Comparative analysis of the method of assignment by classes in GAVaPS

Lic. Laura Lanzarini[1], *Lic. Cecilia Sanz*[2], *Lic. Marcelo Naiouf*[3], *Ing. Fernando Romero*[4]

Laboratorio de Investigación y Desarrollo en Informática[5]
Facultad de Informática
Universidad Nacional de La Plata

Abstract

Three alternatives within the method of assignment by classes are presented for the calculation of individuals lifetime in genetic algorithms with varying population size. (GAVaPS).

In the proposed strategy (assignment by classes) individuals are grouped according to their fitness. The purpose is to use the allowed range of lifetime values in a way which is more suitable to search the optimum than proportional, linear and bilinear strategies.

A comparative study of three possibilities of assignment by classes as related to the traditional methods is carried out, and results are shown over five functions. Finally, some conclusions are presented, along with possible future lines of work.

Keywords: *Evolutive Computation. Genetic Algorithms. Genetic Algorithms with Varying Population Size*

[1] Profesor Adjunto Dedicación Exclusiva. Facultad de Informática. Universidad Nacional de La Plata. E-mail: laural@lidi.info.unlp.edu.ar
[2] Becaria de Perfeccionamiento CIC. Jefe de Trabajos Prácticos Dedicación Exclusiva. Facultad de Informática. Universidad Nacional de La Plata. E-mail: csanz@lidi.info.unlp.edu.ar
[3] Profesor Adjunto Dedicación Exclusiva. Facultad de Informática. Universidad Nacional de La Plata. E-mail: mnaiouf@lidi.info.unlp.edu.ar
[4] Jefe de Trabajos Prácticos Semi Dedicación. Facultad de Informática. Universidad Nacional de La Plata. E-mail: fromero@lidi.info.unlp.edu.ar
[5] Calle 50 y 115 1er Piso, (1900) La Plata, Argentina, TE/Fax +(54)(221)422-7707. http://lidi.info.unlp.edu.ar

Introduction

During the last few years there has been a growing interest in resolution systems based on evolution and inheritance principles. These keep a population of potential solutions, they have some selection process based on the fitness of the individuals, and they also have some genetic operators. These algorithms imitate the principles of natural evolution for parameters optimization problems [Ga98][Go98][Gr85][Mi96]. John Holland's genetic algorithms belong to this type of evolution-based algorithms. They are inspired in Darwin's Theory of Evolution [Ho75].

Genetic algorithms implementations are not always satisfactory due to different reasons:

- the codification of a problem is not suitable and thus the GAs operate over a search space which does not match the problem space.
- there is a limit over the number of generations and over the size of the population.

Therefore, under certain conditions, a premature convergence is produced towards local optimums. This is a typical problem of genetic algorithms.

The size of the population is one of the most important choices and it may be critical in different applications. If it is too small, the GA may converge prematurely, and if it is too large, waiting time for improvements can be very long, which would result in an excessive use of computational resources. On the other hand, both population diversity and selective pressure are affected by the size of the population.

Genetic algorithms begin with a set of solutions or individuals (represented by chromosomes) called population. The solutions to a population are used to generate a new population. This is done in the hopes that the new population will be better than its predecessor. In genetic algorithms with fixed-size populations (SGA) the individuals chosen to create new generations are selected according to their fitness. Those with higher fitness will have more chances of reproduction. This is repeated until a condition such as number of populations or reaching an expected optimum is verified.

Genetic algorithms with varying population size (GAVaPS) attempt to solve some of the problems produced with the SGA. They do not use any of the selection variants for individuals elimination, but they incorporate the concept of **age** of the chromosome (which depends on its fitness). This is equivalent to the number of generations that the chromosome has survived up to that moment, and it changes with generations. Consequently, the age of the chromosome replaces the concept of selection. This makes the size of the population not to remain constant but to vary as generations vary. In addition to this, another, very important concept is introduced: **lifetime**, which indicates the number of generations that an individual should survive. Lifetime is assigned only once to each individual, at the moment they are born, and remains constant during their evolution.

There are different methods to assign lifetimes. This paper analyzes different alternatives of the method of assignment by classes [La99] and its benefits as compared with the traditional methods [Mi96].

Traditional assignment strategies of lifetime in GAVaPS

GAVaPS do not use a selection mechanism, but selective pressure by means of the assignment of a lifetime to every individual. The strategy for the assignment of lifetimes should consider:

- Support individuals whose fitness is higher than the average
- Adjust the size of the current population according to a current stage in the search, mainly avoiding exponential growth.

There are three traditional approaches for the assignment of lifetime to individuals. These use some of the search state measures, such as average, maximum and minimum fitness. These approaches are the proportional, linear and bilinear assignments.

Proportional assignment.

The lifetime of the i-th individual is calculated as follows:

LifeTime(i) := min(MIN_LT + (η * fitness(i) / avgfit), MAX_LT);

where η= ½ (MAX_LT - MIN_LT)

This comes from the idea of roulette selection, where the value of the lifetime of an individual is proportional to its fitness, within the limits MIN_LT and MAX_LT.

This approach has some defects because it only uses the average fitness to characterize the current population. Thus, in order to have the maximum lifetime, an individual has to be very good (its fitness should be twice the average), which means that not all the available range of lifetime values is used suitably.

Linear assignment.

LifeTime(i) := MIN_LT + 2 * η * (fitness(i) – absminfit) / (absmaxfit - absminfit)

where η= ½ (MAX_LT - MIN_LT).

This strategy assigns a lifetime proportional to the maximum and minimum values of fitness found in the previous population. As a consequence of the improvement produced from one generation to another in the fitness of the individuals, higher lifetimes are assigned each time. This causes a great increment in the size of the population.

Bilinear assignment.

If avgfit >= fitness(i)
Then LifeTime(i) := MIN_LT + η * (fitness(i) - minfit) / (avgfit - minfit)
Else LifeTime(i) := ½ (MIN_LT + MAX_LT) + η*(fitness(i)-avgfit)/(maxfit-avgfit)

where η= ½ (MAX_LT - MIN_LT).

This method works with the average fitness and thus divides individuals in two classes. The available range for lifetimes is separated into two equal parts and, depending on the distance of the fitness of the individual from the average, the offset is determined within the corresponding segment.

Assignment by Classes

Assignment by classes is used to assign lifetimes with the following characteristics:

- to obtain a more suitable distribution within the range of allowed values, in the sense of prioritizing those individuals whose fitness is better.
- to limit the growth of the number of individuals to find the optimum
- to avoid convergence towards a local optimum
- to limit dispersion of the individuals once the "area" of the optimum has been located

In this strategy, individuals are grouped in different classes according to their fitness. To do so, a number of classes is entered, which gives the amount of different groupings that will be used. The classes are formed using the k-mean clusterization algorithm [Mar94] [Kau90] [Jai86], which is a typical unsupervised classification method with a "center" or "middle individual" for each class.

Initially, the k-means algorithm takes the k first different fitness values as centers and builds the classes with the rest of the first generation individuals, distributing them according to their shortest distance to a center. Once all the individuals are distributed, the centers are re-calculated. The algorithm continues its iterations until there are no significant changes in the centers of the classes.

For each new generation, the lifetime of the individuals is determined based on the grouping of the individuals of the previous generation.

Depending on the class to which they belong (assigned according to a minimum distance to the center) and on their own fitness value, individuals will receive their own lifetimes, since each class will have a sub-range assigned within the total of lifetimes.

Classes vary from those containing very bad individuals (low fitness) to those containing very good individuals (high fitness).

Two strategies were implemented for the selection of the lifetimes range of each class.

Strategies for the selection of the lifetimes range

a) With a fixed lifetime per class

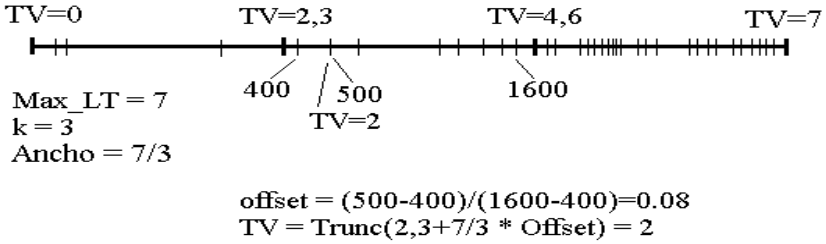
The total range of lifetimes is divided between the k existing classes. The value of the lifetime of an individual depends on its fitness and on the class to which it belongs, without taking into account the rest of the classes.

```
{LifeTime := LifeTime of Former Classes + OffsetLifeTime Closest Class}

WidthClass := MAX_LT / k
LTFormerClasses := (ClosestClass - 1)* WidthClass
LTCurrentClass  := WidthClass
Offset = ( fitness[i] - Classes[ClosestClass].MinFit ) /
         abs(Classes[ClosestClass].MaxFit - Classes[ClosestClass].MinFit))
LifeTime[i] := trunc(LTFormerClasses + WidthClass * Offset)
```

where

- *WidthClass* is the range of lifetimes assigned to each class (given by MAX_LT / Number of Classes)
- *ClosestClass* is the number of the class to which the individual belongs
- *LTCurrentClass* is the width of lifetimes of the class of the individual
- *LTFormerClasses* is the width of lifetimes of the classes previous to the class to which the individual belongs
- Classes[*ClosestClass*].MinFit and Classes[*ClosestClass*].MaxFit are fitness minimum and maximum of the class of the individual under consideration
- Fitness[i] is the fitness of the individual i.
- Offset is a number between 0 and 1 showing the displacement of the individual within a class



b) With a lifetime proportional to the amount of individuals in each class

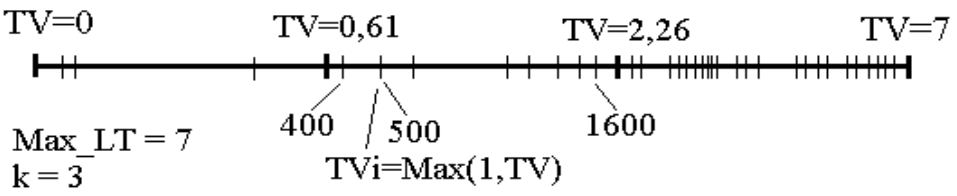
For each class, a function of the number of individuals in that class is taken as the width, so that the classes with a higher number of individuals have a greater sub-range of lifetimes to assign. This causes a selective pressure on evolution.

{LifeTime := LifeTime of Former Classes + OffsetLifeTime Closest Class}

```
FormerAmount := 0
for i:=1 to ClosestClass-1 do FormerAmount := FormerAmount+
Classes[i].IndAmount
LTFormerClasses := MAX_LT * FormerAmount / TotalIndiv;
LTCurrentClass := MAX_LT * Classes[ClosestClass]. IndAmount / TotalIndiv
Offset = ( fitness[i] - Classes[ClosestClass].MinFit ) /
        abs(Classes[ClosestClass].MaxFit - Classes[ClosestClass].MinFit))
Lifetime[i] := trunc(LTFormerClasses + WidthClass * Offset)
```

where

- Classes[ClosestClass].IndAmount is the number of individuals in the closest class
- TotalIndiv is the total number of individuals in the population



offset = (500-400)/(1600-400)=0.08
TV=Trunc(0,61+1,64*0.08)=Trunc(0,74)=0

Based on the two strategies, three alternatives were developed:

1. Assignment by equal classes: it uses the strategy posed in a).

2. Assignment by classes with greater selective pressure: it uses the strategy posed in b).

3. Mixed assignment by classes: it has a first stage where a selective pressure is exerted according to the concept in b), and a second stage that works according to strategy a). The division between one stage and the other is based on a new percentage (25%) of the number of generations.

Termination condition of the assignment by classes

The condition of termination of the algorithm is based upon the following criteria:

- ⇨ The maximum number of generations indicated in the system input parameters
- ⇨ If there are no significant changes in the optimum during a 20% of the maximum number of generations
- ⇨ If there are no significant differences in the center of the best class during the last 7 generations.
- ⇨ The distance between the centers of the two best classes. From generation to generation, the centers of the classes get closer to the optimum, and the distance between them gets shorter. Then, when the centers of the best classes of consecutive generations differ in a certain error (**ε,** system parameter) the optimum is considered to have been reached.

Results Obtained

The GAVaPS with the different assignment strategies was applied to the following functions:

$$G1: \quad f(x) = x^2 \qquad x \in [-50,50]$$

$$G2: \quad f(x) = -x\ sin(10\ \Pi\ x) + 1 \qquad x \in [-2,1]$$

$$G3: \quad f(x) = \text{int}(8x)/8 \qquad x \in [0,1]$$

$$G4: \quad f(x) = x * \text{sgn}(x) \qquad x \in [-1,2]$$

$$G5: \quad f(x) = 0.5 + \frac{sin^2\sqrt{x^2 + y^2} - 0.5}{(1 + 0.001 * (x^2 + y^2))^2} \qquad x, y \in [-100,100]$$

These functions were chosen because they cover a wide range of the kind of functions that can be optimized. Functions G2 and G5 are multimodal, with several local maximums. Function G3 cannot be optimized by means of a gradient technique (there is no gradient in one direction). Function G4 represents a problem known as deceptive problem [Go89].

For the different functions, the following parameters are considered in order to compare the different assignment strategies:

- maximum fitness reached, to evaluate the error when approaching the optimum.
- number of individuals in last generation, which gives an idea of the computational cost.
- variance within the solutions space.

On the other hand, in order to obtain the results, 30 runs of each assignment were averaged over each of the 5 functions.

Figures 1 to 5 show a comparison of assignment strategies, being **0** the **proportional**, **1** the **linear**, **2** the **bilinear**, **3** the **assignment by equal classes**, **4** the **assignment by classes with greater selective pressure**, and **5** the **mixed assignment by classes**.

As it can be appreciated in the Figures corresponding to functions G1, G2, G3, G4, G5, the strategy of assignment by classes (4) (5) and (6) presents a small error in the approximation to the optimum according to the values set in the input parameters, generally working with the lowest number of individuals and with a low variance.

In addition to this, the different experiences allowed to observe that the assignment by classes was the only method that controlled the number of individuals while searching for a solution. The remaining strategies find the optimum by "flooding" the search space with individuals, while assignment by classes finds the solution in a more directed way, trying not to overpopulate the space with bad solutions.

Of the three alternatives for assignment through classes, that with a higher selective pressure is the one that uses less individuals (as it was predictable), even though in some cases it loses some precision in the approximation to the optimum. This "error" is influenced by the value ε. The results for the mixed assignment by classes are found among the other two assignment alternatives by classes as regards number of individuals and variance, and good values for the optimum are registered. Function G5 presents a specific case for the assignment by classes, where the variance is greater than that for traditional strategies. This is due to the presence of a lower amount of individuals and to the need of genetic diversity since this is a function with a great number of local optimums.

Conclusions and future work

Three options are presented for the assignment of lifetimes by classes in GAVaPS, where there are improvements in some aspects as compared to traditional methods. These improvements concentrate the assignment of values for lifetime only in part of all the available range. The assignment by classes makes use of all the available range, making a better distinction between "good" and "bad" individuals according to their fitness.

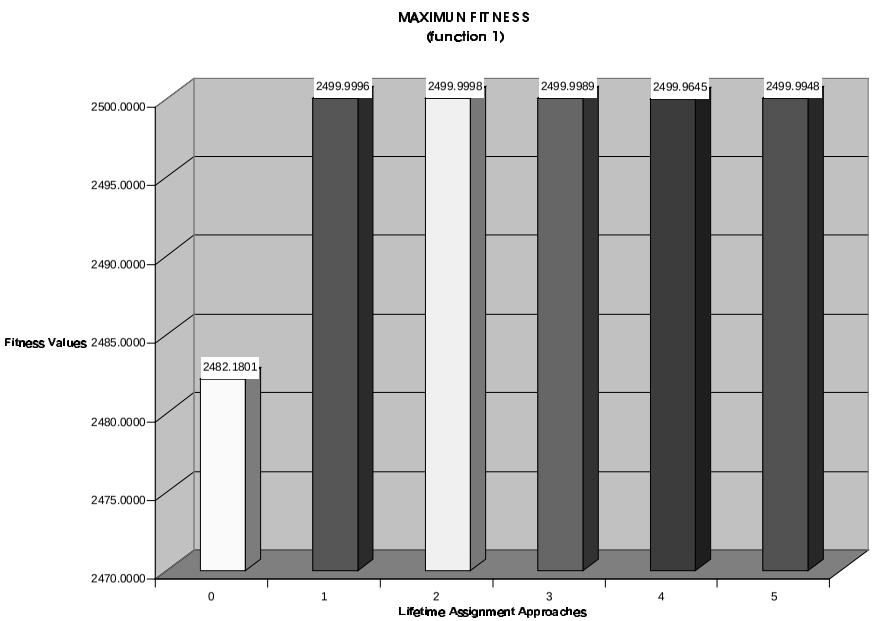
On the other hand, it can be observed that the convergence of the assignment by classes was produced by the conditions related to obtaining the optimum, whereas for the other methods, termination is caused generally when the search space is flooded with individuals.

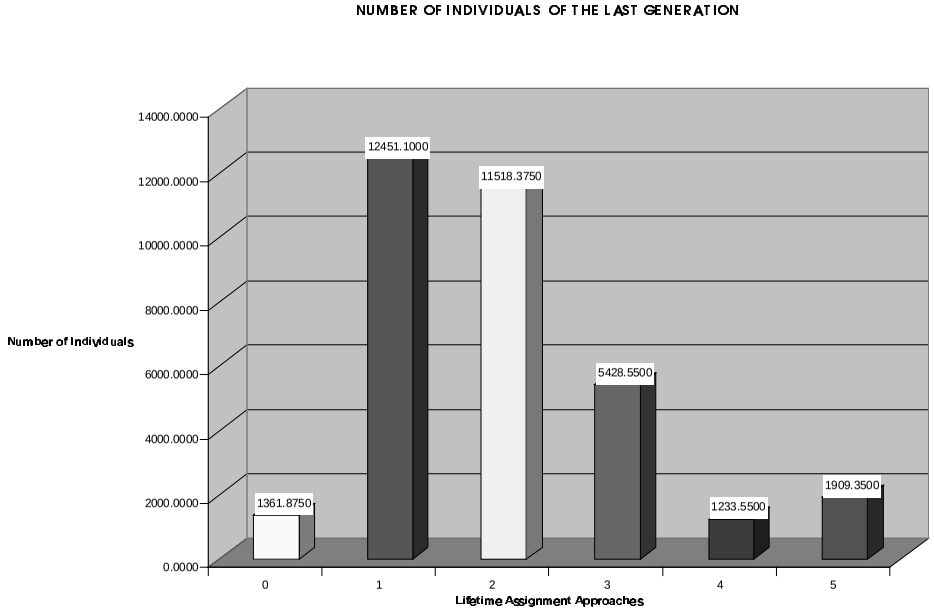
Finally, the results obtained are satisfactory based on the initial objectives.

We are currently working on a new method of assignment by classes, where the number of classes is estimated during the execution according to the current population.

As is known, the k-means method requires that the value of k be indicated beforehand. Even though the results presented here correspond to a distribution in five classes, the value of k is strongly dependent on the problem and the total range of lifetimes. Note that the number of classes is directly proportional to the selective pressure exerted. Therefore, a new method is being developed to analyze the distribution of the fitness of the individuals and creates the necessary classes accordingly.

Figure 1 – Maximum Fitness, number of individuals of the last generation, and variance found for each of the lifetime assignment methods in function G1





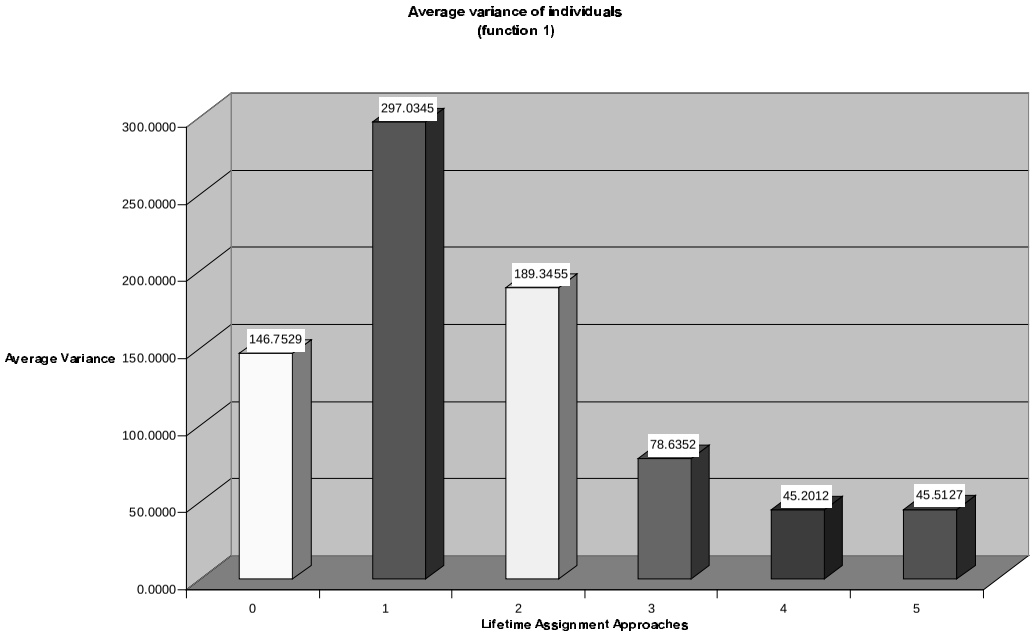
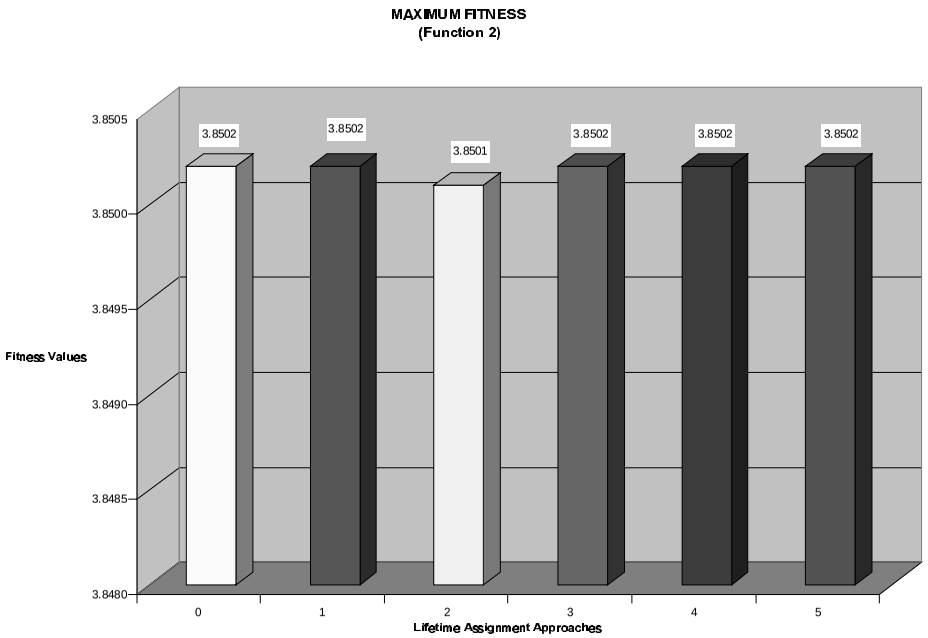
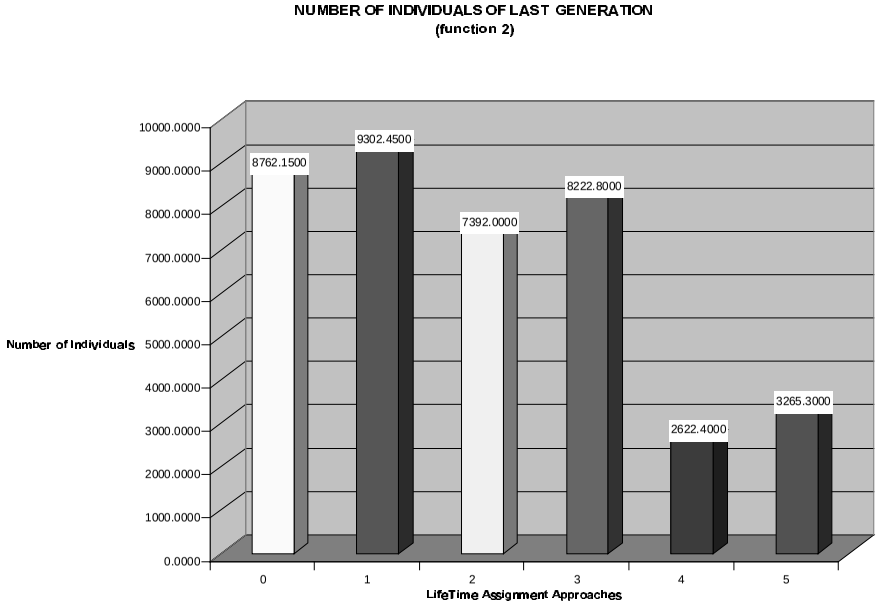


Figure 2 – Maximum Fitness, number of individuals of the last generation, and variance found for each of the lifetime assignment methods in function G2





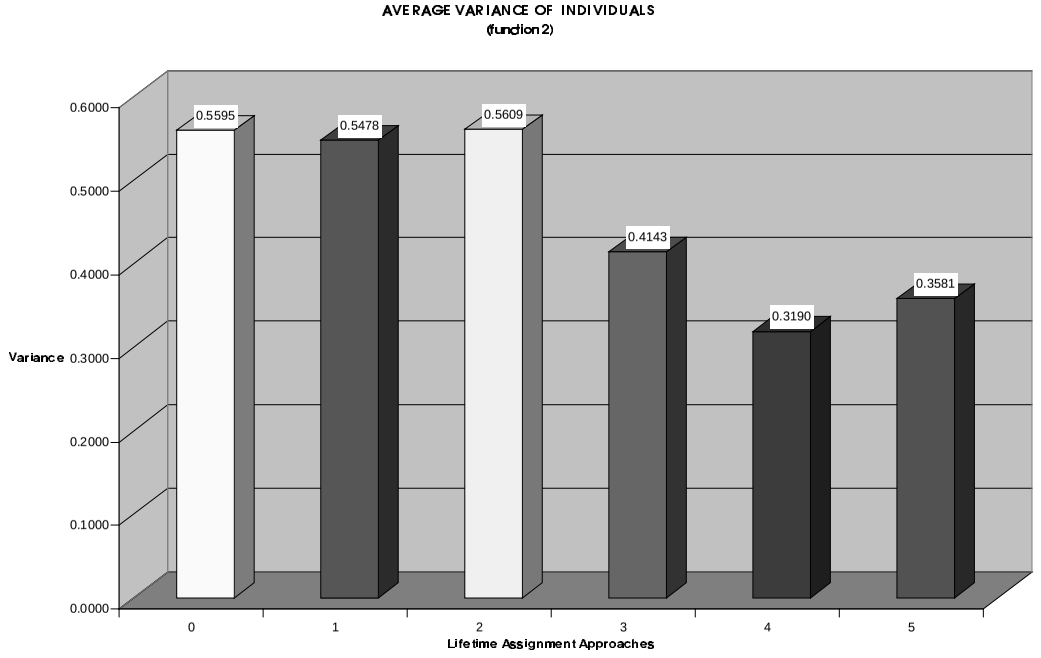
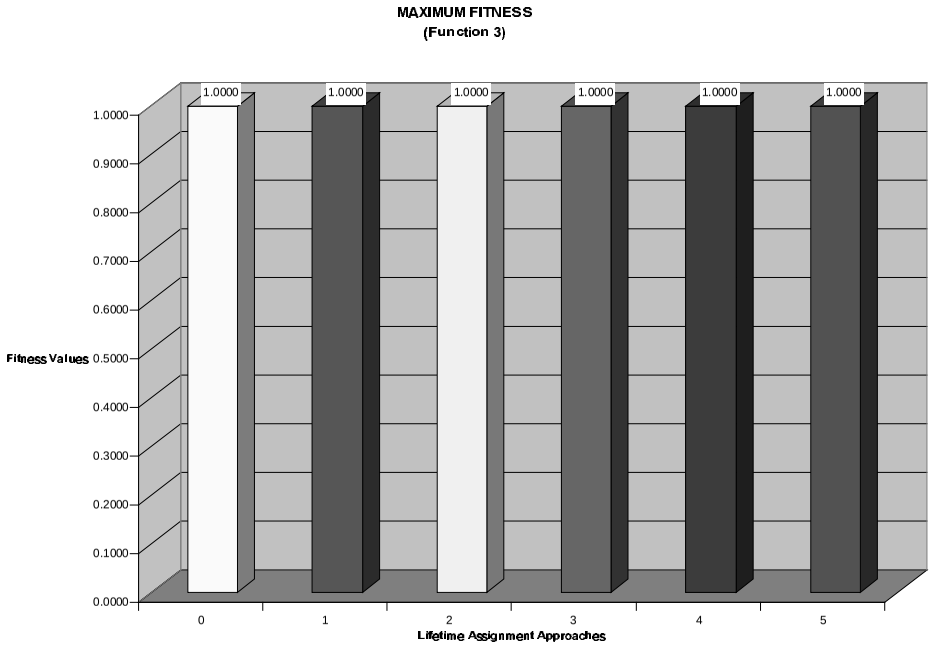
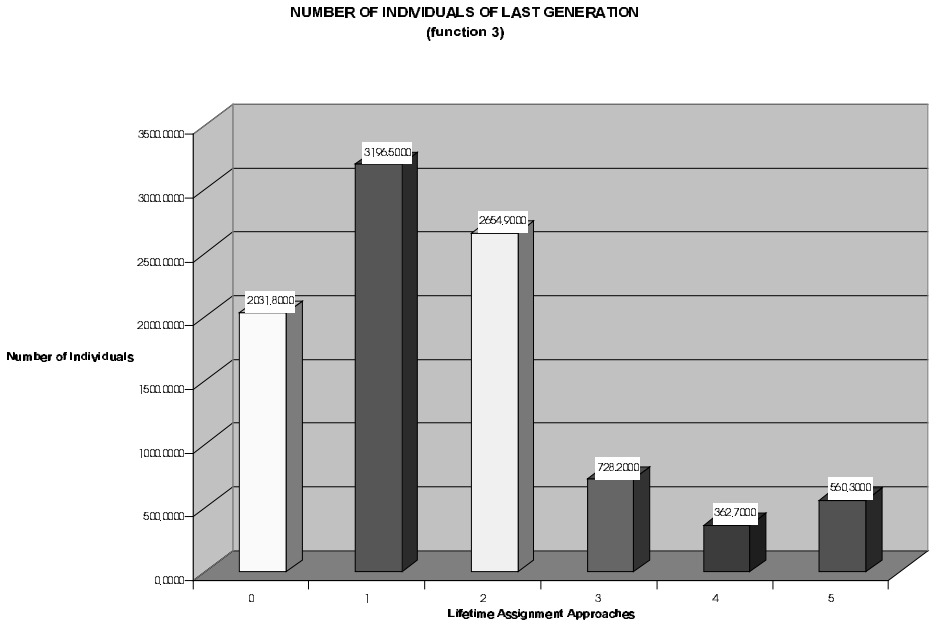


Figure 3 – Maximum Fitness, number of individuals of the last generation, and variance found for each of the lifetime assignment methods in function G3





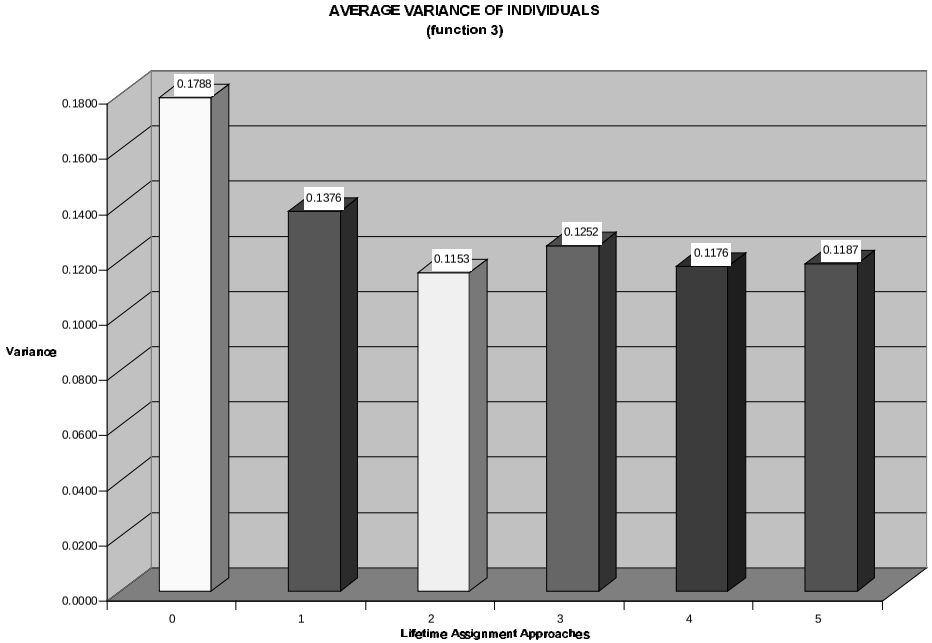
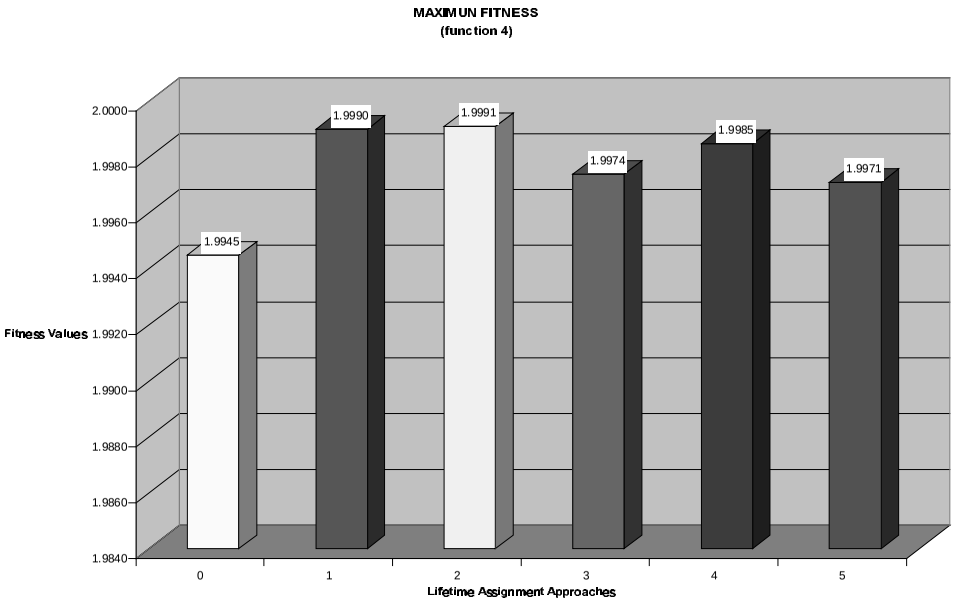
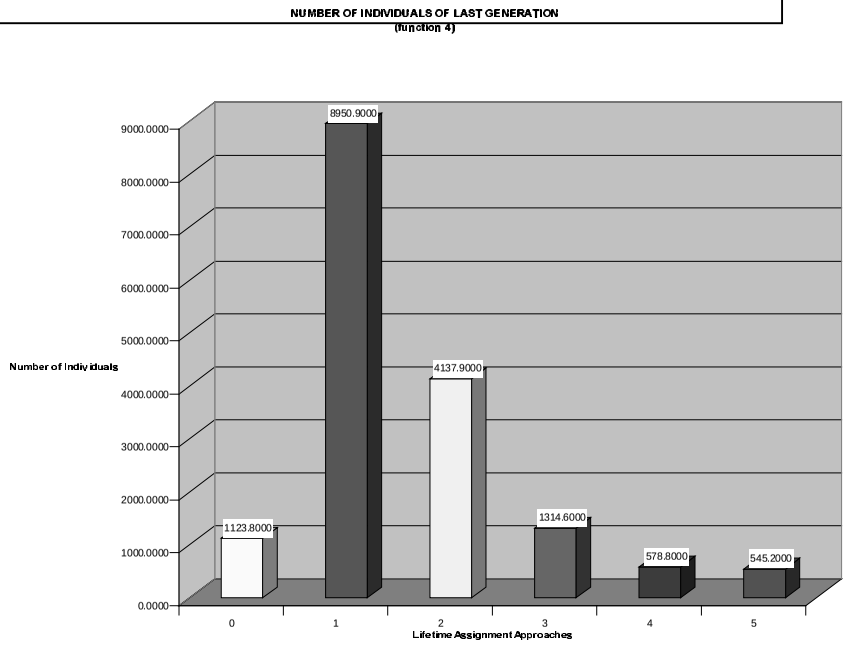


Figure 4 – Maximum Fitness, number of indivduals of the last generation, and variance found for each of the lifetime assignment methods in function G4





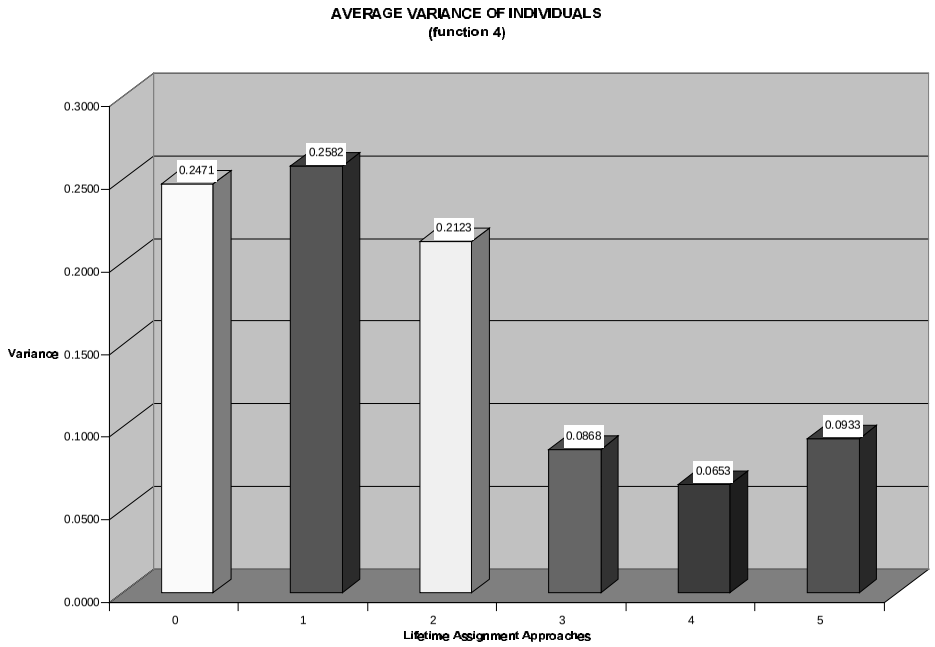
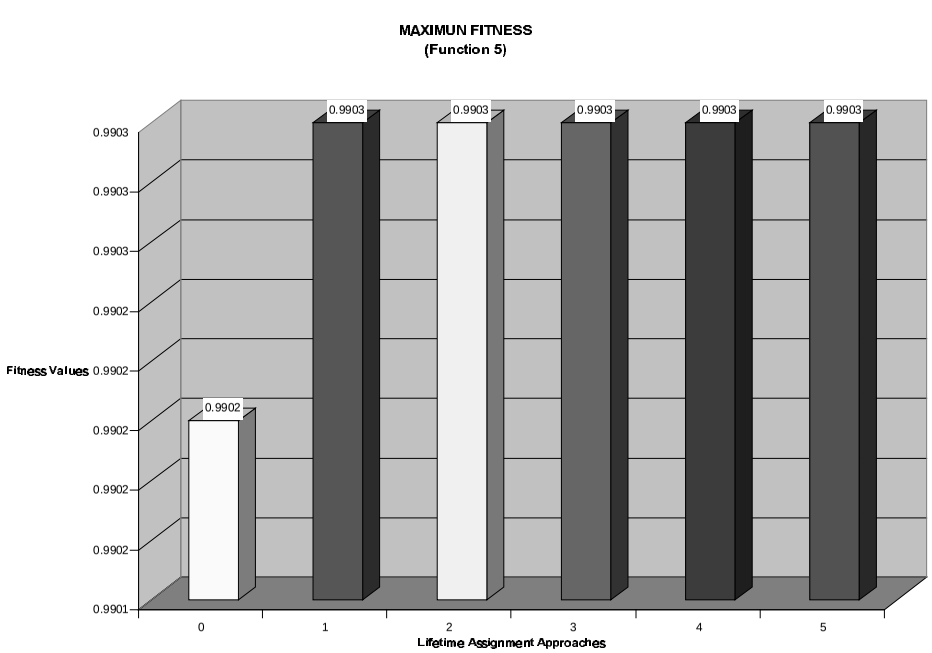
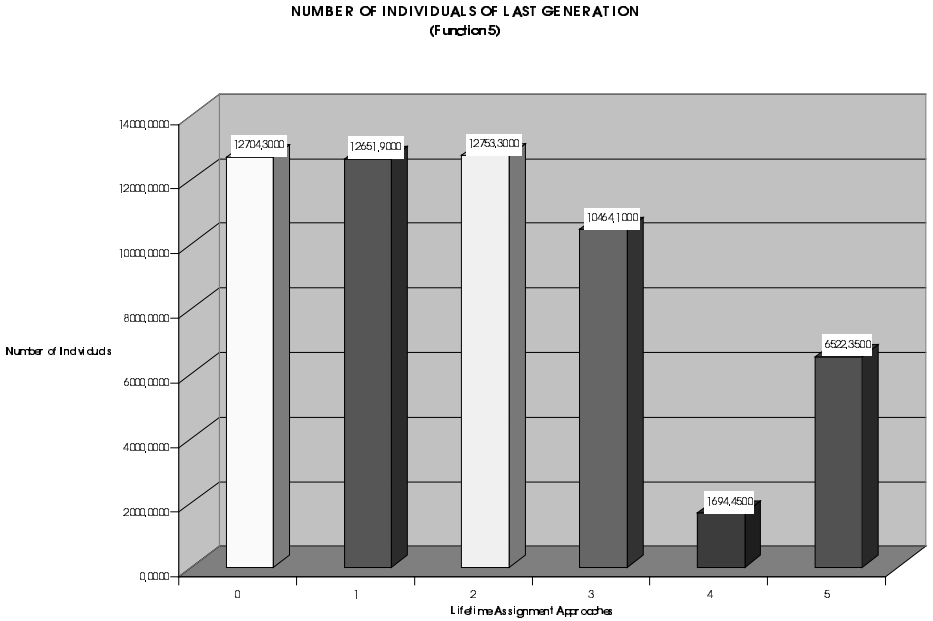
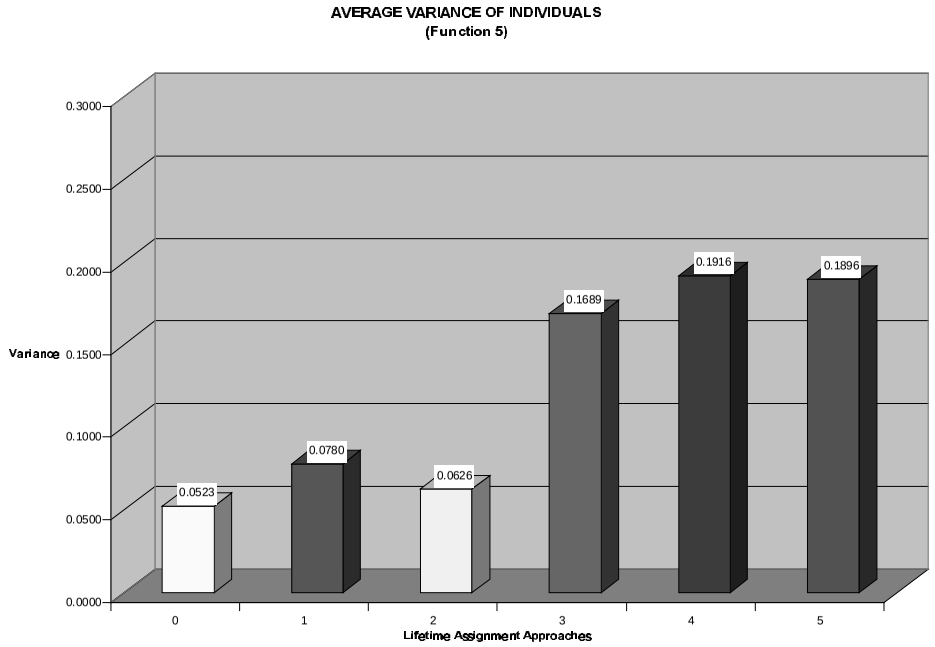


Figure 5 – Maximum Fitness, number of individuals of the last generation, and variance found for each of the lifetime assignment methods in function G5







References

[Ga98] “Técnicas de Computación evolutivas”. Raúl Gallard . Lectures of Postgraduate Course at Dpto. de Informática, Fac. Cs. Exactas. UNLP. 1998.

[Go89] “Genetic Algorithms. In search, Optimization & Machine Learning”, David E. Goldberg. Addison Wesley. 1989

[Go98] “Genetic Algorithms. In search, Optimization & Machine Learning”, David E. Goldberg. Addison Wesley. 1998

[Gr85] “Optimization of control parameters for genetic algorithms”, J.J. Grefenstette. IEEE Transactions of Systems, Man, and Cybernetics, SMC-16, pp. 122-128. 1985

[Ho75] "Adaptation in Natural and Artificial Systems", Holland. 1975

[La99] "Una nueva alternativa en los métodos de asignación de tiempo de vida en algoritmos genéticos con población de tamaño variable", L. Lanzarini, C. Sanz, M. Naiouf, F. Romero. International Congress of Information Engeenering 99.

[Jai86] "Cluster Analysis". Jain. Handbook of Pattern Recognition and Image Processing. Academic Press. 1986. Pp. 33-57.

[Kau90] "Finding Groups in data". L. Kaufman, P.J. Rousseuw. John Wiley. 1990

[Ma94] “Reconocimiento de formas y Visión artificial”, Darío Maravall-Gómez Allende. Addison Wesley. 1994

[Mi96] “Genetic Algorithms+Data Structures = Evolution Programs”. 3er Edition, Zbigniew Michalewicz. Springer. 1996