Genetic Algorithms for the Generation of Models with Micropopulations

Yago Sáez[1], Oscar Sanjuán[1], Javier Segovia[2], and Pedro Isasi[3]

[1] Lenguajes y Sistemas Department, Faculty of Computer Science, Universidad Pontificia Salamanca, Madrid, {ysaez,osanjuan}@vector-it.es
[2] Lenguajes y Sistemas Department, Faculty of Computer Science, Universidad Politécnica de Madrid, fsegovia@fi.upm.es
[3]Departamento de Informática, Universidad Carlos III Madrid, isasi@ia.uc3m.es

Abstract. The present article puts forward a method for an interactive model generation through the use of Genetic Algorithms applied to small populations. Micropopulations actually worsen the problem of the premature convergence of the algorithm, since genetic diversity is very limited. In addition, some key factors, which modify the changing likelihood of alleles, cause the likelihood of premature convergence to decrease. The present technique has been applied to the design of 3D models, starting from generic and standard pieces, using objective searches and searches with no defined objective.

1 Introduction

Evolutionary computation are learning techniques that use computational models that follow a biological evolution metaphor. In these techniques a population of individuals (solutions) evolves until the convergence criterion is reached, usually until finding a near optimal solution. The success of these techniques are based on the maintenance of the genetic diversity, for which it is necessary to work with large populations. However, not always it is possible to deal with such a large populations. There are two main areas where it becomes necessary to work with micropopulations. The first one refers to those problems where adequacy values must be estimated by a human being; and the second one relates to the problems that involve a high computational cost.

Within the first area, object of study of the present article, interactive evolutionary computation techniques can have various applications, such as the design, both artistic and functional, of bidimensional and tridimensional objects. A parallel search for solutions, allows this kind of techniques to evaluate multiple designs, thus easily generating designs that stimulate and even surpass the creativity of artists and engineers. A typical example of the application of evolutionary computation to simple figures can be seen in Biomorphs (Dawkins, 1986). It is a very simple application, where the user interactively decides which figures resemble most a real insect. In

practice, applications have been developed, such as those for the design of coffee tables, (Bentley 1999). The aim in this case is to find a table that meets certain requirements, such as, for instance, working surface, measurements, stability, lack of floating objects, etc.

Applications which make use of this technique for other practical purposes have also been developed, such as the digital treatment of images (Ngo and Marks, 1993) (Sims, 1994 a, b), the composition of musical works (Moore, 1994) and even the design of sculptures (Rowland, 2000), the automatic design of figures (F. J. Vico, 1999), the artistic design (Santos et al. 2000, Unemi, 2000), or the generation of gestures in 3D figures aimed at virtual worlds (Segovia et al. 1999, Berlanga et al. 2000).

In most of the above-mentioned references, the selection criterion is based on a merely artistic and personal point of view. In this kind of applications, problems become more complex, since the criterion used is rather subjective and, furthermore, it varies according to the user's opinions or personal preferences. This paper presents a series of modifications to the usual procedure of interactive evolutionary computation in order to be able to overcome the problem that arises when using micropopulations.

2 Environment of the Problem

The environment of the problem of interactive evolutionary computation that needs to be solved is that of the creation of a 3D design of general-purpose "tables". In this creative process the participation of a human being is needed in order to determine which tables are more creative, more functional, or which ones better fit their needs. In addition, it must be allowed to combine and modify different types of objects within the same design, i. e., the tables are complex objects generated from the genetic mutation of their various pieces.

Care will also be taken that the developed techniques shall be based on a generic design, so that they can be applied to the generation of other types of objects, such as cars, logos, and so on.

In this scheme, each table constitutes an individual of the population to be evolved, and each table is made up of various pieces. Each kind of piece that makes up the table corresponds to a different gene string. There are different types of pieces, namely, the predefined ones, based on real models, hereinafter referred to as 'standard' pieces; and the random ones, generated from cones, cylinders, cubes and spheres, referred to as 'generic' pieces. Each one of them has its own alleles, which correspond to the kind of piece -only if it is standard- material, measurement, angle, position, etc.

3 Mixed Fitness

Within interactive evolutionary computation, when working with small populations, two problems are encountered:

1. As regards individuals, it is necessary to carry out a subjective evaluation and, besides, it is rather complicated to ask the user to allocate numerical values.
2. Due to the fact that there is not a considerable genetic diversity, the algorithm tends to converge towards very different results in few generations.

The solution to these problems means changing the general scheme of the canonical algorithm. See the next figure:

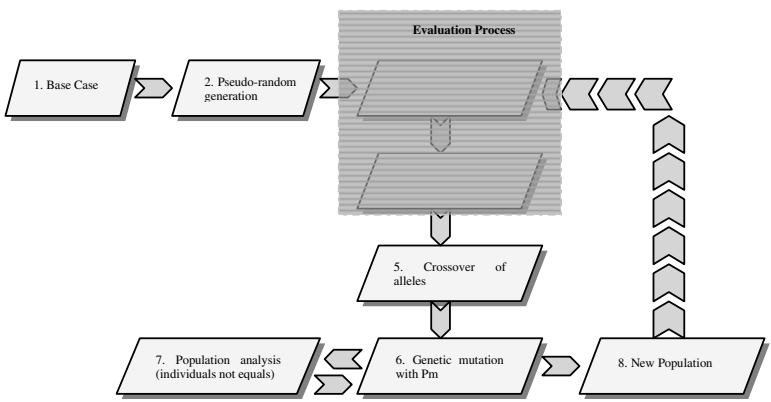


Fig. 1. Flowchart of the Genetic Algorithm that has been used.

The first above-mentioned limitation is imposed by the manner in which the user actually chooses the designs, and also by the complexity that arises when allocating a numerical value to each individual (mostly if the range is wide, or if there is a great variety).

In a genetic algorithm, an evaluation function, which causes the best ones to survive and evolve, is always used. Nevertheless, when this evaluation function depends on the user, he or she cannot be compelled to always having to provide certain fixed values to the generated individuals. It must be taken into account that the user has no knowledge on Evolutionary Computation, and therefore, when asked to supply some

adequacy values for the designs, this must be done in the easiest, most straightforward and understandable way.
In this case, it has been chosen not to give a numerical value for the evaluation of individuals, but to choose those more interesting designs, from the point of view of the user (subjetive measure). The selected individuals are going to be used as a seed for the generation of the next population.

In the application developed as an example, in order to facilitate things, the user will receive six individuals as input, from a total population, and he/she will be able to choose and classify the three that better fit his or her personal preferences, needs, and so on. As an initial option (the case base), the user can also suggest the system the rough measurements of the table he/she wants as well as its number of legs.

Fig. 2. Case base for the application

After this, the system makes a pseudo-random generation in order to create the first population that will be the seed for the evaluation process (see step 3 and 4, figure 1).

As for the second limitation detected, it has been decided to develop a combined evaluation function, where a 'mixed fitness' value is used. This value of mixed fitness is made up of two other values, an implicit fitness value, also called automated fitness (Machado 2002), which is estimated by applying some automatic evaluation functions to individuals, and an explicit fitness value, which is defined by the user.

The mixed fitness allows the widening of genetic diversity, since the population can be increased without the need to complicate the operation on the part of the user. This one will evaluate a subset of the population, which will have been previously filtered by the automatic evaluation function. The implicit fitness is estimated by a function it adds that calculates stability, the gravity centre and the contact among pieces (it is explained forward). In our example, the implicit fitness function is estimated for ten individuals generated through the use of the algorithm described in figure 1, and among these individuals, the best six ones are presented to the user, in order for he/she may evaluate the explicit fitness.
Therefore, the user will receive six individuals as input, from a total population, and he will be able to choose and classify one, two or three that most fit his or her personal tastes, needs, and so on (explicit fitness value).

If the user does not choose any individual at all, then all new genotypes will be generated through random mutation, and the process is the same as in the first generation. It can also occur that the user does not like how the population has

evolved in the last mutation, and desires to return to the previous one, or even start the whole process again.

It was found that when generating designs from standard pieces, the algorithm in very few generations had a tendency towards the same kind of tables, and this is due to the fact that when working with a limited number of standard pieces, designs are restricted to mere combinations. This problem of convergence towards the same attractor was solved firstly through the inclusion of a wide population of standard pieces and secondly making some adjustments on the reproduction mechanism, which allowed us to modify the mutation likelihood and make them be able to inherit from different parents. This measurement was considered necessary as it was observed (see Trace 1) that, during the process of objective searching, there may be situations in which the features of several individuals are of interest; for instance, the type of legs of one individual, the surface of another.

At the very moment in which the user starts the generation process of a new set of individuals, after having selected one, two or three designs through the method of classification, each gene is allocated a fitness value.

The value of mixed fitness is given by the estimation of the implicit fitness value of all elements in the population as well as of the explicit fitness value of each one of them, obtained from the user's judgement on all generated designs.

The implicit fitness value (see step 3, figure 1) is obtained after having performed the following calculations:

1. *Contact Function:* a function that evaluates the existing contact among pieces. If there is no contact among them, then the fitness value is zero.
2. *Stability Function:* a function that calculates the contact between the various pieces and the floor, by drawing a concentric circle starting from the centre of the model. The more stability, the higher the fitness value.
3. *Robustness Function:* a function that calculates the model structure by evaluating the contact that exists among pieces, the higher the contact, the more robustness.
4. *Working Surface Function:* a function that permits taking into account if a table is a practical one, by estimating its working surface.

The implicit fitness value is used to select the "n" more valid individuals and to generate the population that will be showed to the user (showing population).

4 Population Generation

The showing population is composed by six individual, among them, the user can select one, two, or three. The selected individuals(parents) are used for generating the new population. Ten new individuals are generated, less if elitism has been choosen by uniform crossover of the parents (see step 5, figure 1). If only one parent has been

selected, all the new individuals are the same. When more than one individual has been selected as a parent, each new individual is built up by selecting, in a uniform way, a gene from one of the parents.

To this last generated individuals two new operations are made. From one hand, an operator is performed in order to eliminate repeated individuals in the population (step 7). By means of this operator, each generated individual is compared with the rest, if a repeated one is founded, the individual is generated again with the same procedure as before. In the other hand, in order to avoid the premature convergence of the algorithm, and infinite cycles, a mutation operator has been included. The mutation operator is in charge of randomly altering each one of the individual's genes and this is done with a mutation likelihood of Pm. Such likelihood is a variable one and it depends on the user. The user can select it and so decide whether it is better to supply individuals in a population with a greater variety or tend towards more common features. We show it like a creativity factor.

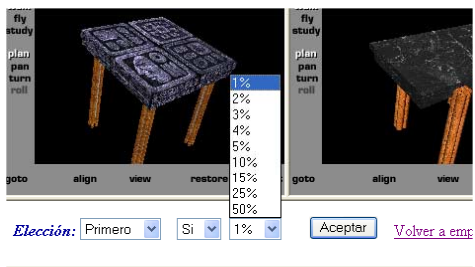


Fig. 3. Mutation likelihood

The aim of mutation is to increase the number of design primitives (Bentley, 1999), apart from avoiding the overuse, after several generations, of only one element of the alphabet, and this one being utilized in the same position for all chromosomes. This would involve that that specific feature would never change, and, therefore, it could happen that an optimal solution was never reached. However, the changing likelihood of the mutation operator must not be high, since that could damage the generation of the designs, see Figure 3. The aim, through the mutation operator, is to create elemental variations in the population and guarantee, in theory, that any point in the searching space can be reached.

During the reproduction process, genes can mutate following this possibilities:
(step 6)

1. The gene does not mutate any value.
2. The value that corresponds to the individual's location mutates.
3. The values responsible for the individual's dimensions and the materials they are made up of, mutate.
4. It mutates that part of the genes that corresponds to dimensions, materials they are made up of and location.
5. The whole gene mutates.

Then simply apply weights and weigh up according to results.

After a playing all this operators, a new population is generated and all the proccess is going to be repeated until the user decides he has founded the right design.

5 Architecture

This application is shown as a useful tool for the design of customized tables, without the need for the user to have any in-depth knowledge on computer-aided design, nor, of course, on genetic algorithms.

This architecture's design has been carried out under the premises of a system that is:

1. *Adaptable:* i.e. flexible in the design of different objects, that is, a system that is easy to modify, so that it can accomplish other kinds of designs, such as cars, motorbikes, and so on.
2. *Universal:* a system is needed that is accessible from multiple languages and different platforms; one that is easily adaptable to new technologies, such as the Internet.

In order to meet the first requirement, a logical architecture of the system has been developed, in which the different functionalities are embedded into various subsystems, trying to have the least possible cohesion among these.
The programming language C++ is a powerful and versatile language, mostly when it is combined with object-oriented methodology. And DCOM components are a standard for the development of applications. In this way it is assured that the work done can be exploited from very different applications, from a remote Web application, up to a local executable.

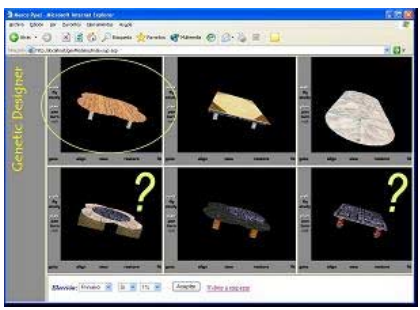
In the developed application, the User Interface has been carried out using the ASP technology of Microsoft®. VRML designs will be displayed through ASP pages and it will concurrently serve as an interface among users and DCOM components. A future objective would involve developing a Java-based architecture in order to use the benefits of X3D standard, still to be approved. From the ASP pages the COM/DCOM component is instantiated on the Web server where the Web application is being run, without the need for the client to download code, nor compile it, thanks to which we get a flexible, reliable and, most of all, platform-independent solution.

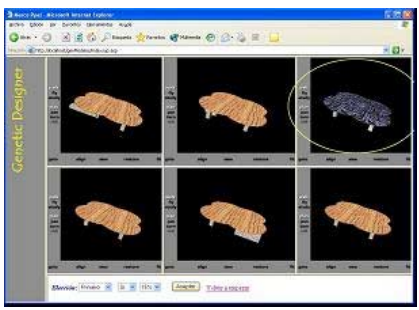
In the last version, among the improvements developed, several record files have been included which allow the system administrator to take a close look to all the events that are happening during the execution of the program and to gather information for further analysis. For the use of this application, the operation is the same as that seen in [18], the improvements have focused on the optimization of generations —through source code debugging (by freeing memory, sorting methods, etc.)— as well as on comparative studies of performance and compatibility among the various VRML

interpreters available on the market, and also, as it has been pointed out, on carrying out a permanent registration of events. It has also been developed a straightforward application in Visual Basic which accesses the component and offers some of its fuctionality from ASP pages.

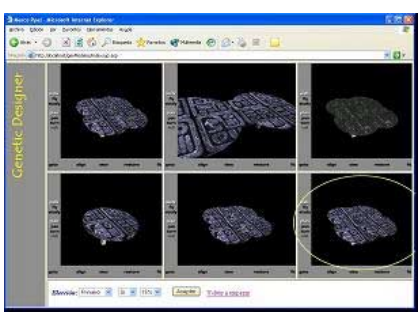
Now some traces, of how the system is actually working, are shown where it is possible to see how the design has evolved and interesting results are easily achieved.

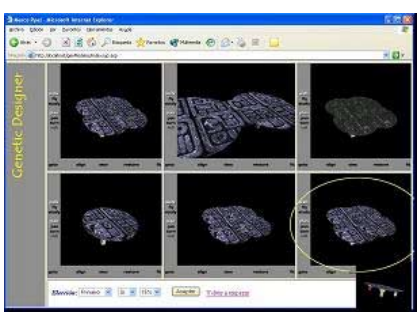
Trace 1. Objective Search with only one parent: a table for the lounge.
Restrictions of the user: Width: 1.5m Length: 1.5m Height: 0.5
After developing twenty different examples we come to the conclusion that an objective search with only one parent always converges to the same model with very few iterations. This feature makes it quite interesting if a model that is liked enough can be found.





Many cases have been detected where the first iteration has several models which are valid for an ongoing search, that is why the possibility of inheriting features of up to three models was enabled. Once that the search space has been restricted by increasing the mutation factor to 15%, we find interesting variations on the model chosen in the previous evolution stage.





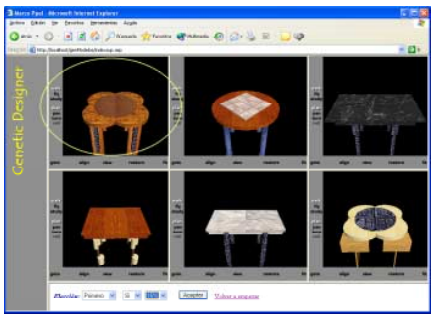
(End of Trace 1)

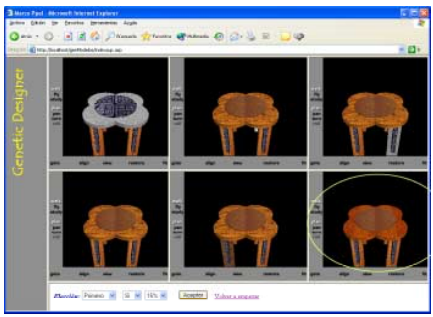
Conclusion: the system initially puts forward very different options, the user refines the features and restricts the search space with low mutation factors, until focusing on

the desired table, and then, according to its features, the user tells the system, by increasing the mutation factor, to offer variations.

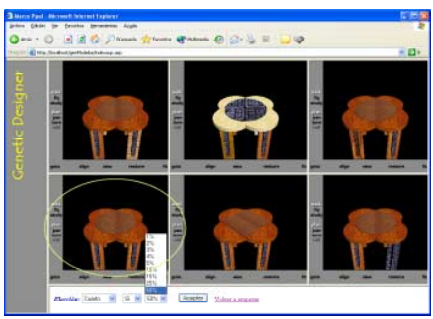
Tests were conducted starting from the first iteration with a mutation factor of 15% or more, with the clear drawback that the features that were chosen by the user could get lost due to gene mutation.

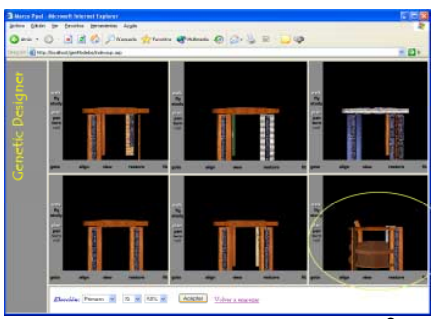
Trace 2. Non-objective Search with only one parent
With no restrictions on the part of the user.
At a non-objective search the user expects the system to provide him with models that are useful as an original idea, with an indeterminate purpose. After having conducted several tests, it is proven how the system, with low mutation factors (15% or lower) needs many iterations to provide any new ideas.

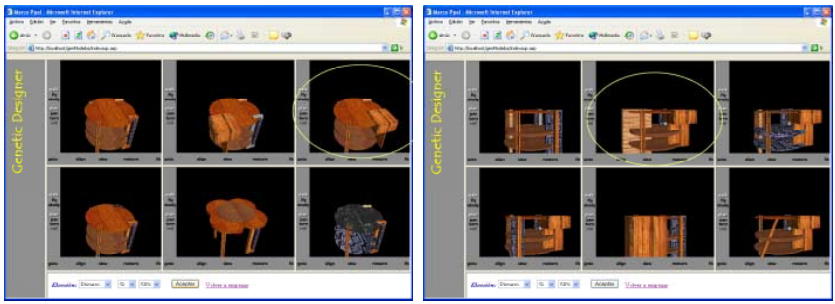




It is observed how, after three iterations, the model has hardly been changed, and the micropopulation demands new features, that is why we set the mutation factor at a 50%.
It is soon seen that the individual variety increases and how one of the proposals further motivates the user (something quite usual in non-objective searches).







(End of Trace 2)

Conclusion: in non-objective searches it is necessary to increase the mutation factor in order to have new proposals from the system starting from the models chosen by the user (creaativity factor for the user). The originality of final designs is obvious in the following example.

6 Conclusions

The restriction imposed by the number of individuals of the population, along with the fact that the user is the one that decides which individuals evolve, has made it necessary to modify the canonical genetic algorithm. The general operation scheme has been modified in order to make use of a mixed evaluation function. Such function makes a previous selection of individuals and then present them to the user, who eventually chooses after having sorted them. Furthermore, the scheme has been changed so as to enable multiparent generation, to utilize an elitist strategy and to permit the return to previous generations and thus avoiding generations with few chances of surviving. As for the reproduction mechanism it has become necessary to modify the selection and the mutation operators, with the aim of avoiding premature convergence of individuals.

This method has been tested in an application with the aim of attaining functional and creative designs; and thus the real object model as well as the generic object model have been tested; experiments with and without objective models have been conducted, with and without tracks, in the base case. After the study, both models are granted for valid, if we take into account the either creative or functional end, that we want to confer; and even very interesting results have also been obtained with the mixed intermediate model, which can combine both working models. It has been proved how the mixed fitness, combined with that modification of the selection and mutation operators favours an optimal design with the least possible number of iterations.

The improvements carried out with respect to debugging and code cleaning-up result in better response times. As for the VRML viewers tested, it is recommended the use

of the already disappeared Blaxxun, as the best one regarding performance (although a problem in texture was detected). The viewer Cortona from ParallelGraphics features a more comfortable, but less efficient interface for the surfer. Anyway, both are seen as valid for the purposes of this application.

References

1. Bentley P. (1999) *From Coffee Tables to Hospitals: Generic Evolutionary Design,* Evolutionary design by computers, Morgan-Kauffman, pp. 405–423.
2. Berlanga A., Isasi P. Segovia J. (2000) *Interactive Evolutionary Computation with Small Population to Generate Gestures in Avatars,* Proceedings of GECCO 2001, Artificial Life, Adaptative Behavior, and agents
3. Chambers L. (1995) *Practical handbook of genetic algorithms.* Vols. 1,2 editado por Lance Chambers, CRC Press.
4. Dawkins R. (1986) *The blind watchmaker,* Longman Scientific and Technical, Harlow.
5. F.J. Vico, F.J. Veredas, J.M. Bravo, J. Almaraz *Automatic design sinthesis with artificial intelligence techniques.* Artificial Intelligence in Engineering 13 (1999) 251–256
6. Holland J.H. (1975) *Adaptation in Natural and Artificial Systems,* University of Michigan Press.
7. Holland J. H. (1991) *The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance.* Proceedings of the First European Conference on Artificial Life, Cambridge, MA: MIT Press. pp.1–3, 6–7.
8. Holland J.H. (1995) *Hidden order: how adaptation builds complexity.* Addison Wesley, Reading Massachussets.
9. Moore, J.H. (1994) *GAMusic: Genetic algorithm to evolve musical melodies*. Windows 3.1 Software disponible en:http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/ genetic/ga/systems/ gamusic/0.html.
10. Ngo J.T. y Marks J., (1993), Spacetime Constraints Revisited. Computer Graphics, Annual Conference Series pp. 335–342.
11. Rowland D. (2000) *Evolutionary Co-operative Design Methodology: The genetic sculpture park.* Proceeedings of the Genetic ad Evolutionary Computation Conference Workshop, Las Vegas.
12. Santos A., Dorado J., Romero J., Arcay B., Rodríguez J. (2000) *Artistic Evolutionary Computer Systems,* Proceeedings of the Genetic ad Evolutionary Computation Conference Workshop, Las Vegas.
13. Segovia J., Antonio A., Imbert R. Herrero P., Antonini R. (1999) *Evolución de gestos en mundos virtuales,* Proceedings of CAEPIA 99.
14. Sims K., (1991) *Artificial Evolution for Computer Graphics,* Computer Graphics, Vol. 25, (4), pp. 319–328.
15. Sims K., (1994a) Evolving Virtual Creatures. In *Computer Graphics.* Annual Conference Series (SIGGRAPH '94 Proceedings), Julio 1994, pp. 15–22.
16. Sims K., (1994b) Evolving 3D Morphology and Behaviour Schemes. In Fogel, L. J. Angeline, P.J. and Back, T. *Proceedings of the 5th Annual Conference on Evolutionary Programming,* Cambridge, MA: MIT Press, pp. 121–129.
17. Unemi T. (2000) SBART 2.4: an IEC Tool for Creating 2D images, movies and collage, Proceedings of the Genetic and Evolutionary Computation Conference Program, Las Vegas.
18. Y.Sáez, O.Sanjuan, J.Segovia (2002) AEB'02 Algoritmos Genéticos para la Generación de Modelos con Micropoblaciones, Mérida, España.
19. Machado, P., Cardoso, A., All the truth about NEvAr. Applied Intelligence, Special issue on Creative Systems, Bentley, P. Corne, D. (eds), Vol. 16, Nr. 2, pp. 101–119, Kluwer Academic Publishers, 2002.