# Applications of Genetic Algorithms

**Peter Schreiber**
Institute of applied informatics, automation and mathematics
Faculty of Materials Science and Technology
Slovak University of Technology in Bratislava
Paulinska 1, 917 24 Trnava, Slovakia
Peter.schreiber@stuba.sk

**Abstract**. *Genetic algorithms are well-known, simple and powerful optimizing method, inspired by Darwin's evolutionary theory. They are used in many areas. Their advantage is that they do not need a mathematical description of the optimized problem. Therefore they are good alternative to mathematical-based methods, if these have failed.*

*More applications of genetic algorithms have been designed and implemented in the author's organization. Some of them are described in this contribution: a system for production optimization in production systems, a system for planning of robot's optimal trajectory and a system for faces (identikits) design by genetic algorithms.*

**Keywords.** Genetic algorithms, optimization, robot, production system, identikit

# 1 Introduction

Analytical optimizing methods are not usable in many cases. Just then some alternative optimizing ways should be used. Genetic algorithms offer a modern and unconventional solution mode.

The genetic algorithms imitate the natural evolution processes and they take advantage of them for the optimizing tasks solution. The principle of genetic algorithms bases on the fact, that individuals of a population (individual = solution of the task), which are more successful (better) then others, go successfully to next generation (generation = computing iteration). In the next generation they can be changed (improved) within genetic operations. The main genetic operations are mutation and crossover operation.

The genetic optimizing algorithms are simple, but robust. They employ stochastic processes and they are able to find a global extreme without deadlock in a local one. They are not restricted by parameters number or by limiting conditions. They can solve tasks with combinatory rising solutions space. Therefore they are applicable for solution of various difficult problems, where analytical approaches fail.

## 1.1 Base terms

Before we discuss the applications of genetic algorithms, some base terms of their terminology should be specified.

Table1. Base terms.

| Term | Explanation |
|---|---|
| Population | A group of individuals of any defined number. Iterative computations take place in the group. Iteration = generation. |
| Chromosome | One individual in the population, i.e. one solution. Ordered set of binary, numerical or symbolic values, which represent properties of the individual. |
| Gene | Elementary chromosome item. |
| Objective function | Function for the chromosome evaluation. |
| Crossover operation | Genetic operation, where two parent chromosomes split on the same random position and change one of two parts. |
| Mutation | Genetic operation, where some gene varies in a new random allowed value. |

## 1.2 Principle of the genetic algorithm

A genetic algorithm consists generally of the next steps: [3], [4], [8]

1. An initial population of n individuals is generated. The process can be random, or

regulated - if some information about solution exists.
2. The objective function of each chromosome is evaluated.
3. Terminal condition's fulfillment (required objective function value, computation time or iterations number) is tested. If the condition is satisfied, the most successful chromosome is the asked solution.
4. Otherwise two groups of chromosomes should be chosen for the next generation. There are "a" best chromosomes in the first group. They go directly into the next generation. This "elitism" ensures that the new solution cannot be worse than the last one. The second group (called work group) consists of n-a chromosomes for the genetic operations (mutation, crossover). There are many methods, how to select chromosomes into work group.
5. The genetic operations are performed on the chromosomes of the work group.
6. The new generation from the chromosomes of the first as well as of the work group is created. The algorithm continues by the step 2.

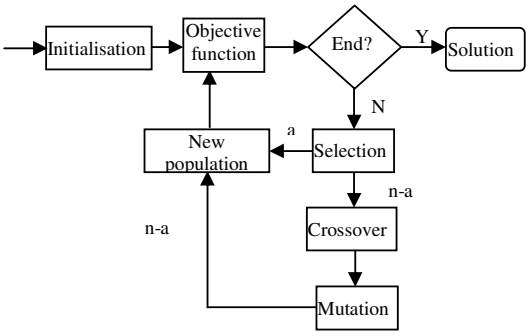


Figure 1. Base genetic algorithm

The main problem in using of GA is the task representation in GA concepts. Possible solutions should be represented as chromosomes; an objective function must be defined etc.

## 1.3 Visualization of GA

GA can be simply visualized to enable a clear understanding of GA operations. The graphical demonstration is inspired by the fact that GA works by Darwinian evolutional theory. [2],[4]

Let us suppose that solution of some technical problem is a vector with 8 numbers. This vector is one individual (e.g. a butterfly) in GA terminology. Each element of the vector represents one property of the butterfly (e.g. wings color, wings form, wings size, tentacles type, tentacles size, body size, body type, body color).

If genetic algorithm solves the technical problem (searches for 8 numbers iteratively), the vectors represent different butterflies in iterative cycles. The butterflies in various generations can be displayed and the evolution is visible.

# 2 Applications

## 2. 1 Production optimization

Let suppose that production costs that depend on the lot sizes and their input intervals should be minimized. The restrictions are flow time, capacity utilization and the number of finished parts. So :

$$N(d_1, t_1, d_2, t_2, ...,d_n, t_n) \rightarrow \min \quad (1)$$
$$T_i(d_1, t_1, d_2, t_2, ...,d_n, t_n) \leq T_{imax},\ i = 1..n \quad (2)$$
$$C_j(d_1, t_1, d_2, t_2, ...,d_n, t_n) \geq C_{jmin},\ j = 1..m \quad (3)$$
$$P_i(d_1, t_1, d_2, t_2, ...,d_n, t_n) \geq P_{imin},\ i = 1..n \quad (4)$$
$$d_{imin} \leq d_i \leq d_{imax},\ i = 1 .. n \quad (5)$$
$$t_{imin} \leq t_i \leq t_{imax},\ i = 1 .. n \quad (6)$$

where N – production costs, n – number of parts, $d_i$ – lot size of the $i^{th}$ part, $t_i$ – input interval of batch of the $i^{th}$ part, $T_i$ – flow time of $i^{th}$ part, $C_j$ – capacity utilization of $j^{th}$ part, M – number of equipment, $P_i$ – number of finished $i^{th}$ part, $T_{imax}$ – maximally acceptable flow time of $i^{th}$ part, $C_{jmin}$ – minimally acceptable capacity utilization of $j^{th}$ equipment, $P_{imin}$ – minimally acceptable number of finished $i^{th}$ parts, $d_{imin}$, $d_{imax}$, $t_{imin}$, $t_{imax}$, - limits of scanned space.

Objective function (1) and constraints (2) – (4) cannot be analytically expressed, but the simulation model built in Witness simulator determines values N, Ti, Cj, Pi. [7]

**Chromosome** (the solution of the problem) is a vector of numbers which represents input parameters of the system ($d_1$, $t_1$, $d_2$, $t_2$, ...,$d_n$, $t_n$). Its elements are genes.

**Population:** Let there are 40 individuals (solutions) in one generation.

**Objective function** is given by relation (1) and evaluates each solution. The value N is obtained by simulation.

**Selection**: Genetic algorithm requires surviving "good" solutions. They are those with small value of N. The fitness of individual solutions is inversely related to their costs. If we use roulette selection, the probability of survival is

$$p_k = \frac{F_k}{\sum_{k=1}^{g} F_k} = \frac{\frac{1}{N_k}}{\sum_{k=1}^{g} \frac{1}{N_k}} = \frac{1}{N_k \sum_{k=1}^{g} \frac{1}{N_k}} \quad (7)$$

where $p_k$ is selection probability of $k^{th}$ solution, $F_k$ is fitness of $k^{th}$ solution, $N_k$ are costs of $k^{th}$ solution obtained by simulation, g is the number of solutions in population (40).

**Elitisms**: The best 4 solutions turn to the next generation unchanged to keep the best solutions. The other 36 solutions are crossovered and mutated.
**Crossover operation**: one-point crossover of two individuals in randomly chosen position. Let $p_c = 1$.
**Mutation**: Mutation deals with probability $p_m = 0.05$.
**Computation termination**: Number of generations or calculation time is the ending condition.

The form of genetic algorithm is following:

1. Random solutions - vectors ($d_1$, $t_1$, $d_2$, $t_2$, ...,$d_n$, $t_n$) are generated, the generated values should fulfill conditions (2) - (6).
2. The solutions are evaluated by simulation in Witness.
3. If the ending condition is fulfilled, the best individual is the desired solution.
4. Otherwise the best 4 solutions are chosen into new generation. The serial numbers 1 – 4 will be assigned.
5. Remaining 36 individuals in new generation will be obtained in the following way:
   a. Roulette selection (according to relation (1)). 36 individuals are given to the positions 5 – 40.
   b. Neighboring pairs crossover. Conditions (2) – (4) are checked by the simulation. If the solution is inadmissible, another random crossover point is used.
   c. Some genes are chosen for mutation in solutions 5 – 40. New genes have to fulfill conditions (5) - (6) and new solutions have to fulfill conditions (2) – (4).
6. New generation is formed by solutions gained in steps 4 and 5. Algorithm continues by the step 2.

The results of this optimization are comparable or better than the values obtained by embedded optimizers of Witness.

## 2. 2 Robot's trajectory planning

The motion of robotic arm, autonomous robot (or robotic systems generally) can be optimized considering various criteria: travel time, energy consumption, length of trajectory and/or other. Let us use the trajectory length as criterion.

In the next text we do not consider the construction and the architecture of robots. We only suppose, that every point of robot's working space is reachable (that is there are minimal 3 degrees of freedom). Each desired position can be reached thru joints rotation and/or translation movement.

The trajectory between two points $X_{start} = X_0$ and $X_{goal} = X_n$ of robot's working space is a sequence of abscises $X_0X_1$, $X_1X_2$, ... , $X_{n-1}X_n$.

Only starting and goal points of movement must be given. The control system estimates then the shortest (optimal) trajectory.

The points number can be chosen according to accuracy of computation. The given open polygon represents the trajectory from starting point $X_0$ to goal $X_n$. The trajectory length is a sum of Euclidean distances between neighboring points. This length should be minimal.

$$\sum_{i=1}^{n} \sqrt{(X_i - X_{i-1})^2} \rightarrow \min \qquad (8)$$

There are constraints $P_i$ in the robot's working space M. $P_i$ are geometrical objects (sets of points) given by analytical description:

$$P_i = \{X;\ g_{ji}(X) \leq 0\}, \text{ where } i = 1\ldots k,\ j_i = 1...m_i \qquad (9)$$

The number of objects is k and the object i is described by $j_i$ inequalities.

$$P = \bigcup_{i=1}^{k} P_i \qquad (10)$$

is then the union of all constraints. Points from P represent collision points, or, by another words, the trajectory can consist only of points X, where

$$X \in M \backslash P. \qquad (11)$$

There are more optimum trajectories planning ways described in literature. Dynamical models using minimum-time criterion, graph-search scheme, exhaustive search methods, full dynamic models etc.

All reported solution ways need complicated mathematical apparatus. For all that a genetic algorithm can be used for the estimation of optimal trajectory. [3], [4]

In our case the chromosome (individual) is a trajectory, i.e. a sequence of points. 100 random sequences (e.g. with 16 points) are generated and evaluated by objective function (8). Best trajectories are selected into new generation (we use 10 or 20 trajectories). The roulette-mechanism is used for the selection of the next 80 or 90 trajectories. They are two cross-overed with probability 0.8 or 0.9 and then their points (genes) mutate with probability 0.01 - 0.1. The cross-over operation means, that two trajectories exchange their parts, the mutation means, that one point of a trajectory is replaced by another one, which is random selected in an allowed space.

An example of two parent chromosomes, the better one of two children after the cross-over operation and the same chromosome after proper mutation are displayed in the next 2D-figures 2, 3 and 4 (for the transparency we used trajectories with only 6 segments (7 points)):

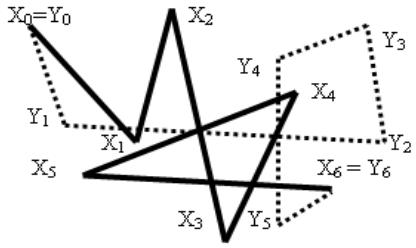


Figure 2. Two trajectories ($X_0X_1X_2X_3X_4X_5X_6$ and $Y_0Y_1Y_2Y_3Y_4Y_5Y_6$)

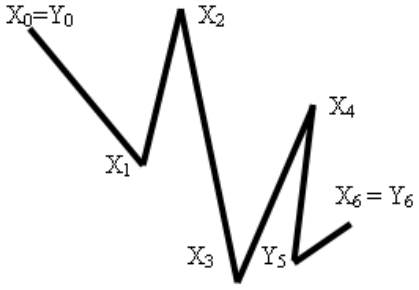


Figure 3. Descendant chromosome (trajectory) $X_0X_1X_2X_3X_4Y_5Y_6$ after crossover operation

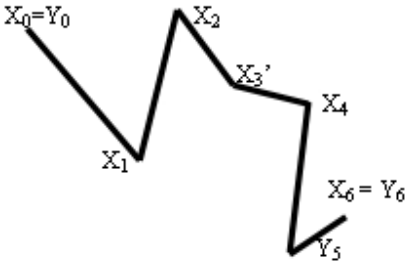


Figure 4. The chromosome after the mutation $X_3 \Rightarrow X_3'$

The sought trajectory can consist only of points, which do not belong to P. The points from P represent collisions.

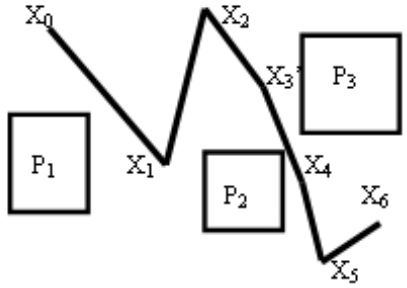


Figure 5. Working space with constraints

The genetic algorithm used in this case has some needed supplements:

1. Only points from working space, which do not belong to P, can be generated in initial population. In addition, no point from the connection of two neighboring points can be from P:

$$X_i \in M \backslash P \qquad \text{for } i = 1 \ldots k \qquad (12)$$

$$X_{i-1}X_i \cap P = \varnothing \qquad \text{for } i = 1 \ldots k \qquad (13)$$

Each trajectory is generated point to point and each new point is tested according to conditions (12) and (13). If the generated point $X_i$ does not fulfill the conditions, it must be refused and replaced by another one.
The fulfilling of condition (13) is tested by computation of intersection (common points) of abscises $X_{i-1}X_i$ and space P. The new point $X_i$ has to fulfill the condition; otherwise it must be found another one.

2. A possible collision by cross-over operation must be eliminated: There are two allowed trajectories crossovered in each operation. Segment $X_aX_{a+1}$ from one trajectory and the segment $Y_aY_{a+1}$ from the second one are allowed, but they will be spited by crossover operation and the descendants will contain segments $X_aY_{a+1}$ and $Y_aX_{a+1}$. The new segments must be controlled according to the condition (13). If one of segments intersects one of constraints, the splitting position in chromosomes must be moved in another one. If all cross-over positions are tested without success, the trajectories can not be cross-overed.
An example of a possible collision by cross-over operation is given in the figure 6. The lines (parents) $X_0X_1X_2X_3$ and $X_0Y_1Y_2X_3$ split in the same random position, e.g. between the second and the third points. The segments $X_0X_1$, $X_2X_3$, $X_0Y_1$ and $Y_2X_3$ connect into new trajectories $X_0X_1Y_2X_3$ and $X_0Y_1X_2X_3$. Even though the ancestors are allowed, the descendents can intersect possible constraints. In the figure 6 the descendent $X_0X_1Y_2X_3$ is allowed, but the descendent $X_0Y_1X_2X_3$ intersect the constraint C and the condition (13) fails.

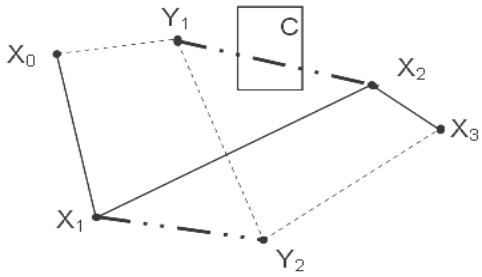


Figure 6. Collision by cross-over operation

3. The similar situation rises by mutation: By the mutation of gene $X_a$ to allowed value $X_a$' the old allowed segments $X_{a-1}X_a$ and $X_aX_{a+1}$ are replaced by new segments $X_{a-1}X_a$' and $X_a$'$X_{a+1}$, which maybe are not allowed. The condition (13) must be checked for new segments. If the condition fails, new value of mutated gene (new point of the trajectory) must be generated and relevant segments must be tested again. It repeats till new segments fulfill the condition, i.e. they do not intersect a constraint.
   An example of this situation is displayed in the figure 7. New segments $X_{a-1}X_a$' and $X_a$'$X_{a+1}$ come into existence after the mutation of point $X_a$ into $X_a$'.The segment $X_{a-1}X_a$' fulfill the condition (13), but the segment $X_a$'$X_{a+1}$ fails because the intersection with the constraint C.

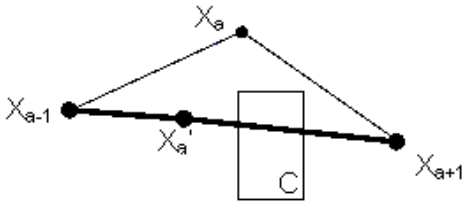


Figure 7. Possible collision by the mutation

Genetic algorithm with 3 given supplements finds the shortest trajectory in a few hundreds generations in robot's working space with constraints. The number of generations and computation time depend on the constraints number and their positions (problems with points generations and with intersections) and their analytical description (number and complexity of equations. [1], [6]

## 2. 3 Identikit design

Manual or computer methods based on a face picture composition from individual elements (for example on individual slides) are used for creation of a face image (composite drawing, identikit) in personnel identification according to instructions of witness. Disadvantage of all present-day methods is that the witness must directly adjust every single face element (eyes size, inclination, distance etc.). However he often cannot say, what is necessary to modify in order to increase the similarity between the picture and the original. The face design in this manner is neither simple nor precise.

This part of the lecture describes a solution of the face generation problem by genetic algorithms. The witness will just indicate the best solution (the most suitable face picture) from a set of displayed faces. The system will generate new faces similar to the marked one according the genetic algorithms theory. The repetition of these steps will achieve the identical or very similar face to the sought one.

The successful solution of this problem requires to find a suitable numerical or symbolic representation of faces and to formulate an objective functions which reflects the similarity.

Each face picture represents one problem solution, i.e. one individual in a population. Formally it is an ordered set of numbers, where the values represent face elements. The software system displays several (6, 9, 16) individuals (faces) and the user clicks the most similar one. He gives to the system the best solution in present generation in this manner. The system evaluates the similarity of all faces in the generation with the marked one and it finds and displays the new faces generation with the using of genetic algorithm. Because the properties (face elements) of successful individuals survive with greater probability in genetic algorithms, the new faces will be similar to the last marked one. The process continues till the user is satisfied with the reached similarity.

Each face is represented as an ordered set of numbers for our purposes. Every number means a property of one face element. Properties are e.g. ears form, size, color, position… The similar characteristics concern the eyes, eyebrow, cheeks, nose, etc., too.

We use together 33 face features. Each feature (face element) can take the value 0 - 1 with the accuracy of 4 decimal positions, i.e. every element can obtain 10 000 different values. Therefore, the total number of possible faces is $10\,000^{33}$, i.e. $10^{132}$.

The objective function evaluates the similarity of each face with the sought one. Let the numerical representation of the face "m" with n elements is ($a_{m1}$, $a_{m2}$, ... , $a_{mn}$) and let the face marked as the best one is represented by the chromosome ($b_1$, $b_2$, ... , $b_n$). User selects by click the best face in each generation from the set of displayed faces. Therefore the difference toward the base form of genetic algorithm is that the optimal solution varies in each generation.

This form of the objective function is used:

$$f_m = \sum_{i=1}^{n} | a_{mi} - b_i | \qquad (14)$$

This function expresses the sum of differences of faces elements.

The genetic algorithm has the next form for our purposes:

An initial population of 40 faces is generated.

The objective function (14) is evaluated for each face. Six faces are displayed.

User tests the terminal condition achievement (required similarity). If the similarity is sufficient, the most successful face represents the asked solution.

If the similarity is not sufficient, the user clicks on the best of the six displayed faces. Two groups of individuals are chosen for the next generation. 10 faces go directly into the next generation. The second

group (work group) consists of 10 best and 20 random chosen individuals.

The genetic operations are performed with the individuals of the work group.

The new generation of 40 faces from the chromosomes of the first as well as of the work group is created. The algorithm continues with the faces evaluation.

Parameters (genes), which represent the face elements, are used in a graphical tool, in order to display the faces. In our pilot solution we have proceeded as follows:

The tool 3D Studio Max with the module Facial studio has been used for the human heads modeling. The models are colored and three-dimensional with the possibility of rotation.

The models have been used in the system Virtools Dev. The look of faces changes in this system according to genetic parameters.

An example of displayed faces after the start of identification process is in the figure 8:

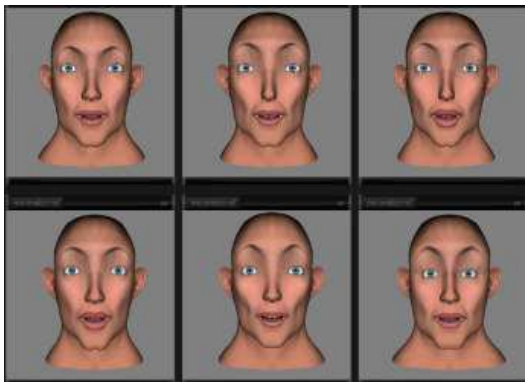

Figure 8 Displayed faces (of the 5th generation).

The described approach has confirmed some expectations, but it has brought some problems, too. The main result is that the genetic algorithms can be used for the human faces design and the genetic operations are a property approach for similar faces generation. But the main goal – automatic face design of optional person – has not been reached yet. There are more reasons:

- graphical tool is not powerful enough
- the number of face parameters is too small, a precise description of the face needs more than 33 parameters
- it is not possible to model hair, moustache, beard and birth-marks, as well as glasses, ear-rings, piercing etc. in the system
- we are not able to display an age of the person,
- the scanned area is too large (the used precision with 10 000 values for each face element is spare)

# 3 Conclusion

Genetic algorithms are simple but powerful search (optimizing) method. The obtained solutions are comparable with results reached by conventional analytical methods, but their advantage is that they do not need the mathematical model of the optimized process. Therefore GAs are usable successfully in many areas. Some applications were presented in this contribution.

# 4 Acknowledgments

The works presented in this paper have been supported by the Slovak Grant Agency of the Ministry of Education within the Project VEGA 1/0368/08 Artificial Intelligence in Flexible Manufacturing Systems Control.

# References

[1] Dalle, B: **The Robot System Movement Optimization by Genetic Algorithms**. Diploma thesis, FMST Trnava, Slovakia, 2008.

[2] De Decker A: **Visualisation of Genetic Algorithms,** Diploma thesis, FMST Trnava, Slovakia, 2008.

[3] Karr C L, Freeman L M: **Industrial Applications of Genetic Algorithms.** CRC Press, Boca Raton, 1999.

[4] Man K F, Tang K S, Kwong S: **Genetic Algorithms. Concepts and Designs**, Springer Verlag, London, 1999.

[5] Schreiber P, Otčenáš J: **Faces design by Genetic Algorithms. Aims, First Steps, Results and Problems.** Proceedings of 13th International Scientific Conference CO-MAT-TECH 2005, 20th -21st October 2005, Trnava, Slovakia, 2005, pp. 1050 – 1054.

[6] Schreiber P, Tanuška P: **Detection of the shortest robot trajectory by genetic algorithms**. DAAAM International Scientific Book 2007, Vienna 2007, pp. 191-198.

[7] Schreiber P, Važan P, Tanuška P: **Production Optimization by Using Genetic Algorithms and Simulation Model of Production System.** Proceedings of 19th International DAAAM Symposium "Intelligent Manufacturing and Automation" 22nd – 25th October, Trnava, Slovakia, 2008. (in printing).

[8] Sekaj I: **Genetic Algorithms** (in Slovak). AT&P Journal, vol. 8, no. 11, Bratislava, Slovakia, 2001, pp. 46 – 48.