Economic Insights – Trends and Challenges	Vol.IV(LXVII) No. 3/2015	125 - 132

Applying Genetic Algorithms to Optimization Problems in Economics

Elena Simona Nicoară

Petroleum-Gas University of Ploieşti, Informatics, Information Technology, Mathematics and Physics Department, Bucureşti Blvd., 39, Ploieşti, 100680, Romania
e-mail: simona.nicoara@upg-ploiesti.ro

Abstract

In the economical environment, most of the optimization problems are related to cost minimization or revenue maximization. In the area of multi-type goods manufacturing, the element to be determined for a maximum profit is often the production structure for a specific manufacturing time horizon. If the problem is simple (small input data and/or short time horizon), no sophisticated algorithm is needed; even empiric solutions are acceptable. But the real instances, in most of the cases, have big input data, they refer to large time horizons and frequently claim the best solutions as fast as possible. In this context, to solve such real problems, both exact and approximate optimization methods are used - from the mathematical programming to greedy methods, genetic algorithms, tabu search and model-based algorithms. Here, the genetic algorithm perspective to find the optimum manufacturing structure for a time horizon is presented and afterwards is tested on a simple problem instance. The results indicate that the genetic algorithms are a valid and efficient alternative to the classical optimization methods in many problems in economics.

Keywords: *genetic algorithm; optimization; profit; manufacturing structure*

JEL Classification: *C63*

Introduction

Optimization problems occur in practically most areas: in economics (investments, production, corporate finance, purchasing, human resources, distribution etc.), in industry (engineering, airlines and trucking, mechanics, oil and gas, electric power etc.), in agriculture and many other fields. The basic model for optimization specifies:

- the variable(s) to be optimized, for example design parameters / number of items to be produced / manufacturing structure etc.,
- the objective(s), for example maximization of profit / flow / resistance / efficiency / utility, or minimization of cost / time spent / loss / risks etc. and, eventually,
- certain constraints.

In the paper we focus the attention on the manufacturing area where the variable to be optimized is the production structure for a specific time horizon and the objective is to maximize the profit associated to that time horizon. Altogether, the further problem analysis and also optimization methods remain valid for all the usual optimization problems in economics.

Classical optimization includes (Fletcher, 1987; Polak, 1997): the *gradient search*, the *Newton method* and its variant called the *Newton method with restarts*, the *greedy method*. Taking into account that many optimization problems in economics are integer programming problems and for such problems the above mentioned methods have certain disadvantages, the *metaheuristics* are recommended in such cases.

The *hill-climbing search* is a very simple metaheuristic, similar to the gradient search without requiring to know the gradient value or its bias. The algorithm alters repeatedly a current candidate solution, initially random generated, and updates it only if the new one is better. The search stops when the current solution is acceptable or the assigned time for run was reached. A more aggressive variant of hill-climbing search is the *steepest ascent/descent hill-climbing*, where many alterations on the current solution are made in each iteration in order to choose the best one to be the next current solution if it is better than the last one. Other variants of hill-climbing search are *hill-climbing with random restarts* and *iterated local search*.

Among the well-known metaheuristics, adequate to optimization in economics, we can also mention (Nicoară, 2013): tabu search (Glover, 1986), GRASP (Feo and Resende,1989), genetic algorithms (Holland, 1975), scatter search (Glover, 1986), guided local search, variable neighborhood search (Mladenović and Hansen, 1997), Ant Colony Optimization (Dorigo, 1992), Particle Swarm Optimization (Kennedy and Eberhart, 1995) and Artificial Bee Colony (Karaboga, 2005).

A broader description and analysis over these methods applied to general optimization problems appear in Nicoară (2012) and Nicoară (2013).

In the following, the framework for a certain class of optimization problem is presented, namely the manufacturing structure optimization. In the next section the genetic algorithm approach to solve such problem is described, and, in order to test it, further a case study is used. Though the analyzed problem it is a specific category of optimization problems, the detailed perspective of genetic algorithm is very easy to adapt to near all the usual optimization problems in economics.

Manufacturing Structure Optimization Problems

In the serial manufacturing shops, the optimal manufacture structure for a time horizon (a shift, a day, a week, a month) is searched. Here, the main optimization criterion is maximization of profit for that time horizon, under the terms of cost limit.

Formally, the problem is defined as it follows:

Input data:

- n types of products to be manufactured: 1, 2, ..., n;
- unitary manufacture cost for each type of product: c_1, c_2, ..., c_n;
- unitary profit for each type of product: pr_1, pr_2, ..., pr_n;
- manufacture capacity for the considered time horizon, for each type of product: r_1, r_2, ..., r_n;
- the maximal cost allowed for the time horizon: *cmax*.

***Requirement*:**

- the manufacture structure (for the specified time horizon):

$$x = (q_1, q_2, ..., q_n),$$

where q_j represents the number of units (or items) of type j.

***Objective*:**

- find the maximum profit manufacture structure, i.e. the manufacture structure that maximize the total profit, computed by formula:

$$f(x) = \sum_{j=1}^{n} q_j * pr_j \quad . \tag{1}$$

***Constraint*:**

- the total cost for the specified horizon to not exceed the maximal cost allowed, *cmax*:

$$\cos t(x) \leq cmax \tag{2}$$

Here, *cost*(*x*) is the total cost of the manufacture structure *x*, computed as:

$$\cos t(x) = \sum_{j=1}^{n} q_j * c_j \quad . \tag{3}$$

The problem is a multimodal integer programming problem, fact that restricts the applicability of many classical optimization techniques. The gradient search method needs the first derivative of the objective function to be determined. If the variable to be optimized is compound - as in the case of manufacturing structure optimization - the method needs all the partial first derivatives of the objective function. This aspect makes more appropriate the gradient method to the problems where the variables to be optimized are in real domains. Moreover, the method guarantees to find the global optimum if and only if the problem is unimodal. Even the Newton method converges more rapidly to the solution than gradient method, it needs additionally the second derivative of the objective function and also it is not adequate to multimodal problems. The greedy method is simple to apply, but does not guarantee that finds the global optimum to all problems in integer domain.

In a nutshell, a multimodal integer programming problem calls for specific measures to be solved, and metaheuristics bring the stochastic factor in such a manner that best or near best solution is found in a reasonable time. Further, the genetic algorithm framework is detailed.

Genetic Algorithms

The genetic algorithm is a metaheuristic in evolutionary computation area, simulating the evolution of natural species, in generations. The basic idea – solving an optimization problem upon a model of natural evolution – was presented by Holland (1975). Such an algorithm transforms a set of mathematical objects (the population of the so-called candidate-solutions of the problem) in a new set of objects (new candidate-solutions), by three operations similar to those in natural evolutionary processes:

- proportional reproduction based on performance (the principle “survival of the fittest”);
- genetic crossover (of two candidate-solutions);
- random mutation.

The genetic operators (selection, crossover, mutation) are applied on “chromosomes”, which are the abstract representations of the candidate-solutions, called also individuals. In Table 1 a comparative taxonomy between evolution of species and optimization problem solving is presented, upon genetic algorithm theory is based on.

Table 1. Evolution of species – solving optimization problems: comparative taxonomy (Nicoară, 2013)

Evolution of species	*Optimization problems solving*
Environment	Problem
Individual	Candidate-solution (individual)
Chromosome	Abstract representation of the candidate-solution (chromosome)
Gene (segment in the chromosome)	Gene (elementary position in chromosome)
Allele (possible value of a gene)	Allele (possible value of a gene)
Genotype	Search space (chromosome space)
Phenotype	Objective space (space of the objective values associated to the candidate-solutions; often, fitness space)
Performance	Performance (fitness, quality)

Initially, a candidate-solution population is (pseudo) randomly generated and each individual is evaluated based on the performance – quality of the solution computed based on the objective(s). Then, in every generation, the genetic operators are applied in order to change the candidate-solution population.

First, there are selected some individuals to mate each other - in fact, to obtain other candidate-solutions based on the "genetic material" of the selected parents. The crossover operator specifies the manner how the offspring are generated. Furthermore, some few individuals are selected to be mutated (in fact, little altered, similar to mutations found in nature). By crossover and mutation, the algorithm explores new regions in the search space and therefore tends to avoid a premature convergence to a local optimum solution. In figure 1 the schemata of a simple genetic algorithm is depicted.

1. t <- 0 (first generation)
2. pseudo-random initialization of the population P_t
3. evaluate(P_t)
4. while evolution is not ended
    4.1. t <- t + 1
    4.2. selection in P_t
    4.3. crossover of parents selected
    4.4. insert the descendents in the new population P'_t
    4.5. mutation for P'_t
    4.6. evaluate P'_t
    4.7. P_t <- P'_t
5. return the best solutions in P_t

Fig. 1. The genetic algorithm framework

Over the generations, the quality of the candidate-solutions in the current population increases based on this specific performance-guided process where the random factor plays an important role – as in all metaheuristics in fact.

The "evolution" of the candidate-solution set is ended when a good enough solution is found or when the a priori settled runtime was reached or when any other criterion or a combination of

criteria was satisfied. The solution of the genetic method is the best solution in the population on the last generation.

A Case Study: A Genetic Algorithm Applied to Manufacturing Structure Optimization Problem

In the following, a simple example is illustrated. Suppose that a manufacture center is producing three types of food supplements: Multimineral (60 capsules per bottle), Calcium (100 tablets per bottle) and Spirulina (90 tablets per bottle). Hence, to meet the notations in the problem formulation, $n = 3$. The unitary manufacture costs, the unitary profits and the manufacture capacity for a time horizon of eight-hour shift are specified in Table 2.

Table 2. Input data for the considered instance

Type of product	Unitary cost (m.u.)*	Unitary profit (m.u.)*	Capacity (items)	*cmax* (m.u.)*
Multimineral	60	50	200	
Calcium	100	30	250	40 000
Spirulina	200	100	100	

Note (*): monetary unit

The optimum manufacture structure for an eight-hour shift is looked for, i.e. the list of three integers (number of units of Multimineral, Calcium and respectively Spirulina to be manufactured) that maximizes the total profit under the constraint to not exceed the total cost, *cmax,* of 40 000 monetary units.

An individual in the population that „evolves", namely a candidate-solution, is therefore a manufacture structure for the shift, more precisely a list of three positive integers, such as:

- (166, 196, 52), translated by 166 bottles of Multimineral plus 196 bottles of Calcium plus 52 bottles of Spirulina;
- (200, 200, 65), translated by 200 bottles of Multimineral plus 200 bottles of Calcium plus 65 bottles of Spirulina;
- (200, 71, 98), translated by 200 bottles of Multimineral plus 71 bottles of Calcium plus 98 bottles of Spirulina;
- (156, 0, 98), translated by 156 bottles of Multimineral, no bottle of Calcium and 98 bottles of Spirulina.

The search space of that instance comprises, based on the proposed genetic encoding and on the production capacity in Table 2, 5 095 551 (201 x 251 x 101) candidate-solutions. Among these, some are not feasible because the constraint is not satisfied. Even so, for that very small instance, many candidate-solutions remain to be verified and evaluated by the total profit assigned to them.

Among the feasible solutions, it is not so simple – in general – to identify the best one. Table 3 presents some examples to illustrate this. The second solution, even if it has the best quality, it is not feasible because the total cost exceeds the maximum cost allowed. Among the feasible ones, the third has the best performance based on the criterion of maximum total profit. However, in realistic circumstances, the ratio profit / cost indicates that the fourth solution is the best.

Table 3. Quality analysis for some candidate-solutions

Candidate-solution (x)	Cost (x)	Profit (x)	Profit (x) / Cost (x)
(166, 196, 52)	39960	19380	0.485
(200, 200, 65)	45000 – not feasible	22500 – best profit	-
(200, 71, 98)	38700	21930	0.566
(156, 0, 98)	28960	17600	0.607

The population of individuals is designated by a matrix $q[nmax][3]$, where $nmax$ is the maximal number of individuals in population, and 3 is the number of types of products. Hence, the individual i is $q[i][j]$, $i \in \{1, ..., nmax\}$, $j \in \{1,2,3\}$.

Initially, a population formed by $N \leq nmax$ candidate-solutions is pseudo-randomly generated. An individual i is a list of three numbers as it follows:

$$q[i][1] \in \{0, ..., 200\},$$
$$q[i][2] \in \{0, ..., 250\} \text{ and}$$
$$q[i][3] \in \{0, ..., 100\}$$

in order to satisfy the capacity constraint for an eight-hour shift (see formulae (2) and (3)). The initial population, containing only feasible solutions, is further evaluated by the fitness function, identical with the objective function (1).

The elitist selection is applied – there are selected the best cr % individuals (the current parents) and the pairs of parents are mated under the one point crossover rule, as Figure 2 depicts. Here, cr represents the crossover rate settled before the algorithm is run.

Parent 1: $q[2]$	(102, | 70, 81)
Parent 2: $q[7]$	(111, |203, 50)
Offspring 1	(102, |203, 50)
Offspring 2	(111, | 70, 81)

Fig. 2. The one-point crossover example, where the crossing point is after the first gene of the chromosome

All the offspring are added to the population and evaluated by the fitness function.

The genetic mutation is applied to mr % individuals, where mr, representing the mutation rate, has a small value (for example 2%), as in nature. Mutation consists in adding a small quantity (between 10 and 20) to a random selected gene of the individual.

After applying the genetic operators in many generations, fact that translates the searching process in the search space, the „evolution" is ended when the population dimension becomes grater than the constant $nmax$ a priori settled.

The solution returned by the algorithm is the best performance candidate-solution in the last generation. Table 4 brings together the stochastic results obtained by runing the genetic algorithm repeatedly with crossover rates between 10% and 50% and mutation rates between 2% and 5%.

Table 4. Results of the genetic algorithm, function of *N* and *nmax*

Solution: manufacture structure	Performance: profit (u.m.)	Initial pop. dimension (*N*)	Maximal pop. dimension (*nmax*)	Generation when the solution was generated
(166, 196, 52)	19 380	30	545	
(200, 80, 100)	**22 400**	40	450	206
(200, 80, 100)	**22 400**	50	379 ... 545	165 ... 248
(200, 71, 98)	21 930	50	300	126
(200, 79, 100)	22 370	80	545	233
(200, 84, 98)	22 320	100	376 ... 545	139 ... 223
(197, 84, 95)	21 870	100	300	101

The best solution obtained by the genetic algorithm is (200, 80, 100), solution returned also by the classical greedy algorithm. From the results table we conclude that does not exist a liniar relation between the quality of the solution and the initial number of candidate-solutions or the maximal population dimension. However, in general, grater values for these parameters lead to better performance for the solutions.

Conclusion

Economics is a rich optimization field. Some problems are of integer programming type, others are real valued optimization and other mixed integer programming. Many optimization methods were proposed, tested, and largerly used in theory and practice as well with purpose to solve such problems: the classical techniques (gradient search, Newton methods, greedy method) and metaheuristics (hill-climbing search, iterated local search, tabu search, GRASP, genetic algorithms, scatter search, guided local search, ACO, PSO, ABC and many more).

In the present paper, the genetic algorithm approach to solve the integer programming optimization problems is presented and further is tested on a simple manufacturing structure optimization instance. The results obtained, similar with the results returned by the greedy algorithm, prove that the genetic algorithms are suited to this kind of problems. The supplimentary advantages of the genetic algorithms additionally recommend it; a genetic algorithm is easy to implement, easy to extend and easy to parallelize in order to gain computational efficiency.

Altogether, the considered problem analysis and also the genetic solving perspective remain valid for all the usual optimization problems in economics.

References

1. Dorigo, M., *Optimization, Learning and Natural Algorithms*, Ph.D. Dissertation (in Italian), Politecnico di Milano, Italy, 1992.
2. Feo, T., Resende, M., A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters*, 8, 1989, pp. 67–71.
3. Fletcher, R., *Practical methods of optimization* (2nd ed.), New York, John Wiley & Sons, 1987.
4. Glover, F., Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research,* 13, 1986, pp. 533–549.
5. Holland, J., *Adaptation in Natural and Artificial Systems*, MIT Press, The University of Michigan Press, Ann Arbor, 1975.
6. Karaboga, D., An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

7. Kennedy, J., Eberhart, R., Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942-1948.
8. Luke, S., *Essentials of Metaheuristics*, second edition, Lulu, Department of Computer Science, George Mason University, 2013.
9. Mladenović, N., Hansen, P., Variable neighborhood search, *Computers and Operations Research* 24 (11), 1997, pp. 1097–1100.
10. Nicoară, E.S.,, Population-Based Metaheuristics: A Comparative Analysis, *International Journal of Science and Engineering Investigations* (IJSEI), 1(8), 2012, pp.88-92.
11. Nicoară, E.S., *Metaeuristici*, Petroleum – Gas University of Ploieşti Publishing House, 2013.
12. Polak, E., *Optimization: Algorithms and Consistent Approximations,* Springer-Verlag, 1997.