

# Application of Cellular Genetic Algorithms and Space Efficient Chromosomes to Wells Placement in Oil Fields

Alexandre A.L. Cunha<sup>(✉)</sup>, Giulia Duncan Coutinho, Alan Porto Bontempo,  
and Marco Aurélio C. Pacheco

Applied Computational Intelligence Lab - ICA - DEE,  
Pontifical Catholic University, Rio de Janeiro, RJ, Brazil  
ashade@ele.puc-rio.br, alexandre@ashade.com.br, giuliaduncan@gmail.com,  
alanbontempo@gmail.com, marco@ele.puc-rio.br

**Abstract.** This work presents two new approaches to the optimum wells placement problem in oil fields using evolutionary methods. The presented approaches are a new space-efficient chromosome and the application of cellular genetic algorithms to guarantee healthy population diversity. Usually, authors define a domain representation having wells coordinates and types at any arbitrary position of the chromosome. On the other hand, the proposed representation enforces a unique relative wells position for each combination of wells, so, in general, there is only one problem domain representation for every concrete solution. Therefore, our representation reduces the search space, thus making the optimization more efficient. Furthermore, this paper employs a cellular genetic algorithm and presents a comparison between it and the classical genetic algorithm usually applied to this problem. The experiments with the UNISIM-I reservoir model indicate an enhancement of 6–10 times of the final *NPV* when comparing the proposed representation and the traditional one. Furthermore, the cellular genetic algorithm with the suggested chromosome performs better than the classical genetic algorithm by a factor of 1.5. These models are valuable not only for the oil and gas industry but also to every integer optimization problem that employs evolutionary algorithms.

**Keywords:** Oil field optimization · Cellular genetic algorithm · Order-aware representation · Space-efficient chromosome · Optimum wells placement

## 1 Introduction

The problem of optimizing wells placement in oil fields is essential for oil companies. Many engineering and geological variables affect the reservoir and generate complicated constraints. Accordingly, decision making is not a simple task and finding solutions to minimize cost and maximize profits is an important and challenging problem. Optimizations are a way to seek solutions for a well location, trajectory and type and automate the process.

This work deals with nonconventional wells, which are arbitrary wells regarding shape [1]. There are several works on this matter [1–6], most of them using commercial reservoir simulators in conjunction with proved optimization heuristics to determine a suitable wells alternative.

The works [1–4] use standard genetic algorithms with chromosomes that include the location of each well and their type. Both locations and types are integer genes constrained to simple domain boundaries, without any direct relationship. Furthermore, the cited works also use activation bits to denote whether a well is present in the decoded solution, thus allowing a variable number of wells.

There are possible disadvantages of these models. There is an exponential increase in the size of the search-space as the number of wells in the solution gets higher. As a result, the number of simulated scenarios required to achieve a proper solution grows too large as the maximum allowed wells count rises. Another disadvantage of the cited models is the redundancy of the chromosome model because many distinct chromosome instances decode to the same concrete implementation. Especially, any permutation of the wells within a given chromosome produces the same physical solution. Consequently, the genetic algorithm tends to fragment its population, since many distinct solutions have the same fitness, which, in turn, makes the convergence rate very slow.

Another limitation of standard genetic algorithms is the lack of control of their population diversity. As the algorithm iterates, very adapted individuals rapidly replace less adapted ones, resulting in undesirable convergence to local minima. The literature has proposed many variations of the classical genetic algorithm to solve this issue. In particular, the cellular genetic algorithm is an adaptation of standard genetic algorithm that enforces diversity by defining geographic locations of the individuals and allowing only neighbors to recombine [7].

The following sections describe a new model to solve the optimal wells placement problem using evolutionary heuristics and geophysics simulations. This model derives from the model of [3] and enforces a unique domain representation for each physical implementation. Therefore, the proposed model has less redundancy when compared to the models in the literature, and it should result in less fragmented populations with better evolution curves. Moreover, this paper applies a proved cellular genetic algorithm with online diversity control, and compare it to the standard genetic algorithms.

## 2 Problem Description

This work approaches the problem of optimizing the reservoir exploration alternative by deciding its wells locations, trajectories, and types to maximize the NPV (net present value) of the reservoir. This approach computes the NPV using the per well oil production and costs, which it obtains using a commercial reservoir simulator.

The simulations require a 3-dimensional discrete geological model of the reservoir. Therefore, the wells are placed in discrete positions, represented by the coordinates of the grid blocks. Moreover, this model represent the wells using line segments. Therefore, only the start and end blocks of the well are required.

The possible well types are injector and producer. The first type represents a well that injects water into the reservoir, whereas the latter type accounts for a well that extracts a mixture of oil, gas, and water. Additionally, the optimizer should also be able to define the optimal number of wells.

The NPV function is computed using an economic scenario specified beforehand. The simulation output is a time-step based function, having oil production and water injection amounts per time-step. Hence, for each time-step, a discount ratio is applied to calculate the NPV of that particular time-step, and finally, all the net present values are accumulated to generate the NPV of that particular wells alternative.

In addition to maximizing the NPV, we must attend to a number of restrictions involving the wells placed in the reservoir. Each well should have a length not bigger than a pre-specified value. Additionally, every pair of wells should obey a minimal separation distance. Finally, there are regions of the reservoir that should not have any wells. This work will call them no-well regions.

### 3 Methodology

This section presents a formulation of the optimization as an integer programming problem. It presents two algorithms to solve this problem: a conventional genetic algorithm and a cellular genetic algorithm.

#### 3.1 Mathematical Formulation

This model uses six integer variables to represent the discrete coordinates of two blocks wherein the well ends reside. Furthermore, the model also has one binary integer to indicate that a specific well exists and another binary integer to define the well type.

Let  $\bar{i}$ ,  $\bar{j}$ ,  $\bar{k}$ ,  $\underline{i}$ ,  $\underline{j}$ ,  $\underline{k}$  denote, respectively, the  $(i, j, k)$  coordinates of the initial block and the  $(i, j, k)$  coordinates of the final block of the well. Let  $a$  denote the activation bit and  $\tau$  denote the well type. If  $N$  is the maximum allowed wells count, then the decision vector  $\mathbf{x}$  is:

$$\mathbf{x} = \left( a_1, a_2, \dots, a_N, \right. \\ \left. \bar{i}_1, \bar{j}_1, \bar{k}_1, \underline{i}_1, \underline{j}_1, \underline{k}_1, \tau_1, \right. \\ \left. \bar{i}_2, \bar{j}_2, \bar{k}_2, \underline{i}_2, \underline{j}_2, \underline{k}_2, \tau_2, \dots, \right. \\ \left. \bar{i}_N, \bar{j}_N, \bar{k}_N, \underline{i}_N, \underline{j}_N, \underline{k}_N, \tau_N \right) \quad (1)$$

where the numeric subscripts indicate the index of the well. For simplicity, we write  $w_k = (\bar{i}_k, \bar{j}_k, \bar{k}_k, \underline{i}_k, \underline{j}_k, \underline{k}_k, \tau_k)$  and then the Eq. (1) reads:

$$\mathbf{x} = (a_1, a_2, \dots, a_N, w_1, w_2, w_N). \quad (2)$$

The objective function is defined over all values of  $\mathbf{x}$ . The relative order of the wells in Eq. (1) is not relevant, that is, the NPV depends only on the values

---

**Algorithm 1.** Boolean function  $\leq (w_1, w_2)$ .
 

---

```

1: procedure  $\leq (w_1, w_2)$ 
2:    $\triangleright w_k = (\bar{i}_k, \bar{j}_k, \bar{k}_k, \underline{i}_k, \underline{j}_k, \underline{k}_k, \tau_k)$ 
3:   if  $\bar{i}_1 < \bar{i}_2$  then
4:     return true
5:   else if  $\bar{i}_1 > \bar{i}_2$  then
6:     return false
7:   if  $\bar{j}_1 < \bar{j}_2$  then
8:     return true
9:   else if  $\bar{j}_1 > \bar{j}_2$  then
10:    return false
11:     $\vdots$ 
12:   if  $\underline{k}_1 < \underline{k}_2$  then
13:     return true
14:   else if  $\underline{k}_1 > \underline{k}_2$  then
15:     return false
16:   if  $\tau_1 \leq \tau_2$  then
17:     return true
18:   else
19:     return false

```

---

of the wells position, its types, and activation bits. Therefore, if, for example,  $\mathbf{x}' = (a_2, a_1, \dots, a_N, w_2, w_1, \dots, w_N)$  and  $f$  is the objective function, then  $f(\mathbf{x}) = f(\mathbf{x}')$ . Hence, there is a large number of points in the problem domain with the same evaluation, which might lead to many different optimal solutions that actually represent the same physical solution. Moreover, since it is interesting to apply genetic algorithms, the redundancy in the problem domain makes the search space unnecessarily large, so the algorithm tends to converge much slower to a relevant solution.

To solve this issue, the present work proposes a new strategy for representing the decision vector  $\mathbf{x}$ , where the order of the vectors  $w_1, w_2, \dots, w_N$  is unique. Therefore, for a given set of distinct wells, there is only one representation of  $\mathbf{x}$  having

$$w_1 \leq w_2 \leq \dots \leq w_N. \quad (3)$$

We specify the relation “ $\leq$ ” for any pair of wells in the Algorithm 1.

The Algorithm 1 works by sequentially comparing the coordinates of the vectors  $w_1$  and  $w_2$  until they differ or all the coordinates are compared. This model first compares the initial  $i$  coordinate, namely  $\bar{i}$ , and if  $\bar{i}_1 < \bar{i}_2$  then  $w_1 < w_2$ . Clearly, if  $\bar{i}_1 > \bar{i}_2$ , then  $w_2 < w_1$ . Finally, if the coordinates are equal, then it repeats the comparison on the next coordinate. Hence, for instance, the wells  $w_1 = (1, 0, 2, 2, 2, 2, injector)$  and  $w_2 = (1, 0, 3, 2, 2, 2, injector)$  satisfy  $w_1 \leq w_2$ . The presented model defines *injector*  $<$  *producer*.

The objective function to maximize is the NPV of the platform. This model assumes there is only one platform, and it has all the active wells. There are some models in the literature for calculating the NPV. The model from [1] uses discrete time-steps from the simulator outputs, which reports the total production of oil or gas and the total injection of water for each well and each period. After that, the authors of [1] calculate the well profits per produced or injected volumes and multiply them by the outputs of the simulator to determine the total profit of each time-step. Ultimately, the NPV is the sum of all discounted time-step profits. This model, however, considers only vertical or horizontal wells, which is an oversimplification of the problem in question. Furthermore, the problem takes into account other costs associated with the wells, as the abandonment costs, the costs depending on the wells length, the flowline costs, drilling complexity costs, and others.

The model is based in this paper in the work of [3], which models the NPV as the sum of the NPV of all wells minus the platform cost. The platform cost is the total expense of building a platform on the reservoir and it is a constant specified beforehand. Conversely, the NPV of the well is dependent upon the decision variable  $\mathbf{x}$  and considers many aspects. The Eq. (4) depicts this model.

$$NPV = \sum_{k=1}^N NPV_w(k) - C_P \quad (4)$$

In the Eq. (4),  $NPV$  and  $NPV_w(k)$  are, respectively, the platform NPV and the NPV of the  $k$ th well. Additionally,  $C_P$  is the total platform cost. The NPV of the well is the difference between the total present value of the income and the well costs:

$$NPV_w(k) = (1 - I) \cdot \sum_{t=1}^T \frac{R(k, t) - C_o(k, t)}{(1 + D)^{y_t}} - C_w(k), \quad (5)$$

where  $t$  is the discrete time,  $T$  is the number of time-steps,  $R(k, t)$  is the revenue between times  $t - 1$  and  $t$ ,  $C_o(k, t)$  is the operational cost of the well between times  $t - 1$  and  $t$ ,  $D$  is the annual discount rate and  $y_t$  is the number of years measured from the start of the reservoir operation to time  $t$ . Furthermore,  $I$  is the tax rate, and  $C_w(k)$  is the cost of the  $k$ th well.

The revenue between times  $t - 1$  and  $t$  is:

$$R(k, t) = O_p(k, t) \cdot P_o(t) + G_p(k, t) \cdot P_g(t) \quad (6)$$

where  $O_p(t)$ ,  $P_o(t)$ ,  $G_p(t)$ , and  $P_g(t)$  are, respectively, the oil production, the oil price, the gas production and the gas price between times  $t - 1$  and  $t$ . The productions are a simulation output, whereas the prices are pre-specified quantities.

The operational costs include the fixed costs of the well, the maintenance costs in the time-step, the variable costs in the time-step, the royalties, and the costs associated with the amount of fluid production or injection. The Eq. (7) shows the general equation.

$$\begin{aligned}
C_o(k, t) = & [C_M \cdot (y_t - y_{t-1})] \\
& + C_{vf} + R_y \cdot R(k, t) \\
& + \left( O_p(k, t) \cdot O_{pc} + G_p(k, t) \cdot G_{pc} \right. \\
& + W_p(k, t) \cdot W_{pc} + G_i(k, t) \cdot G_{ic} \\
& \left. + W_i(k, t) \cdot W_{ic} \right)
\end{aligned} \tag{7}$$

In the Eq. (7),  $C_M$  is the maintenance cost per year,  $C_{vf}$  is a constant cost, and  $R_y$  is the royalties percentage. Moreover,  $O_{pc}$  is the oil production cost per volume of oil,  $G_{pc}$  is the gas production cost per volume of gas,  $W_p(k, t)$  is the water production of the  $k$ th well between times  $t - 1$  and  $t$ , and  $W_{pc}$  is the water production cost per volume of water. Finally,  $G_i(k, t)$  is the gas injected between times  $t - 1$  and  $t$ ,  $G_{ic}$  is the gas injection cost per unit of gas volume,  $W_i(k, t)$  is the volume of water injected into the  $k$ th well between times  $t - 1$  and  $t$ , and  $W_{ic}$  is the water injection cost per unit of water volume.

The well development cost,  $C_w(k)$ , is a complicated non-linear function of the well length, the well position, the well inclination, and the well type. This function includes the drilling costs, the distance between the  $k$ th well and the platform, and the cost of shutting down the  $k$ th well. For simplicity, we choose to omit this function herein.

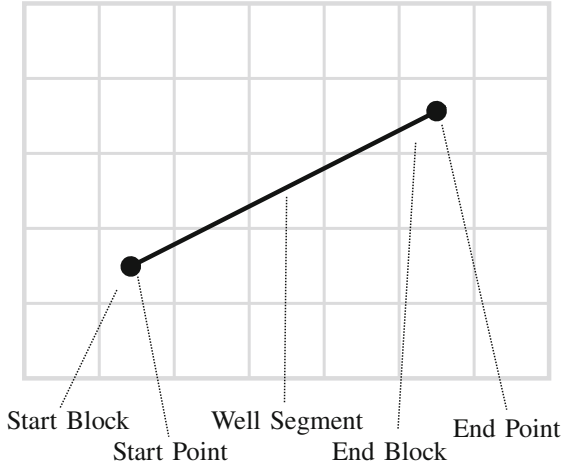
To guarantee the physical meaning of the solution  $\mathbf{x}$ , we should define suitable restrictions involving the wells length, the wells pairwise distances and the total well count. The positive integer constant  $N$  specifies the maximum number of wells the platform could handle. Since this solution use activation bits,  $a_k$ , to indicate whether the  $k$ th well exists or not, the decision variable  $\mathbf{x}$  always have  $N$  wells and the solution may have any number of wells from 0 to  $N$ .

For operational reasons, the length of each well should not exceed a maximum constant length,  $L$ . This problem models the wells as line segments whose ends are the center points of the wells start and end blocks. Therefore, the well length  $l(k)$  is simply the Euclidean distance between the line segments ends and, for each well  $k$ ,  $1 \leq k \leq N$ , we have:

$$l(k) \leq L \tag{8}$$

Similarly, there is a minimum wells distance, that is, it is not possible to place two wells closer than a minimum distance  $d_{min}$ . Hence, for each pair of wells  $k_1$  and  $k_2$ ,  $k_1 \neq k_2$ , we enforce the restriction:

$$d(w_{k_1}, w_{k_2}) \geq d_{min}. \tag{9}$$



**Fig. 1.** Representation of well in a grid.

Since the problem models the wells as line segments (Fig. 1), the distance between two wells,  $d(w_{k_1}, w_{k_2})$ , is the minimum distance between the two line segments that geometrically represent the wells  $w_{k_1}$  and  $w_{k_2}$ . In [8], the author explains this problem in detail.

In conclusion, this problem seeks the solution  $\mathbf{x}$ , as in the Eq. (2), that maximizes the value of the *NPV* in Eq. (4), subject to the nonlinear restrictions (8) and (9). The following section describes the algorithm this paper employs for solving this problem efficiently on a digital computer.

### 3.2 Solution to the Optimization Problem

The present work uses evolutionary algorithms to solve the optimization problem of the Sect. 2. In particular, this paper uses a classical genetic algorithm and a more modern approach, the cellular genetic algorithm, as the author describes in [9]. The main goal is comparing the performance of these two methods using two possible chromosomes, one that does not enforce the wells order in the representation and the proposed model that reduces the search space by ensuring a certain wells order.

The current work employs the chromosome of the Eq. (2). The fitness function is the *NPV* of the platform, as the Eq. (4) exhibits.

Both the cellular and classical genetic algorithms require mutation and recombination operators. Since there are use two possible chromosome representations, the model needs to develop these operators according to each representation.

**Algorithm 2.** Activation mutation.

---

```

1: procedure ACT_MUTATE( $\mathbf{x}$ ,  $p$ )
2:    $\triangleright \mathbf{x} = (a_1, \dots, a_N, w_1, \dots, w_N)$ .
3:    $\triangleright p$  is the gene mutation probability.
4:    $\mathbf{x}_{new} \leftarrow \mathbf{x}$ 
5:   for  $k \leftarrow 1 \dots N$  do
6:      $r \leftarrow \text{RANDOM}(0, 1)$ 
7:     if  $r \leq p$  then
8:       Flip  $a_k$  of  $\mathbf{x}_{new}$ 
9:   return  $\mathbf{x}_{new}$ 

```

---

**Algorithm 3.** Uniform mutation.

---

```

1: procedure UNIF_MUTATE( $\mathbf{x}$ )
2:    $\triangleright \mathbf{x} = (a_1, \dots, a_N, w_1, \dots, w_N)$ .
3:    $k \leftarrow \text{RANDOMINT}(1, N)$ 
4:    $i \leftarrow \text{RANDOMINT}(1, 7)$ 
5:    $\mathbf{x}_{new} \leftarrow \mathbf{x}$ 
6:    $min \leftarrow$  minimum of the  $i$ th gene of the  $k$ th well.
7:    $max \leftarrow$  maximum of the  $i$ th gene of the  $k$ th well.
8:    $r \leftarrow \text{RANDOM}(min, max) \triangleright$  new gene value.
9:   Replace the  $i$ th gene of the  $k$ th well of  $\mathbf{x}_{new}$  by  $r$ .
10:  return  $\mathbf{x}_{new}$ 

```

---

- (1) *Mutation Operators:* Mutation operates on a single individual, possibly generating a new (mutated) individual. The present model utilizes two types of mutation: the activation mutation and the uniform mutation. The former operates on the activation bits, thereby not changing the relative order of the wells in the chromosome. The latter influences the position and type genes. Hence, it is possible that a solution satisfying the order criterion Eq. (3) do not keep satisfying it after mutation.

The activation mutation is a simple random bit mutation. It samples a random number between 0 and 1 using a uniform distribution and, if this sample is less than or equal to a mutation probability, then it flips the bit. We apply this process to all activation genes of the individual, as the Algorithm 2 shows.

The case of the uniform mutation needs to be distinct between the two chromosome representations. The case where the relative order of the wells is simpler, and we show it in the Algorithm 3.

The Algorithm 3 first selects the gene to mutate. This gene can be a position or a type gene. After that, it samples a random integer in the range of possible values for the selected gene. Finally, it returns a copy of the original individual, but having the new mutated gene. This algorithm, however, does not enforce the relative order of the wells of the individual, as in the equation Eq. (3). Therefore, it takes a small modification to render this algorithm useful for the order-sensitive representation.



---

**Algorithm 4.** Uniform mutation.

---

```

1: procedure UNIF_MUTATE_ORDER( $\mathbf{x}$ )
2:   repeat
3:      $\mathbf{x}_{new} \leftarrow \text{UNIF\_MUTATE}(x)$ 
4:   until  $\mathbf{x}_{new}$  satisfies the eq. (3)
5:   return  $\mathbf{x}_{new}$ 

```

---



---

**Algorithm 5.** Single Point Crossover

---

```

1: procedure SINGLE_CROSS( $\mathbf{x}_1, \mathbf{x}_2$ )
2:   Randomly choose an index  $i$ ,  $1 \leq i \leq 8N$ 
3:   left( $\mathbf{x}_1$ )  $\leftarrow$  genes of  $\mathbf{x}_1$  having index  $\leq i$ .
4:   left( $\mathbf{x}_2$ )  $\leftarrow$  genes of  $\mathbf{x}_2$  having index  $\leq i$ .
5:   right( $\mathbf{x}_1$ )  $\leftarrow$  genes of  $\mathbf{x}_1$  having index  $> i$ .
6:   right( $\mathbf{x}_2$ )  $\leftarrow$  genes of  $\mathbf{x}_2$  having index  $> i$ .
7:    $\mathbf{x}'_{new} \leftarrow$  left( $\mathbf{x}_1$ ) concatenated with right( $\mathbf{x}_2$ ).
8:    $\mathbf{x}''_{new} \leftarrow$  right( $\mathbf{x}_1$ ) concatenated with left( $\mathbf{x}_2$ ).
9:   return ( $\mathbf{x}'_{new}, \mathbf{x}''_{new}$ )

```

---

The Algorithm 4 shows how to guarantee that the mutated individual satisfies the criterion (3) provided the original individual satisfy it. First, it tries to mutate using the Algorithm 3. If the mutation result does not meet the order, then it attempts to mutate again. By doing so, it guarantees that all ordered individuals have approximately equal probability of generation.

- (2) *Recombination Operator*: recombination operates on two inputs and generates two more individuals. There are two types of recombination employed in the present work: the single point crossover and the arithmetical crossover. The Algorithms 5 and 6 describe these two methods.

The function ROUND( $\mathbf{v}$ ) rounds each element of the vector  $\mathbf{v}$  to its nearest integer. The authors of [3] explain the single point crossover and the arithmetic crossover in detail. These operators are valid only for the representation that does not emphasize the relative order of the wells in the chromosome.

The case of the order-aware representation modifies the arithmetical crossover in a similar the uniform mutation does: it applies the crossover until it finds a pair of individuals that satisfy Eq. (3). The Algorithm 7 displays this procedure.

- (3) *The Optimization Algorithms*: The article [3] explains the classical GA (genetic algorithm) applied. The algorithm uses the Genocop III technique of [10] to handle the nonlinear constraints. To generate a viable initial population, we randomly generate individuals until we obtain the number of required individuals.

**Algorithm 6.** Arithmetic Crossover

---

```

1: procedure ARITH_CROSS( $\mathbf{x}_1, \mathbf{x}_2$ )
2:    $\alpha \leftarrow \text{RANDOM}(0, 1)$ 
3:    $\mathbf{x}'_{new} \leftarrow \text{ROUND}(\alpha\mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2)$ 
4:    $\mathbf{x}''_{new} \leftarrow \text{ROUND}((1 - \alpha)\mathbf{x}_1 + \alpha\mathbf{x}_2)$ 
5:   return ( $\mathbf{x}'_{new}, \mathbf{x}''_{new}$ )

```

---

**Algorithm 7.** Arithmetical Crossover for order-aware representation.

---

```

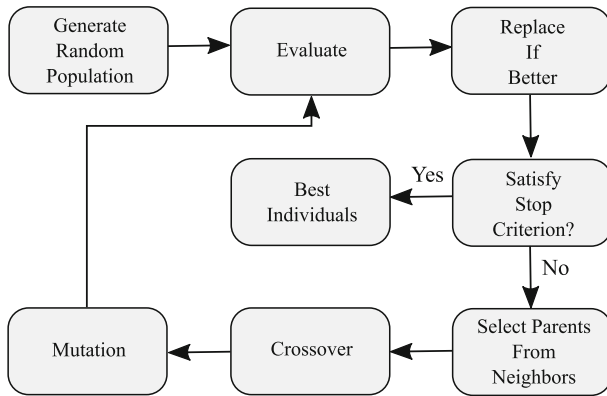
1: procedure ARITH_CROSS_ORDER( $\mathbf{x}_1, \mathbf{x}_2$ )
2:   repeat
3:     ( $\mathbf{x}'_{new}, \mathbf{x}''_{new}$ )  $\leftarrow$  ARITH_CROSS( $\mathbf{x}_1, \mathbf{x}_2$ )
4:   until  $\mathbf{x}'_{new}$  and  $\mathbf{x}''_{new}$  satisfy the eq. (3)
5:   return ( $\mathbf{x}'_{new}, \mathbf{x}''_{new}$ )

```

---

On the other hand, the cellular genetic algorithm (cGA) is a variation of the conventional GA that enforces a geographic location for each individual. As a result, the selection operation takes the individuals locations into account, so the algorithm restricts the recombination to neighbor individuals. Moreover, the substitution operates only on individuals with the same geographic location, so a solution is replaced by a new one only if the new individual is better and has the same geographic position. The authors of [9] make a comparison between classical genetic algorithms and cellular genetic algorithms. Additionally, [7] explains in detail how cGA works. The Fig. 2 and the Algorithm 8 illustrates the cGA workflow.

The next section details our experiments and compare the results of the optimization for both GA and cGA.



**Fig. 2.** Sketch of the cGA algorithm.

---

**Algorithm 8.** Description of the cGA algorithm.
 

---

```

1: procedure EVOLVE_CGA(gen)
2:   repeat
3:     pop  $\leftarrow$  random feasible population
4:     for each individual ind in pop do
5:       parent1  $\leftarrow$  random neighbor of ind
6:       parent2  $\leftarrow$  random neighbor of ind
7:       child  $\leftarrow$  RECOMBINATE(parent1, parent2)
8:       indnew  $\leftarrow$  MUTATE(child)
9:       EVALUATE(indnew)
10:      if indnew evaluation  $\geq$  ind evaluation then
11:        Replace ind by indnew
12:   until gen  $\geq$  genmax
13:   return Best Solution

```

---

## 4 Results

The experiments were divided into three classes: classical GA with orderless representation, classical GA with order-aware representation and cGA with order-aware representation. Since the orderless chromosome implies a bigger search space due to increased redundancy, it is expected better results with the order-aware chromosome.

The experiments aimed to maximize the NPV of exploring the UNISIM synthetic reservoir model [11]. This model simulates in the IMEX simulator [12], and a typical simulation time ranges from 2 to 20 min. Therefore, practical optimizations cannot have much more than a thousand fitness function evaluations, or it would take too long to complete.

This work placed the experiments in the OCTOPUS 2 reservoir management platform [2], by developing a new optimization plugin. This way, we were able to focus solely on the scientific aspects of the experiments, namely the evolutionary algorithms and the optimization results.

Both the classical GA and the cGA used binary tournament selection, where the winning probability is proportional to the fitness value. Further, the algorithms utilized the appropriate arithmetic crossover for the chromosome and, more specifically, the classical GA with orderless representation also adopted the single point crossover. Finally, both algorithms employed a “replace if better” substitution principle, where the new individuals replace the older ones only if they have a better fitness.

Additionally, the cGA algorithm employed an adaptive grid scheme to maintain a healthy population diversity. This paper uses the technique of [7] to control the entropy of the population. Whenever the entropy is decaying too fast, it changes the grid to a more narrow shape, thus making it harder to propagate the best individuals. Conversely, whenever the entropy is decaying too slowly (or decaying at all), it reshaped the grid to make it more square, thus allowing a faster

convergence rate. Hence, it avoids convergence to local maxima and maximized the chances of finding the global maximum of the problem.

Table 1 summarizes the parameters used in each experiment. In particular, the cGA needs the threshold  $\epsilon$ , which controls the grid shape switching procedure. As we show, we tried to make the experiments as even as possible, so we believe the comparisons among their results are fair.

**Table 1.** Summary of the optimization parameters

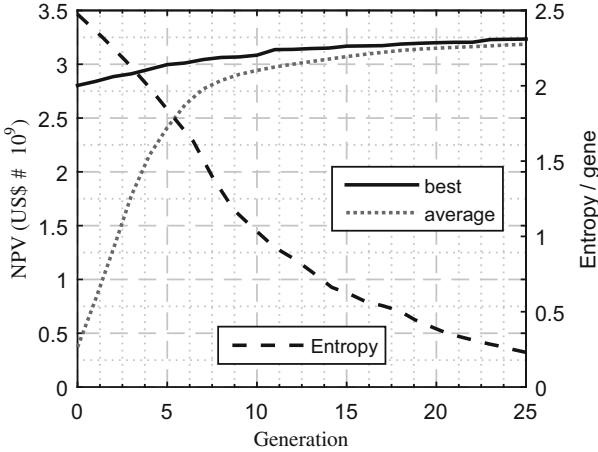
	Classical GA	Cellular GA
Population size	24	48
Generations	50	25
Mutation rate	10%–70%	10%–70%
Mutation	Uniform and activation	Uniform and activation
Recombination	Single Pt. and Arithmetic	Arithmetic
Threshold $\epsilon$	–	0.05
Max. wells ( $N$ )	20	20
Wells radii	0.0762 m	0.0762 m

In addition to the optimization parameters, a typical economic scenario is shared for all experiments. Specifying herein all the constants of the Eqs. (4)–(6) would be impractical, so Table 2 exhibits only a few of them.

**Table 2.** Economic scenario parameters

Platform cost ( $C_p$ )	US\$	1476451065.25
Oil price ( $P_o$ )	US\$/m <sup>3</sup>	250.00
Gas price ( $P_g$ )	US\$/m <sup>3</sup>	0.05
Oil prod. cost ( $O_{pc}$ )	US\$/m <sup>3</sup>	40.00
Water prod. cost ( $W_{pc}$ )	US\$/m <sup>3</sup>	2.00
Water inj. cost ( $W_{ic}$ )	US\$/m <sup>3</sup>	2.00
Gas inj. cost ( $G_{ic}$ )	US\$/m <sup>3</sup>	0.002
Gas prod. cost ( $G_{pc}$ )	US\$/m <sup>3</sup>	0.002
Tax rate ( $I$ )	%	34.00
Discount rate ( $D$ )	%	9.00

The Figs. 3, 4 and 5 illustrate the preliminary results, respectively, of the classical GA with orderless and ordered chromosomes and the cGA with ordered

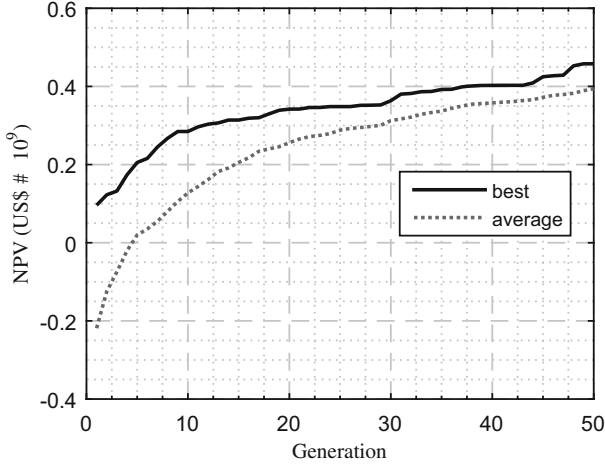


**Fig. 3.** Average of 10 runs of the cGA algorithm using the order-aware representation.

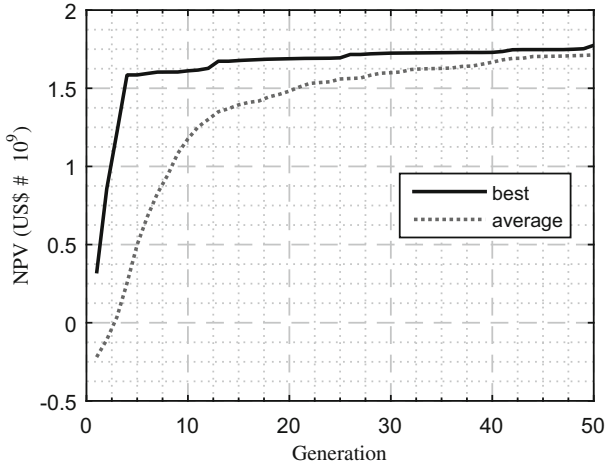
chromosome. All experiments executed 10 times, and the figures display the averaged *NPV*. Moreover, all three experiments had about 800 evaluations of the fitness function, and thus they took about the same total time to complete. Since the cGA method used an entropy based control of diversity, the entropy is a secondary axis in the plot in Fig. 3.

On average, the proposed representation using order-aware chromosomes reaches an optimum with roughly 6–10 times bigger *NPV*. In particular, the cGA version performs better than the GA with order-aware chromosome by a factor of 2, which was concluded by comparing the curves “best” of Figs. 3 and 5. Also, it is possible to note that the average individual of the cGA with the ordered representation reaches an 1 billion *NPV* in generation 3, whereas the best solution of the classical GA with orderless representation does not find this value at all. Hence, we tend to think that the chromosome representation that enforces the relative wells order is more efficient than the traditional orderless representation.

Table 3 compares the final results of the three optimization models. The “relative” results use the formula  $\frac{[NPV(\text{end}) - NPV(1)]}{|NPV(1)|}$ , where  $NPV(\text{end})$  is the final *NPV* and  $NPV(1)$  is the initial *NPV*. As it can be checked from the row “Best *NPV*”, the cellular GA gives the higher final result, with over 3 billion US\$ *NPV* for the best individual and the average population fitness. On the other hand, traditional GA with orderless representation yields the worst result of the three models. Additionally, it can be observed that the relative improvements are also bigger whenever we employed the proposed order-aware chromosome. Hence, although these findings are still preliminary, we believe that they reflect the consequences of integer optimization with smaller search spaces.



**Fig. 4.** Average of 10 runs of the GA algorithm using the orderless representation.



**Fig. 5.** Average of 10 runs of the GA algorithm using the order-aware representation.

When comparing conventional and cellular GA with the proposed representation, we concluded that the cGA is generally better than the conventional GA, for the cGA finds more adapted individuals on an average of 10 experiments. This fact is observable by comparing the “average” curves of Figs. 3 and 5. In theory, this behavior relates to the controlled diversity nature of the cGA. In our experiment, the population entropy is controlled during the evolution, thereby avoiding local maxima, which is a weakness of the classical GA.

**Table 3.** Comparison among the three models after 10 runs. The rows “relative” show the relative improvement of the algorithm

	GA (orderless)	GA (ordered)	cGA (ordered)
Best <i>NPV</i> (US\$)	$0.45786 \times 10^9$	$1.7747 \times 10^9$	$3.2341 \times 10^9$
Relative best	377%	462%	15%
Average <i>NPV</i>	$0.39411 \times 10^9$	$1.7137 \times 10^9$	$3.1861 \times 10^9$
Relative average	280%	879%	760%

## 5 Conclusions

This work presented two new approaches for the wells placement and type optimization problem using evolutionary algorithms. These are the cGA algorithm and the order-aware chromosome model.

The Sect. 2 depicted the fundamental problem approached, explaining its discrete nature, the need for a reservoir simulator and the adopted idea of maximizing the net present value of the reservoir under analysis. Then, the Sect. 3 proposed the order-aware representation, in contrast to the classical chromosome utilized for representing the wells alternatives. Next, the text describes the conventional and cellular genetic algorithms, emphasizing the cGA is a new approach to this kind of optimization problem. Finally, the Sect. 4 presented three experiments: classical GA with the traditional and the order-aware representations and the cGA with the order-aware representation. The preliminary findings showed that the order-aware chromosome proposed is generally better, for the experiments that used it converged to higher *NPV*. We believe that this better behavior is due to reduced search space since the proposed order-aware chromosome reduces the redundancy of individuals because, in general, there is only one possible representation for each decoded physical implementation. Moreover, we also observed that the cGA performed better than the traditional GA, for the average population and the best individual of the cGA evolved to a higher *NPV*. We credit it to the population diversity control that is a natural part of the cGA algorithm, and that is absent on the classical GA. Hence, the cGA features a smaller probability of hanging in local maxima of the fitness function than the standard GA.

The next steps of this work are analyzing the solution itself that the presented models deliver. It needs to analyze the best individuals in the context of oil field engineering, to guarantee that the outputted wells alternative is indeed a valid and implementable physical solution. Besides, it is necessary to replicate the experiments to other oil fields and check whether the present conclusions still hold. On the other hand, we should improve the crossover and mutation operators of the order-aware representation, to remove the need of reapplying the operator if its output did not satisfy the order of Eq. (3).

The presented model is widely applicable beyond the area of oil field optimization. In particular, the concept of an order-aware chromosome that is space

efficient is relevant for any integer optimization problem using evolutionary algorithms. Moreover, any improvement in the area of oil field optimization increases the economic viability of reservoirs and is of particular concern to top oil companies in the world.

**Acknowledgment.** The authors would like to thank the support of CENPES—PETROBRAS throughout all the research steps.

## References

1. Yeten, B., Durlofsky, L.J., Aziz, K., et al.: Optimization of nonconventional well type, location and trajectory. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers (2002)
2. Lima, R., Abreu, A.C., Pacheco, M.A., et al.: Optimization of reservoir development plan using the system octopus. In: Offshore Technology Conference. OTC Brasil (2015)
3. Emerick, A.A., Silva, E., Messer, B., Almeida, L.F., Szwarcman, D., Pacheco, M.A.C., Vellasco, M.M.B.R., et al.: Well placement optimization using a genetic algorithm with nonlinear constraints. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers (2009)
4. Morales, A.N., Gibbs, T.H., Nasrabadi, H., Zhu, D., et al.: Using genetic algorithm to optimize well placement in gas condensate reservoirs. In: SPE EUROPEC/EAGE Annual Conference and Exhibition. Society of Petroleum Engineers (2010)
5. Bittencourt, A.C., Horne, R.N., et al.: Reservoir development and design optimization. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers (1997)
6. Nasrabadi, H., Morales, A., Zhu, D.: Well placement optimization: a survey with special focus on application for gas/gas-condensate reservoirs. *J. Nat. Gas Sci. Eng.* **5**, 6–16 (2012)
7. Dorronsoro, B., Alba, E.: *Cellular Genetic Algorithms*. Springer, New York (2008)
8. Eberly, D.: Robust computation of distance between line segments. Technical report, Geometric Tools, LLC (2015)
9. Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q., Li, J.-J.: Distributed evolutionary algorithms and their models: a survey of the state-of-the-art. *Appl. Soft Comput.* **34**, 286–300 (2015)
10. Michalewicz, Z., Nazhiyath, G.: Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In: IEEE International Conference on Evolutionary Computation, vol. 2, pp. 647–651. IEEE (1995)
11. Avansi, G.D., Schiozer, D.J.: UNISIM-I: synthetic model for reservoir development and management applications. *Int. J. Model. Simul. Pet. Ind.* **9**(1), 21–30 (2015)
12. Three-Phase, black-oil reservoir simulator, CMG (Computer Modeling Group Ltd.). [http://www.cmgl.ca/uploads/files/pdf/SOFTWARE/2015ProductSheets/IMEX\\_Technical\\_Specs\\_15-IM-04.pdf](http://www.cmgl.ca/uploads/files/pdf/SOFTWARE/2015ProductSheets/IMEX_Technical_Specs_15-IM-04.pdf) (2015)