6th World Congresses of Structural and Multidisciplinary Optimization
Rio de Janeiro, 30 May - 03 June 2005, Brazil

Mooring Pattern Optimization using Genetic Algorithms

Alonso J. Juvinao Carbono [1,2]
Ivan F. M. Menezes [2]
Luiz Fernando Martha [1,2]

[1]Department of Civil Engineering and [2]Computer Graphics Technology Group (Tecgraf)
Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro - RJ, Brazil
alonso@tecgraf.puc-rio.br
ivan@tecgraf.puc-rio.br
lfm@tecgraf.puc-rio.br

1. Abstract
This paper presents the development of a Genetic Algorithm (GA) to solve the problem of the mooring pattern of floating units used in oil exploitation operations. The distribution of mooring lines is one of the factors that directly influence the displacements (offsets) suffered by floating units when subjected to environmental conditions such as winds, waves and currents. Thus, the GA seeks an optimum distribution of the mooring lines whose final goal is to minimize the units' displacements. The computation of the floating unit's static equilibrium position is accomplished by applying the catenary equilibrium equation to each mooring line in order to obtain the out-of-balance forces on the unit, and by using an iterative process to compute the final unit equilibrium position. The objective function consists of the sum of the squares of the floating unit's displacements for each set of environmental conditions. The developed GA as well as a representative example and some conclusions are also presented.

2. Keywords
Mooring Pattern, Structural Optimization, Genetic Algorithm

3. Introduction
With the increasing demand for oil, oil companies have been forced to exploit new fields in deep waters. Currently, in Brazil, Petrobras extracts about 17% of its production in shore, 19% in shallow waters, and 64% in deep waters (over a thousand meters deep). Due to the high cost of oil exploitation operations, the development of technologies capable of increasing efficiency and reducing costs is crucial.
In this context the use of floating units becomes more frequent. The positioning of the floating units during oil exploitation operations is done using mooring lines, which are flexible structures usually made of steel wire, steel chain and/or synthetic cables. The distribution of the mooring lines is one of the factors that directly influence the displacements suffered by the floating units when subjected to environmental conditions (e.g. winds, waves and currents). Therefore, the determination of an optimum mooring pattern results in an optimization problem whose final goal is to minimize the displacements of floating units.
As an optimization strategy, Genetic Algorithms (GAs) seem to be appropriate to solve this problem. They belong to the field of Evolutionary Computation and consist of heuristic combinatorial search techniques that are based on the concepts of Darwin's evolution theory [1] and genetics.
When searching for the global optimum solution for complex problems, especially those with many local minimums, traditional optimization methods fail to efficiently provide reliable results. According to Andre *et. al.* [2] the standard binary-encoded GA could constitute an interesting alternative to perform the global minimization of a function (objective function) of several continuous variables. This motivated much research to employ GAs as efficient tools for solving continuous optimization problems.
The first GAs were developed in the 1960s. In 1975, Holland's pioneering book, *Adaptation in Natural and Artificial Systems* [3], which had been originally conceived for the study of adaptive search in Artificial Intelligence, formally established GAs as valid search algorithms [1].
GAs differ from the most common mathematical programming techniques in several aspects, such as: they use a population of individuals or solutions instead of a single design point; they work on a codification of the possible solutions instead of the solutions themselves; they use probabilistic transition rules instead of deterministic operators; they can handle, with minor modifications, continuous, discrete or mixed optimization problems; and they do not require further information (such as the gradient of the objective function) [4].
One of the disadvantages of GAs is their high computational cost, due to the large number of evaluations of the objective function necessary to achieve numerical convergence. In the present work, the steady-state GA [1] has been implemented, which performs the substitution of only one or two individuals per generation [5]. The basic operators used in this algorithm are mutation and crossover.
The remaining sections of this paper are organized as follows: first, an introduction to the design of mooring systems as well as the mathematical formulation of the optimization problem are discussed; second, a brief literature review about GAs and a detailed discussion of the proposed GA are presented; next, a representative example is shown in order to demonstrate its efficiency and robustness; and finally some conclusions and suggestions for future works are made.

4. Mooring Systems Design
There are several factors to be considered when designing a mooring system, such as the type of anchorage, the strength of the mooring lines, the position of the anchors, the different sets of environmental conditions the unit will be subjected to, the seabed's shape and the time the floating unit will remain anchored.
Mooring lines are usually composed of different types of materials, each with different physical and mechanical properties. Buoys and/or clump weights may be attached to them in order to deal with the restriction imposed either by other mooring systems or by the seabed's layout.

As in most engineering problems, cost vs. effectiveness is certainly an issue in the mooring system design process. Selecting materials, defining the environmental conditions to be imposed on the floating unit and any other choices will influence directly both cost and effectiveness. A typical example of a catenary mooring system is illustrated in Figure 1.

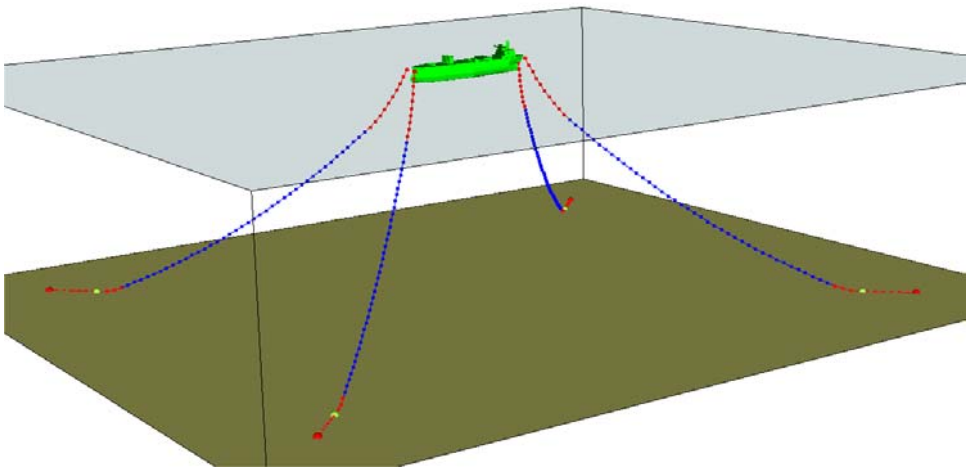

Figure 1. Catenary mooring system.

5. Formulation of the Optimization Problem

The optimum mooring pattern can be expressed as an unconstrained continuous optimization problem as follows:

$$\textit{minimize}: \quad \sum_{i=1}^{m} \Delta_i^2(\boldsymbol{\alpha}) = \sum_{i=1}^{m} [\Delta x_i^2(\boldsymbol{\alpha}) + \Delta y_i^2(\boldsymbol{\alpha})] \tag{1}$$

$$\textit{subjected to}: \quad \alpha_{i\min} \le \alpha_i \le \alpha_{i\max}, \quad i = 1 \ldots n \tag{2}$$

where $\Delta_i(\boldsymbol{\alpha})$ is the resulting floating-unit displacement (which can be decomposed into the components $\Delta x_i(\boldsymbol{\alpha})$ and $\Delta y_i(\boldsymbol{\alpha})$) for a given set of environmental conditions; $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is a vector holding the design variables (i.e. the azimuth of each mooring line); n is the number of independent design variables; m is the number of sets of environmental conditions; and the inequalities shown in Eq. (2) are the side constraints. Equations (1) and (2) represent a typical unconstrained optimization problem.

Figure 2 shows a floating unit anchored by 8 mooring lines and their distribution (α_i) to be considered in the optimization problem.

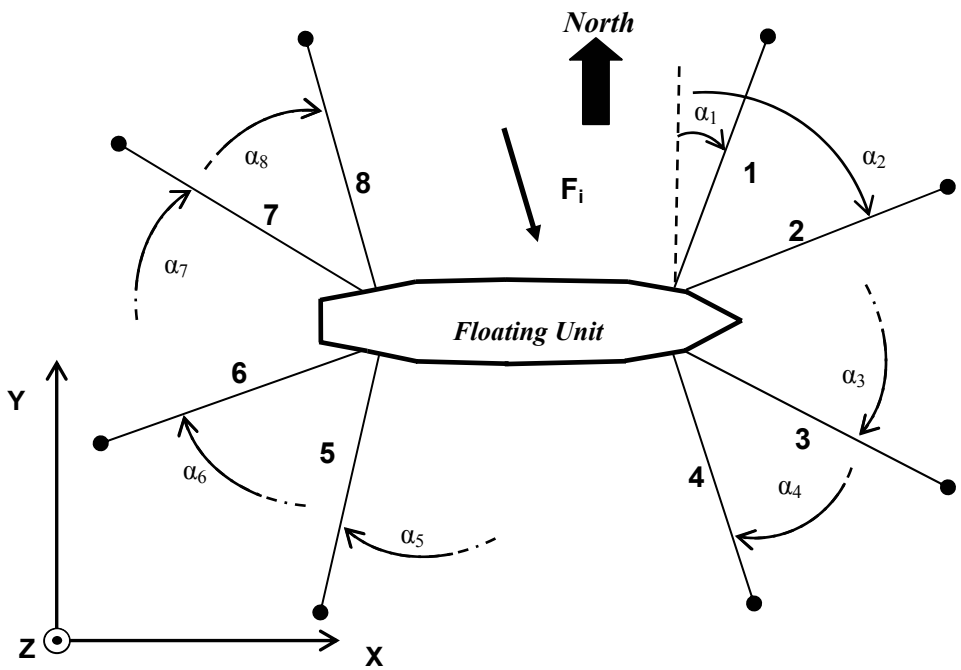

Figure 2. Representation of a mooring system with 8 lines.

This work proposes an algorithm to minimize the displacements (offsets) suffered by the floating unit when subjected to environmental conditions. Therefore, it is necessary to find an optimum mooring-line distribution in order to reduce the relative displacements Δ_x and Δ_y, as shown in Figure 3.

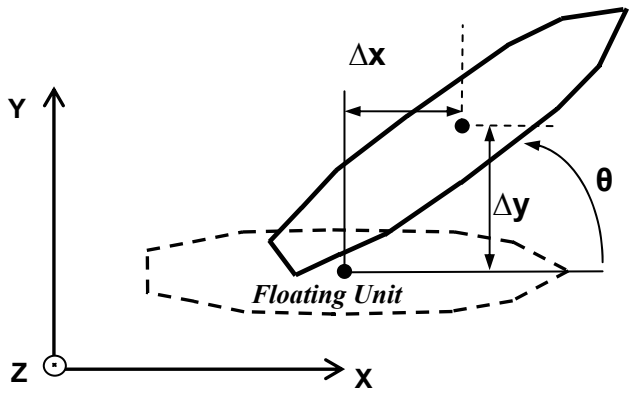

Figure 3. Displacements suffered by a floating unit when subjected to external loads.

6. Genetic Algorithms

Preliminary ideas about GAs date back to the 1960s. Considerable research has been conducted in this area. The basic concepts about GAs, used in this work, can be found in the book by Holland [3], which is based on the process of natural evolution. Holland's interests were not limited to optimization problems; he was also interested in the study of complex adaptable systems, whether biological (such as the immunity system) or not (such as the global economy).

GAs have proven to be a very versatile tool for the solution of optimization problems. They result in a very robust search mechanism for complex and discontinuous design spaces, which are very difficult or even impossible to search with traditional calculus-based methods. In general, the term "genetic algorithm" refers to any population-based model that uses several operators (e.g. selection, crossover and mutation) to evolve [6]. In a basic GA, each individual could be a real or binary-valued string, which is usually referred to as chromosome (genotype). The principles on which the GAs are based are simple: according to Darwin's theory, the selection principle privileges the fittest individuals, who have a bigger chance of surviving and, consequently, a bigger chance of reproducing. Therefore, an individual with more descendents (offspring) will have more chances to perpetuate its genetic codes to next generations. Such codes are the identities of every single individual and they are represented in the chromosomes.

Some important features of GAs are [1]: they work with a coding of the design variables (genotypes) instead of the variables themselves. They work with multiple concurrent search points, and not with a single search point. Moreover, they do not require further information as gradient of the objective function, hence the probability of getting stuck in a local minimum is reduced and consequently the search will have more chances of achieving the global minimum. They also do not require a wide knowledge of the specified design problem, which makes GAs capable of dealing with different types of problems. Finally, they use probabilistic instead of deterministic transition rules.

According to Yeh [7], a GA's basic execution cycle can be described by the following steps:

- Step 1: *Reproduction* – from an existing population of individuals a new one is created, according to their evaluated fitness function.
- Step 2: *Recombination* – to create a new population, crossover and mutation operators are applied to individuals selected randomly from a population. Crossover is an operator responsible for the propagation of the characteristics of the fittest individuals by exchanging genetic material [8]. Examples of this operator are: one-point crossover (1X), two-point crossover (2X), three-point crossover (3X) and uniform crossover, amongst others. Mutation is a reproduction operator that creates a new chromosome by modifying the value of genes in a copy of a single parent's chromosome.
- Step 3: *Replacement* – the existing population is replaced with the new one. The process of moving from the current population to the next one constitutes a generation in the evolution process. Finally, if some convergence criterion is satisfied, *stop*; otherwise, *go to* Step 1.

7. Implementation Details

In this section the main features of the proposed algorithm, together with some implementation details, are described. Several algorithms found in the technical literature [1, 8, 9] have been adapted and implemented here in order to solve the mooring pattern optimization problem. Among them, the one that performed best was the version of the steady-state GA, found in the work by Wu and Chow [1]. For this reason, the algorithm implemented in this work is called SSGA (steady-state GA).

7.1. Coding Design Variable

There are many different ways to represent the design variables, such as through binary-digit strings, floating-point representations, permutation lists, among others [1]. In this paper the design variables were coded using a fixed-length binary-digit string representation, constructed with the binary alphabet *{0,1}* and concatenated head-to-tail to form one long string. This concatenated structure represents the chromosome. Thus, every chromosome contains all design variables.

The length L of the binary string is computed as [8]:

$$L = n \times \lambda \tag{3}$$

where n is the number of design variables and λ is the number of bits needed to represent the design variables in the search space. For example, considering that a design variable x has the side constraints $8 < x < 10$, then the amplitude of the function's domain is given by: $I = 10 - 8 = 2$. With accuracy of two decimal places, i.e. $m = 2$, the interval I can be divided into equal sub-intervals S in the following way:

$$S = I \times 10^m \tag{4}$$

In this particular case $S = 2 \times 10^2 = 200$ points; thus the binary sequence must have at least 8 bits because $2^7<200<2^8$. Considering a problem with only two design variables ($n = 2$) and according to Eq. (3), the string will have a total length of $L = 2 \times 8 = 16$ bits.

7.2. Decoding

To obtain the real values of the design variables in the domain region, each chromosome must be decoded [8]. Consider, for instance, a chromosome C represented by the following string: $C = [b_7 b_6 .....b_2 b_1 b_0 a_7 a_6 .....a_1 a_0]$. First, the decimal value associated with the substring composed by the "b" elements is obtained as:

$$x_{decimal} = \sum_{i=0}^{m-1} b_i \times 2^i \tag{5}$$

Finally, the real value of the design variable within its domain is given by:

$$x = x_{\min} + x_{decimal} \times \frac{x_{\max} - x_{\min}}{2^\lambda - 1} \tag{6}$$

where $x_{\min}$ *and* $x_{\max}$ are the lower and upper bounds of the side constraints, respectively, and λ is the length of the binary string.

7.3. Offset Computing

A set of environmental conditions relevant to the mooring problem consists of waves, winds and sea currents, and it may be transformed into an equivalent external static force (F_i) acting on the floating unit. When computing the units' displacements, mooring lines are treated as non-linear springs (see Figure 4) that impose restoring forces on the units. Such forces, expressed as a function of the horizontal distance between the anchor and the connection point on the top of the line, are obtained through the restoring curves of each mooring line, which are generated using the catenary equation.

Once the out-of-balance forces are obtained, a new static equilibrium position of the floating unit is computed by solving the following system of non-linear equations:

$$\boldsymbol{K}\,\boldsymbol{d} = \boldsymbol{F}_i - \boldsymbol{R}_{int} \tag{7}$$

where $\boldsymbol{K}$ is the global stiffness matrix; $\boldsymbol{d}$ is the unknown displacement vector; $\boldsymbol{F}_i$ corresponds to the external forces due to each set of environmental conditions; and $\boldsymbol{R}_{int}$ is the resultant of the internal (restoring) forces considering all mooring lines.

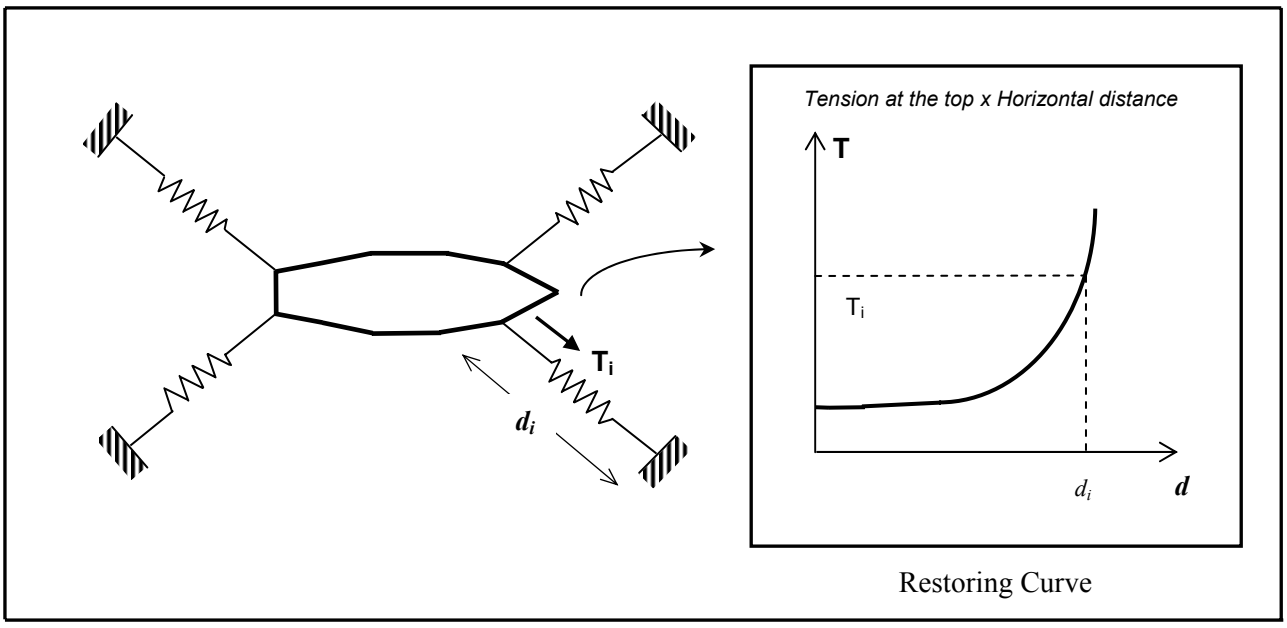

Figure 4. Representation of mooring lines by means of non-linear springs.

The solution of Eq. (7) leads to a new position of the floating unit. This computation is repeated until the resultant of the obtained displacements is smaller than a given tolerance (convergence criteria), i.e., until the unit reaches its final static equilibrium position.

7.4. Fitness Function

Fitness is a quality value that is a measure of the reproducing efficiency of individuals in a population according to the principle of the survival of the fittest [1]. This means that a chromosome with a higher fitness value will have greater probabilities of being selected as a parent in the reproduction process. Therefore, the minimization problem must be transformed into a maximization problem of a fitness function, using the following expressions:

$$F_i = 1 - \frac{\phi_i}{\phi_{\max}} \tag{8}$$

and

$$\phi_i = \frac{\Delta_i^2}{\Delta_{avg}^2} \tag{9}$$

where Δ_i^2 is the objective function (see Eq.(1)); and Δ_{avg}^2 is the average objective function value. The fitness function is normalized in order to exclude negative values.

7.5. Selection

Chromosomes are selected as parents to produce children and this selection depends on their fitness values. In this paper the ranking selection technique has been adopted in accordance with the work by Wu and Chow [1], which outperforms other selection techniques such as the roulette-wheel selection scheme. According to the authors the advantages of this selection technique are that it can prevent the domination of extra-ordinary individuals and, therefore, prevent a premature convergence of the solution; and that it can prevent wanderings among near-equals or even prevent stagnation in a population.

7.6. Crossover Operator

The standard crossover operator, which is also known as simple crossover or one-point crossover (1X) [1], has been adopted herein. In this type of crossover a crossing point is randomly chosen along the string, then the two chromosomes selected as parents exchange partial genetic material above the crossing point in order to produce new children. For example, considering the strings "*000000*" and "*111111*" and supposing that the randomly chosen crossing point is 3, the new children will be "*000111*" and "*111000*", respectively.

7.7 Mutation Operator

For binary-digit string representation, this operator changes the bit (allele) from 1 to 0 or vice versa. The mutation probability (*Pm*) must be carefully prescribed. According to Wu and Chow [1], if it is set with a low value, the algorithms often get stuck at a local minimum; but if it is high, then the algorithms will degenerate into a random search method. It is recommended to adopt a value for the mutation probability in the following range: $0.1\% < Pm < 1\%$ [8]. Mutation can not only prevent premature convergence but also restart an evolution process that has become stalled.

7.8. Generation Gap

The generation gap (G) is a parameter that controls the percentage of the population that will be replaced in each generation [10]. In traditional GAs this parameter is often set as 1.0, which means that all individuals in the population are replaced. This strategy may lead to a computational problem, due to the large number of objective-function evaluations. In addition, valuable genetic material can be lost. To avoid these problems, a small generation gap is recommended, according to Wu and Chow [1]. In the present work, the following value of G has been adopted:

$$G = \frac{2}{n} \tag{10}$$

where n is the population's size.

According to Eq. (10), only two individuals are selected to reproduce and the two new offspring replace the two worst chromosomes of the current population. Consequently, the reduction in the number of function evaluations is given by [1]: $(n-2)/n$.

8. Computational Procedures

The main computational steps of the proposed SSGA are summarized in Figure 5 [1].

1. Start

- Initialize parameters: population size, crossover and mutation probabilities, among others.

2. Seeding

- Initial population is generated randomly (binary-string digit codification)
- Initial population is decoded – Eqs.(5) – (6)
- Fitness values of each individual is computed by applying Eq.(8)

3. Reproduction

- Two chromosomes are selected as parents (ranking selection technique)
- Application of the crossover operator
- Application of the mutation operator

4. Updating

- The two new offspring chromosomes substitute the two worst chromosomes of the current population

5. Evaluation

- The two new chromosomes are decoded
- Fitness of the two new chromosomes is computed

6. Stopping criterion satisfied

If so, then go to step 7; else, go back to step 3 (Note: the stopping criterion used in this work is the maximum number of iterations)

7. Report

8. End

Figure 5. Computational steps of the proposed algorithm.

9. Numerical Example

The computational procedures, as shown in section 8, have been implemented here in a computer program (using the C language) to solve a mooring pattern optimization problem. Crossover and mutation probabilities were set as 1.0 and 0.01, respectively.

9.1. Example - Floating Unit with 18 mooring lines

A floating unit anchored by 18 mooring lines as illustrated in Figure 6 was considered; each line is composed by three different materials whose properties are shown in Table 1. The 18 lines were divided into 6 groups with the same side constraints (see Table 2).

Table 1. Material properties.

Material (from unit to anchor)	Φ (mm)	Length (m)	EA(kN)	Ww (kN/m)	Breaking Load (kN)
1 – Chain	120	250	854427	2.4580	13573
2 – Polyester	225	1500	222113	0.0859	13734
3 – Chain	120	500	854427	2.4580	13573

Table 2. Side constraints of the lines groups.

Group	Lines	Lower bounds (degrees)	Upper bounds (degrees)
1	1,2,3	0	45
2	4,5	45	90
3	6,7	90	135
4	8,9,10	135	180
5	11,12,13,14	202.5	247.5
6	15,16,17,18	292.5	337.5

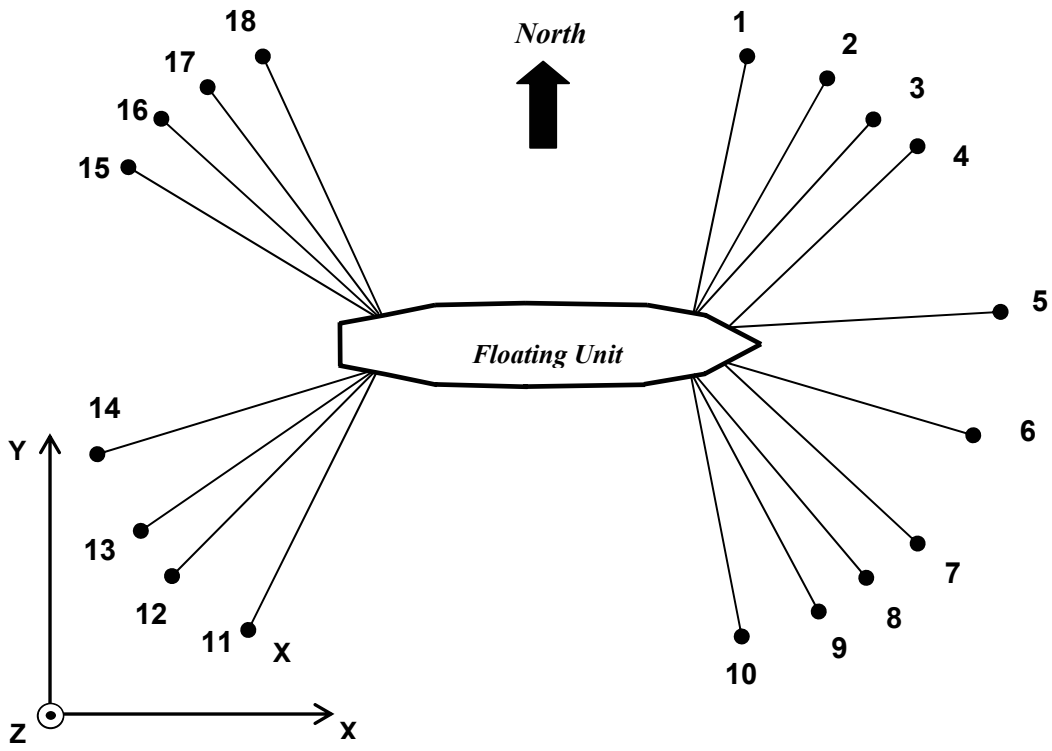

Figure 6. Floating unit with 18 mooring lines.

The floating unit was subjected to a set of environmental conditions that have been combined according to a collinear approach, i.e., with currents, winds and waves acting simultaneously in the same direction. For this example, eight combinations have been considered. For each combination, an external force of 5000 kN was applied to the floating unit in a particular direction, as shown in Table 3.

Table 3. External forces acting on the floating unit.

Angle w.r.t. X axis (degrees)	Direction
0	E
45	NE
90	N
135	NW
180	W
225	SW
270	S
315	SE

The total number of iterations adopted here was 10000 and the minimum value of the objective function (9120 m^2) was reached at iteration 3192. Table 4 presents the final azimuths of each mooring line, and the final mooring pattern is illustrated in Figures 7 and 8.

Table 4. Computed azimuths of the mooring lines.

Line	Azimuth - α_i (degree)	Line	Azimuth - α_i (degree)
1	21.5	10	158.5
2	24.5	11	225.5
3	27.5	12	228.5
4	63.0	13	231.5
5	66.0	14	234.5
6	114.0	15	305.5
7	117.0	16	308.5
8	152.5	17	311.5
9	155.5	18	314.5

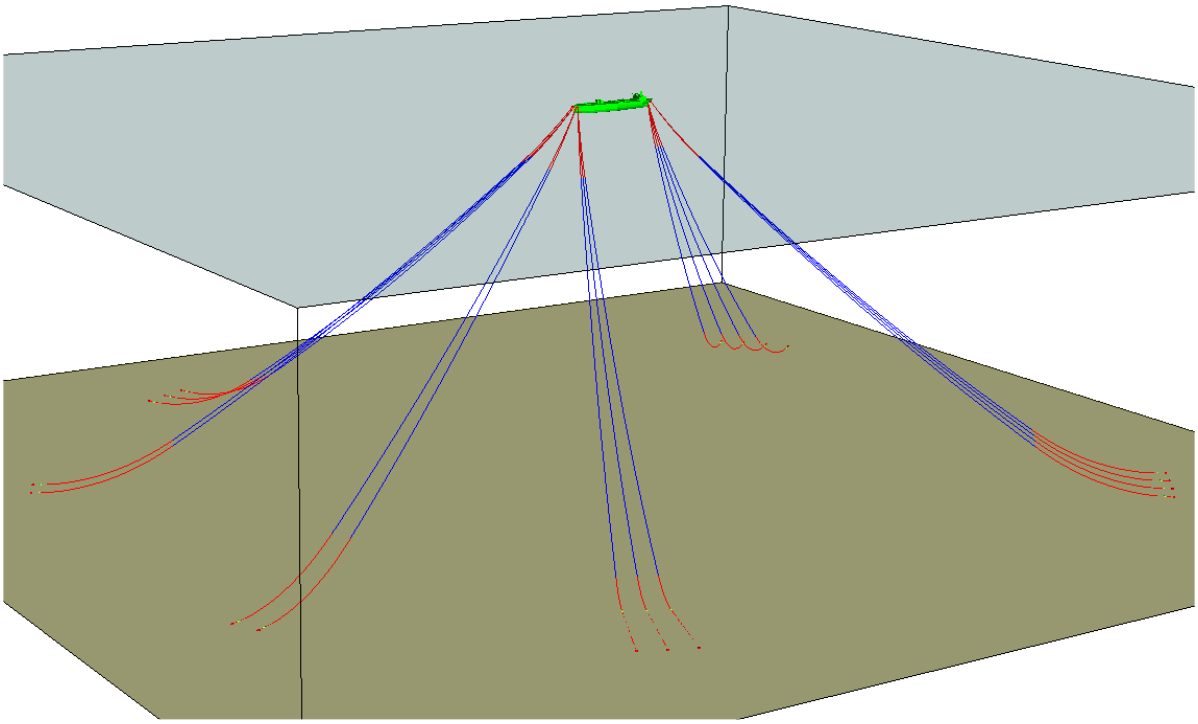

Figure 7. General view of the mooring system with 18 lines.

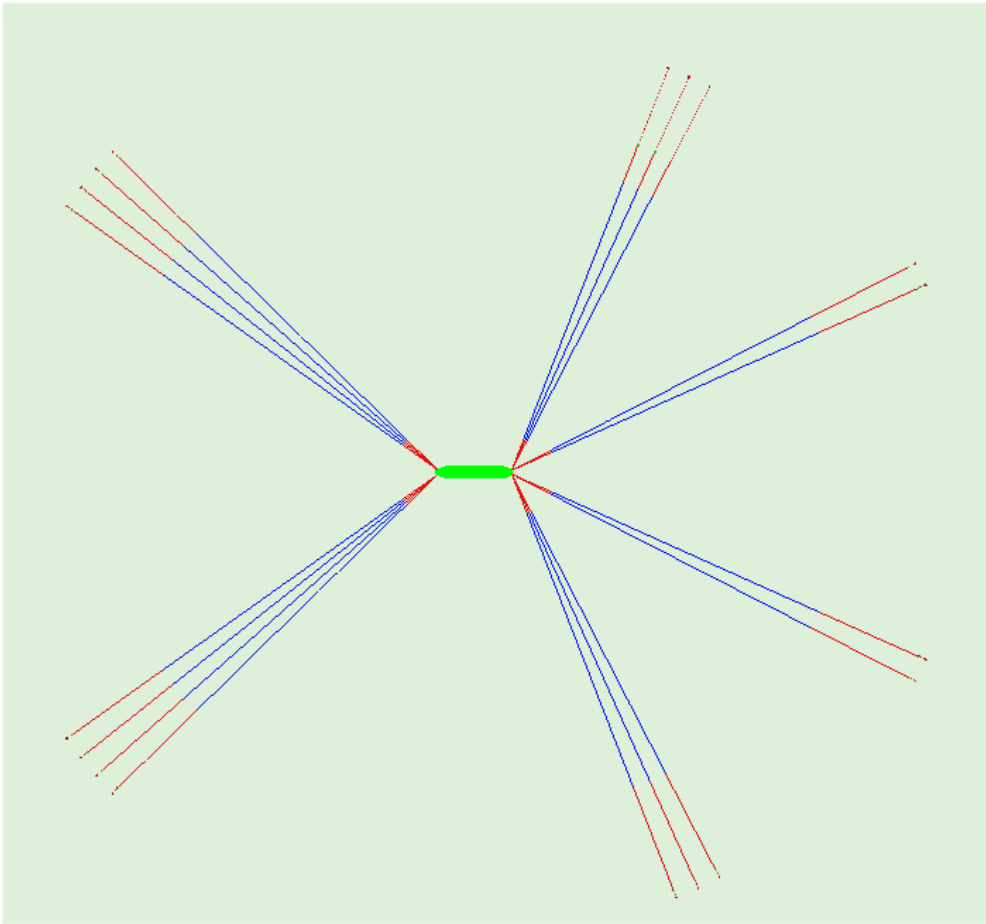

Figure 8. Final mooring pattern.

10. Conclusions

A steady-state genetic algorithm (SSGA) to solve mooring pattern optimization problems was presented. The main feature of this algorithm is the substitution of only one or two individuals per generation. The ranking selection technique was adopted. The SSGA implemented in this work had very good performance in the solution of problems with continuous variables. A representative example was provided, which illustrated the robustness and effectiveness of the implementation. An extension of this work to consider the dynamic analysis of the mooring systems to compute the displacements of floating units is currently under investigation by the authors.

11. Acknowledgements

The first author acknowledges the financial support provided by CAPES, a Brazilian agency for research and development. Most of the computation in the present work have been performed in Tecgraf/PUC-Rio (Computer Graphics Technology Group) using the Prea3D program [11]. The authors are especially grateful to the Prea3D development group for providing the resources necessary for this research, and CENPES/Petrobras for the financial support for the development of Prea3D.

12. References

1. S.J. Wu. and P.T. Chow. Steady-State Genetic Algorithms for Discrete Optimization of Trusses. Computers & Structures Vol. 56, No.6, 1995, 979-991
2. J. Andre, P. Siarry and T. Dognon. An Improvement of the Standard Genetic Algorithm Fighting Premature Convergence in Continuous optimization. Advanced in Engineering Software 32, 2001, 49-60
3. J.H. Holland. Adaptation in Natural and Artificial Systems. The university of Michigan Press, Ann Arbor, MI (1975).
4. H.J.C. Barbosa. Algoritmos Genéticos para Otimização em Engenharia: Uma Introdução aos Algoritmos Genéticos. 2ª Escola de verão em computação científica, LNCC, Rio de Janeiro, RJ, 1997
5. M. Mitchell, L.D. Davis. Handbook of Genetic Algorithms, Artificial Intelligence 100, 1998, 325-330
6. N.D. Lagaros, M. Papadrakakis and G. Kokossalakis. Structural Optimization using Evolutionary Algorithms. Institute of Structural Analysis & Sismic Research, National Technical University Athens. Computers and Structures 80, Athens-Greece, 2002, 571-589.
7. I.C. Yeh. Hybrid Genetic Algorithms for Optimization of Truss Structures. Computer-Aided Civil and Infrastructure Engineering 14, 1999, 199-206
8. F.P. Samarango. Métodos de Otimização Randômica: Algoritmos Genéticos e Simulated Annealing,. Sociedade Brasileira de Matemática Aplicada e Computacional, Rio de Janeiro, Brazil.
9. S. Rajeev and C.S. Krishnamoorthy. Discrete Optimization of Structures using Genetic Algorithms.
10. J.J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. IEEE Trans. Systems, Man, and Cybernetics SMA-16 (1), 1986, 122-128.
11. Masetti, I. Q.; Rolo, L. F.; Silveira, E. S. S.; Carvalho, M. T. M.; and Menezes, I. F. M., Sistemas Computacionais para análises Estática e Dinâmica de Linhas de Ancoragem, Proceedings of XVIII CILAMCE, 1901-1908, October, 1997.