were focused in the aggregation of the criteria by means of scalarization. However, from the late 60s, other different approaches have appeared in the literature, based on aggregation, on the Pareto efficiency and on other types of orders, as you can see for example in Fonseca and Fleming (1993). Jakob *et al.* (1992) made use of the Weighted-Sums method to aggregate the criteria, but this method usually produces not so balanced solutions and therefore it doesn't explore the whole efficient frontier. Ritzel *et al.* (1994) tried to use the Constraint Method to obtain a better approximation of the efficient frontier, although with a bigger computational cost because of multiple resolutions must be carried out. To avoid problems when aggregating the criteria, many of the efforts in the literature have gone toward approaches not based on scalariazation. This way, Schaffer (1985) developed the method VEGA (Vector Evaluated Genetic Algorithm) whose only difference with an usual genetic algorithm is the way the selection is carried out for recombination. In this method, on each generation the population groups in a certain number of subpopulations (usually the number of criteria) where only one criteria is taken into account. This is a very popular method in the literature, although in many problems it produces not so balanced solutions. Fourman (1985) used a lexicographic order to carry out the selection, obtaining good results since with the lexicographic order you can always compare two individuals and then can build a ranking among the population. This approach also has the advantage of not needing to normalize the criteria, because of these type of binary comparisons.

Also you can find some approaches based on Pareto domination. With this type of methods, the fitness value of each individual depends on its efficiency or dominance inside the population. This way, Goldberg (1989) made a non-dominance ranking to solve the "speciation" problems of VEGA. This method showed superior to VEGA in some cases, as you can see in Hilliard *et al.*(1989). Fonseca and Fleming (1993) developed this idea into the MOGA method (Multi Objective Genetic Algorithm), where each individual's ranking depends on the number of elements in the population dominating it. Srinivas and Deb (1994) used the idea of the non-dominance ranking in the NSGA method (Non-Dominated Sorting Genetic Algorithm) that also showed efficient when solving problems with multiple criteria.

Finally, Sakawa *et al.*(1997,2000) developed a modified genetic algorithm for 0-1 Fuzzy Multiobjective Programming. More research relating Multiobjective Programming and evolutionary algorithm can be found among others in Busacca *et al.* (2001), Jaszkiewicz (2002), Hanne (1999), Sarker *et al.* (2002), Coello *et al.* (2000), etc. A survey on Multiobjective Evolutionary Algorithm can be found in Van Veldhuizen and Lamont (2000).

On the other hand, we cannot find many papers on the literature relating specifically Goal Programming with genetic algorithms. This way, Gen *et al.*(1997) developed a genetic algorithm for Fuzzy Nonlinear Goal Programming models. Mirrazavi *et al.* (2001) analysed the use of genetic algorithms to solve Integer Goal Programming models on a general frame. Wilson and Macleod (1993) and Chen and Liu (1994) used genetic algorithm to solve a Weighted Goal Programming model.

But, in fact, a Lexicographic Goal Programming problem can adapt perfectly to the use of a fitness function based on a lexicographical order, this is, the lexicographic order corresponding to the priority levels of the LGP problem. This way, you can take advantage in a simple way of the good properties of this type of lexicographic genetic algorithms, as it is shown in Fourman (1985).

3. Application.

As an example to show the efficiency of this type of algorithms, we will solve in this section a real Lexicographic Integer Goal Programming model. We chose this problem because it is a very well known fact that among the Mathematical Programming problems, the Integer Programming problems offer a bigger computational complexity when being solved. This way, the Integer Programming problems require, in general, a great computational cost, in comparison with the rest of Mathematical Programming problems, even making the resolution unfeasible in many cases. However, Integer Programming is a very important branch of Mathematical Programming because you can find an enormous variety of real problems of this type, related with areas so diverse as the combinatorial analysis, planning and production, transport, assignment, scheduling, flow, location, distribution, machine sequencing, etc. On the other hand, Integer Goal Programming is one of the more popular approaches on the literature on Multiobjective Integer Programming because, with this type of problems, obtaining completely or a good approximation of the efficient set is a really difficult task, being then more easy to obtain satisfacing solutions.

This problem is based on a model developed in Caballero *et al.* (2002). It is a Lexicographic Integer Goal Programming model for the efficient assignment of the financial resources of a Spanish university and we will apply it to the particular case of the University of Málaga. This problem uses 3124 integer variables (22 variables by department, having the University of Málaga 142 departments), 1420 deviational variables (continuous), 712 hard constraints and 1420 soft constraints associated with the goals of the problem. These goals are allocated into five priority levels.

Firstly, this problem, with the data of the University of Málaga, was solved with the Multiobjective Programming software PROMO (Caballero *et al.* (2000)). This software uses the CPLEX Integer Programming libraries, version 6.5.1., that are among the most efficient ones for the resolution of integer problems and offer all the existent techniques to solve a problem of this type efficiently: cuts, pre-processing, heuristics, etc. We carried out several resolutions with different parameters in a personal computer Pentium III 1 Ghz, requiring all

them more than 3 hours of processing for its resolution, and some of them even more than 4 hours.

Later on, a genetic algorithm has been developed to solve this problem and to evaluate the computational cost saving that a genetic algorithm can offer for this type of problems. This algorithm uses an integer representation for the 22 variables of each department. For the initial population, 1300 individuals were generated (for the whole resolution the population's size was 1300 individuals), where for each individual some of the variables were randomly generated and the rest was calculated according to these values to ensure the feasibility of each one of the initial population's individuals. However, for the rest of the algorithm the procedure adopted to deal with feasibility was death penalty, this is, if a chromosome encodes an infeasible solution, it is destroyed and replaced).

For the fitness evaluation, we used a function based on the lexicographic order and the goals associated to the ILGP problem. This way, each individual's evaluation consisted on a vector with the value of the achievement functions associated to the five priority levels of the problem. This lexicographic evaluation allows to carry out the population's complete ordination (a complete ranking), since any two elements are comparable by means of this lexicographical order. This way, the used selection operator was a lineal ranking, where 200 couples (parents) were selected to reproduce, among the population's 1300 individuals.

Each one of these couples gave place to an offspring of 4 individuals. Then, this reproduction operator produces a total of 800 new individuals that will substitute the 800 worse individuals of the previous generation, becoming an elitist algorithm.

Among this offspring of 4 children by couple, two of them are built exchanging (conditionally) the 22 values of the departments of the parents. This is, one of the children receives entirely (the 22 genes) the father's even departments and the mother's odd departments, and the other son receives them inversely. Then, this is an uniform crossover of 142 points. However, this crossover operator firstly estimates the goal values on each department of the individual to which this son will substitute in the population. This way, this son will receive from one of the parents the 22 values of a department if it is estimated that this exchange will improve the current values on the individual to be substituted. Conversely, if no improve is estimated, the son will receive the 22 values from the individual to be substituted, but it is carried out a random test that can allow to take the parents' department values although it introduces a worse estimation. For each parent's other two children it was simply carried out an uniform crossover of 142*22 points with a random test, this is, and exchange of all the variables. Relating the mutation operator, mutation was allowed (with probability 0,01) only for the genes of those departments whose goal values were under the target value. As we mentioned above, death penalty was used to deal with infeasibility, this is, any infeasible individual obtained by the crossover or the mutation operator is eliminated. We also used a measure of the infeasible individuals rate (number of infeasible individuals created in comparison with the total offspring) to control some parameters of the mutation and crossover operator. With this rate we were trying to estimate the efficiency of the crossover and mutation operator, because if this rate is high then most of the computational effort of this two operators is lost because of the death penalty. Then, we used this rate as a feedback parameter to adjust some other parameters of the crossover and mutation operator. Also, we could found this rate as a good estimation of the convergence state, because as the population become more uniform (convergence is near to be reached) this rate tends to 0. The reason for this relation could be found in the building blocks hypothesis (Goldberg (1989)), because as the convergence is being reached the diversity of the blocks to be combined on each crossover

operation is smaller and these different blocks better fit, producing then less infeasibility. In this example, this rate showed to be so useful to improve the crossover and mutation operator and to estimate a stopping condition.

This algorithm was also combined with a local search (30 iterations) by means of a GRASP method (an heuristic using a greedy randomised function) starting from the 10 better individuals on the ranking each 50 generations. Finally, as a stopping condition, we used the infeasibility rate to avoid a large number of iterations and finally we could carry out no more than 200 iterations for each resolution.

This algorithm was implemented using C++ (Microsoft Visual C++ 6.0 compiler) language to obtain a Windows software supporting some graphics to show the evolution of the resolution process. With this simple genetic algorithm we solved our ILGP problem again with a personal computer Pentium III 1 Ghz, and we obtained the CPLEX optimal solution within a computational time not exceeding in any case 20 minutes. This fact showed us the efficiency that this type of algorithm can offer for complex integer problems as our example was.

4. Conclusions.

With this work we seek to show the efficiency and the advantages that a genetic algorithm can introduce when solving Integer Goal Programming complex problems. To this aim, it has been solved an Integer Goal Programming complex real problem with a traditional exact method (CPLEX Branch and Bound method), using some of the most efficient libraries in this field, the CPLEX libraries. Later on it has been developed and implemented a simple genetic algorithm to solve this problem, obtaining the same optimal solution, but within a saving of computational time of more than 80%. This example could show the proposal of this work: when solving a complex IGP problem, the use of a genetic algorithm can be an efficient option from the point of view of computational cost as well as the quality of the solution.

5. References.

1. **Busacca, D., Marseguerra, M., Zio, E.** (2001). *Multiobjective Optimization by genetic algorithms: Application to safety systems*. Reliability Engineering and System Safety, v. 72, pp. 59-74.
2. **Caballero, R., Gónzalez, M., Hernández, M., Luque, M., Molina, J., Ruiz, F.** (2002). *A Decision Model, via Integer Goal Programming, for Hiring and Promoting Staff in the Deaprtments of a University.* Köksalan, M., Zionts, S.(ed.) In Multiple Criteria Decison Making in the New Millennium. Springer-Verlag. pp. 403-413.
3. **Caballero, R., Ruiz, F., Luque, M., Molina, J.**(2000). *PROMO (Programación Multiobjetivo)*. Registered Software. R.P.I. Ma-6739.
4. **Chen., Y., Liu, C.**(1994) *Multiobjective VAR Planning using the goal attainment method*. IEE Proceedings on Generation, Transmission and Distribution. vol. 141, pp. 227-232.
5. **Coello, C.A., Christiansen, A.D.** (2000): *Multiobjective Optimization of trusses using GAs*. Computers and Structures. vol. 75, pp. 647-660
6. **Fonseca, C.M., Fleming, P.J.** (1993). *Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion and Generalization*. In S. Forest (ed.), Proceedings of th Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, pp. 416-423.

7. **Fourman, M.P.** (1985). *Compactation of symbolyc layout using GAs*. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, pp. 141-153.
8. **Gen, M., Ida, K, Lee, J., Kim, J.** (1997): *Fuzzy Nonlinear Goal Programming Using GAs*. Computers Ind. Engng. vol. 33, pp. 39-42
9. **Gandibleux, X., Ehrgott, M.** (2000). *A survey and annotated bibliography of multiobjective combinatorial optimization*. OR Spektrum, 22, pp. 425-460.
10. **Goldberg, D.E.** (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co.
11. **Hanne, A.** (1999): *On the convergence of Multiobjective Evolutionary Algorithms*. European Journal of Operational Research. vol. 117, pp. 553-564
12. **Hilliard, M.R., Liepins, M. Palmer, M., Rangarajen, G.** (1989). *The computer as a partner in algorithmic design: Automated discovery of parameters for a multiobjective scheduling heuristic*. In R. Sharda, B.L. Golden, E. Wasil, O. Balci and W. Stewart (eds.), Impact of Recent Computer Advances on Operations Research. North-Holland Publishing Company.
13. **Jakob, W., Gorges-Schleuter, M., Blume, C.,** (1992). *Application of genetic algorithm to task planning and learning*. In R. Männer and B. Manderick (eds.), Parallel Problem Solving from Nature, 2nd Workshop. North-Holland Publishing Company, pp. 291-300.
14. **Jaszkiewicz, A.** (2002): *Genetic Local Search for Multi-Objective Combinatorial Optimization*. European Journal of Operational Research. vol. 137, pp. 50-71
15. **Mirrazavi, K., Jones, D.F., Tamiz, M.** (2001): *A comparison of genetic and conventional methods for the solution of integer goal programmes*. European Journal of Operational Research. vol. 132, pp. 594-602
16. **Ritzel, B.J., Eheart, W., Ranjithan, S.** (1994). *Using genetic algorithm to a solve a multiple objective groundwater pollution containment problem*. Water Resources Research, 30, pp. 1589-1603.
17. **Sakawa, M.**, **Kato, K., Sunada, H. Shibano, T.** (1997). *Fuzzy Programming for Multiobjective 0-1 Programming Problems through revised genetic algorithms.* European Journal of Operations Research,v. 97, pp. 149-158.
18. **Sakawa, M.**, **Yauchi, K.,** (2000). *Interactive Decision Making for Multiobjective Nonconvex Programming Problems with Fuzzy numbers through coevolutionary genetic algorithms.* Fuzzy Sets and Systems,v. 114, pp. 151-165.
19. **Sarker, R., Liang, K, Newton, C.** (2002): *A new Multiobjective Evolutionary Algorithm*. European Journal of Operational Research. vol. 140, pp. 12-23
20. **Schaffer, J.D.** (1985). *Multiple Objective optimization with vector evaluated genetic algorithms*. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, pp. 93-100.
21. **Srinivas, N., Deb, K.** (1994). *Multiobjective Optimization using Nondominated Sorting in Genetic Algorithm*. Evolutionary Computation, 2, pp.221-248.
22. **Veldhuizen, D., Lamont, G.** (2000): *Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art*. Evolutionary Computation. 8(2), pp. 125-147
23. **Wilson, P.B., Macleod, M.D.** (1993). *Low implementations cost IIR digital filter design using genetic algorithms*. In IEE/IEEE Workshop on Natural Algorithms in Signal Processing. Chelmsford, pp. 4/1-4/8.