

Optimum Wells Placement in Oil Fields Using Cellular Genetic Algorithms and Space Efficient Chromosomes

Alexandre Ashade L. Cunha^(✉), Giulia Duncan, Alan Bontempo, and Marco Aurélio C. Pacheco

ICA, PUC-RJ, Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro, Gávea, RJ, Brazil

{giuliaduncan,alanbontempo}@gmail.com, alexandre@ashade.com.br

Abstract. The present work introduces a new approach to the optimum wells placement problem in oil fields using evolutionary computation. In particular, our contribution is twofold: we propose an efficient algorithm for initialisation of highly constrained optimisation problems based on Monte-Carlo sampling and we propose a new optimisation technique that uses this population sampling scheme, a space-efficient chromosome and the application of cellular genetic algorithms to promote a large population diversity. Usually, authors define a domain representation having oil wells placed 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. Therefore, the suggested scheme diminishes the problem size, thus making the optimisation more efficient. Moreover, by also employing a cellular genetic algorithm, we guarantee an improved population diversity along the algorithm execution. The experiments with the UNISIM-I reservoir indicate an enhancement of 6 to 10 times of the final NPV when comparing the proposed representation and the traditional one. Besides, the cellular genetic algorithm with the suggested chromosome performs better than the classical genetic algorithm by a factor of 1.5. The proposed models are valuable not only for the oil and gas industry but also to every integer optimisation problem that employs evolutionary algorithms.

1 Introduction

The problem of optimising wells placement in oil fields is essential for oil companies. Many engineering and geological variables affect the reservoir and produce complicated constraints. Accordingly, decision making is not a simple task and finding solutions to minimise cost and maximise profits is an essential and challenging problem. In this context, optimisations are an automated process to seek solutions for oil well locations, trajectories and types, providing safe reservoir exploration plans.

The present text handles nonconventional wells, which are arbitrary wells regarding slope, shape and type [1]. There are numerous works on this matter [1–9], most of them using commercial reservoir simulators in conjunction with proved optimisation heuristics to determine a suitable wells placement alternative.

Articles [1–4] use classical 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, these 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 some possible disadvantages to these models. For instance, the search-space size increases exponentially whenever the maximum number of wells in the chromosome increases. As a result, the number of simulated scenarios required to achieve a proper solution grows too large. Since simulation of oil fields is a computationally intensive task, any optimisation algorithm that would require too many simulations to reach the optimum is impractical.

Another disadvantage of the above models is the redundancy of their chromosome representation: many distinct chromosome instances decode to the same concrete implementation. In particular, 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 different solutions have the same fitness, which, in turn, slows down the convergence rate.

An additional limitation of standard genetic algorithms is the lack of control of their population diversity. As the algorithm iterates, highly 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. In particular, the cellular genetic algorithm is an adaptation of standard genetic algorithm that enforces diversity by defining geographic locations of the individuals and applying recombination exclusively between neighbours [10].

Lastly, the standard genetic algorithm models in the literature do a naive population initialisation: they randomly sample individuals until the required number of feasible individuals is found. While this scheme works for simple optimisation problems, it is not efficient for the kind of black-box nonlinear constraints present in the optimisation of oil fields. Therefore, we also propose a new way to find a random feasible initial population using Monte-Carlo sampling.

The following sections describe a new model to solve the optimal wells placement problem using evolutionary heuristics and geophysics simulations. Our model derives from the work of [3] and enforces a unique domain representation for each physical implementation. Therefore, the proposed model has less redundancy when compared to those in the literature, and it should result in less fragmented populations with better evolution curves. Moreover, we apply a proved cellular genetic algorithm (CGA) with online diversity control, and compare it to the standard genetic algorithms. Additionally, the proposed model adapts the Genocop III technique to make it meaningful with CGA and integer genes. Finally, we also describe how to use Monte-Carlo sampling to efficiently find a feasible initial population to start the evolution process.

2 Problem Description

This text addresses the problem of optimising reservoir exploration alternatives by deciding its wells locations, trajectories, and types to maximise the *NPV* (net present value) of the oil field.

The *NPV* is calculated using the per well oil production and costs, acquired from a commercial reservoir simulator [11]. The commercial simulator needs a 3-dimensional discrete geological representation of the petroleum reservoir, provided beforehand. An example of freely available reservoir model is the UNISIM-I [12] provided by UNICAMP, a state university from Brazil.

Considering that the petroleum reservoir model is a discrete grid of cells, then the wells trajectories assume a finite set of values, represented by the coordinates of the grid blocks. Hence, we represent them using a single line segment, and only the endpoints of the well are required.

Regarding well types, the adopted simulation model sets the available choices. The black oil model [13–15] allows two types of wells: water injector and oil producer. The first type represents a well that injects water into the reservoir, whereas the latter type accounts for a well that extracts oil from the reservoir. On the other hand, the compositional model [16–18] is a more complex scheme that allows additional well types, including the water alternating gas (WAG) type and the cyclic well type. Since the compositional model uses more well types, the search-space is usually bigger and, consequently, the optimisation of wells alternatives become more demanding.

Additionally, the optimiser should also be able to define the optimal number of wells in a field, considering the possibility of pre-existing wells. Usually, this is accomplished by setting a maximum number of wells in the oil field and by including auxiliary binary variables that define which wells are active. Then, only active wells are considered a part of the concrete implementation. This approach increases the problem domain by a factor of 2^N , where N is the maximum number of wells in the oil field. This work depicts an alternate chromosome representation that reduces this factor from 2^N to N .

Finally, it is useful to be able to optimise the wells placement under uncertainty. In particular, the industry is interested in finding a wells placement alternative that is robust to poorly-known parameters. These parameters are typically related to geological or geophysical properties of the oil field, which are modelled by statistical distributions. Since some of the problem parameters are random variables, the resulting *NPV* is a random variable. Consequently, we try to maximise the *expected value* of the *NPV*. This work also shows how to use the proposed optimisation algorithm to approach this kind of problem.

3 Methodology

The purpose of this section is to present the mathematical formulation of the optimisation problem and the proposed solution model.

At first, we express the optimisation as a black-box nonlinear integer programming problem. Thus all decision variables are integer numbers, and

all mathematical functions are analytically unknown, but numerically computable. In particular, the numerical computation of the fitness function is time-consuming, deterministic and robust. Conversely, the numerical calculation of all constraints is relatively easy.

Next, this section presents two algorithms to solve it: a conventional genetic algorithm (GA) and a cellular genetic algorithm (CGA). The CGA as applied to the optimal wells placement problem is an innovation, and we also propose novelties to the classical CGA present in the current literature. These novelties include a black-box constraint handling scheme for CGA based on the Genocop III [19] heuristic, an efficient initial population sampler based on the Metropolis-Hastings algorithm [20, 21] that guarantees that all sampled individuals are feasible, and a space-efficient chromosome representation, which reduces the search space size by removing redundancy. We also propose a recombination operator and a mutation operator for this new representation. Finally, we proposed an adaptation of the model to handle uncertainties in the definition of the reservoir geophysical parameters.

3.1 Mathematical Formulation

The standard representation uses six integer variables to represent the discrete coordinates of two blocks in the well ends locations, one binary integer to indicate that a specific well exists (the activation bit) and another 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 τ denote the well type. If N is the maximum allowed wells count, then the decision vector \mathbf{x} is:

$$\begin{aligned} \mathbf{x} = & (a_1, a_2, \dots, a_N, \\ & \bar{i}_1, \bar{j}_1, \bar{k}_1, \underline{i}_1, \underline{j}_1, \underline{k}_1, \tau_1, \\ & \bar{i}_2, \bar{j}_2, \bar{k}_2, \underline{i}_2, \underline{j}_2, \underline{k}_2, \tau_2, \dots, \\ & \bar{i}_N, \bar{j}_N, \bar{k}_N, \underline{i}_N, \underline{j}_N, \underline{k}_N, \tau_N) \end{aligned} \quad (1)$$

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 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 $f(\mathbf{x}) = f(\mathbf{x}')$. More generally, the NPV is invariant to any permutation of the wells. 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. The proposed model starts by comparing 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*.

Another proposal to reduce even more the search space is to replace the per-well activation bits by a single integer variable that represents the number of active wells. Therefore, by rearranging the chromosome such that the active wells appear before the inactive ones, the new chromosome has only N distinct possibilities, as opposed to the former 2^N possibilities of the chromosome presented in Eq. (1), where is the maximum wells count. Then, our final space-efficient chromosome reads:

$$\mathbf{x} = (\eta, w_1, w_2, \dots, w_N) \quad (4)$$

where w_1 to w_η are active wells.

The objective function is the NPV of the platform. The presented 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, we need to take 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 proposed model is based on 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. (5) depicts this model.

$$NPV = \sum_{k=1}^N NPV_w(k) - C_P \quad (5)$$

In the Eq. (5), 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:

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

```

1: procedure  $\leq (w_1, w_2)$ 
2:    $\triangleright w_k = (\bar{i}_k, \bar{j}_k, \bar{k}_k, \bar{l}_k, \bar{j}_k, \bar{k}_k, \tau_k)$ 
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  $\bar{k}_1 < \bar{k}_2$  then
13:     return true
14:   else if  $\bar{k}_1 > \bar{k}_2$  then
15:     return false
16:   if  $\tau_1 \leq \tau_2$  then
17:     return true
18:   else
19:     return false

```

$$NPV_w(k) = (1 - I) \cdot \sum_{t=1}^T \frac{R(k, t) - C_o(k, t)}{(1 + D)^{y_t}} - C_w(k), \quad (6)$$

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:

$$R(k, t) = O_p(k, t) \cdot P_o(t) + G_p(k, t) \cdot P_g(t) \quad (7)$$

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 expenses in the time-step, the variable expenses in the time-step, the royalties, and the costs associated with the amount of fluid production or injection. The Eq. (8) shows the general equation.

$$\begin{aligned}
 C_o(k, t) = & [C_M \cdot (y_t - y_{t-1})] \\
 & + C_{vf} + R_y \cdot R(k, t) \\
 & + \left(O_p(k, t) \cdot O_{pc} + G_p(k, t) \cdot G_{pc} + \right. \\
 & W_p(k, t) \cdot W_{pc} + G_i(k, t) \cdot G_{ic} + \\
 & \left. W_i(k, t) \cdot W_{ic} \right)
 \end{aligned} \tag{8}$$

In the Eq. (8), 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 amount 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 uses 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 centre 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:

$$l(k) \leq L \tag{9}$$

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:

$$d(w_{k_1}, w_{k_2}) \geq d_{min}. \tag{10}$$

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 [22], the author explains this problem in detail.

In conclusion, we seek the solution \mathbf{x} , as defined in the Eq. (4), that maximises the *NPV* in Eq. (5), subject to the nonlinear restrictions (9) and (10). The following section describes the algorithm this paper employs for solving this problem efficiently on a digital computer.

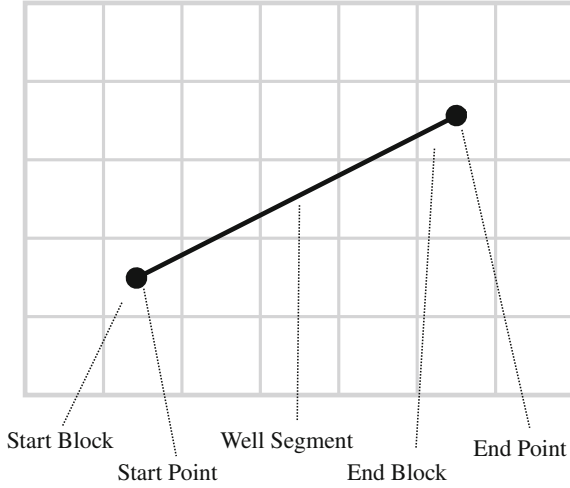


Fig. 1. Representation of well in a grid

3.2 Solution to the Optimisation Problem

This work uses evolutionary algorithms to solve the optimisation problem of the Sect. 3.1. In particular, we use a classical genetic algorithm and a more modern approach, the cellular genetic algorithm [23, 24]. The classical GA utilises a chromosome representation that does not enforce the wells order and has one activation bit for each well. On the other hand, the proposed CGA model uses the space-efficient representation as mentioned earlier, which reduces the search space by ensuring a certain wells order and by using a single gene to represent the number of active wells, as opposed to activation bits.

The current work employs the chromosome of the Eq. (2). The fitness function is the *NPV* of the platform, as the Eq. (5) exhibits.

Both the cellular and classical genetic algorithms require mutation and recombination operators. Since there are two possible chromosome

representations, the optimisation model needs to develop these operators according to each representation.

Mutation Operators

Mutation operates on a single individual, possibly generating a new (mutated) individual. Our model employs 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. For each activation bit, we sample a random number between 0 and 1 using a uniform distribution, and if this random number is less than or equal to a mutation probability, the activation bit is flipped. This mutation type is only meaningful if the representation uses activation bits.

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  $\mathbf{x}_{new}$ 

```

Contrary to the activation mutation, the uniform mutation needs to distinct between the two chromosome representations. For the orderless chromosome representation, the algorithm 3 depicts the process.

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 of the selected gene. Finally, it returns a copy of the original individual with 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.

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.

Recombination

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 $\text{ROUND}(\mathbf{v})$ 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 emphasise the relative order of the wells in the chromosome.

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  $\mathbf{x}_{new}$ 

```

Algorithm 4 Uniform mutation (order-aware representation).

```

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:    $\text{left}(\mathbf{x}_1) \leftarrow$  genes of  $\mathbf{x}_1$  having index  $\leq i$ .
4:    $\text{left}(\mathbf{x}_2) \leftarrow$  genes of  $\mathbf{x}_2$  having index  $\leq i$ .
5:    $\text{right}(\mathbf{x}_1) \leftarrow$  genes of  $\mathbf{x}_1$  having index  $> i$ .
6:    $\text{right}(\mathbf{x}_2) \leftarrow$  genes of  $\mathbf{x}_2$  having index  $> i$ .
7:    $\mathbf{x}'_{new} \leftarrow \text{left}(\mathbf{x}_1)$  concatenated with  $\text{right}(\mathbf{x}_2)$ .
8:    $\mathbf{x}''_{new} \leftarrow \text{right}(\mathbf{x}_1)$  concatenated with  $\text{left}(\mathbf{x}_2)$ .
9:   return  $(\mathbf{x}'_{new}, \mathbf{x}''_{new})$ 

```

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 ( $\mathbf{x}'_{new}, \mathbf{x}''_{new}$ )

```

For the case of the order-aware representation, we modify the arithmetical crossover similarly to the adaptation of the uniform mutation. We apply the crossover until a pair of individuals that satisfy Eq. (3) is found. The algorithm 7 displays this procedure.

Population Initialisation

Initialising the population is the process of creating random individuals for the first generation of the evolutionary algorithm. Since the Genocop III technique requires a fully feasible initial population, we need to find a way to sample individuals from the feasible set of solutions. In other words, the sampling method should be able to choose individuals from the set of all feasible individuals uniformly.

A naive solution would be to uniformly sample individuals until the required number of feasible individuals is found. Therefore, if a sampled individual does not satisfy one of the constraints, then it is discarded, and the process continues to sample new individuals. Algorithm 8 depicts this method.

The main drawback of algorithm 8 becomes clear whenever the feasible search space is small when compared to the whole search space. In this case, the probability of uniformly choosing a chromosome that satisfies all the constraints is small; thus it might take too long to find a valid initial population. Therefore, this article proposes to sample individuals based on a distribution function that has small probability in infeasible subsets of the search space and has a higher (and almost uniform) probability within feasible subsets of the search space.

The chosen algorithm to draw samples from a pre-specified custom distribution is the Metropolis-Hastings method [20,21]. This algorithm requires the definition of the target density function $\Pi(\mathbf{x})$ and a conditional density function $Q(\mathbf{x}|\mathbf{x}')$ (so-called the candidate's proposal or kernel). Given a current sampled individual \mathbf{x}_t , a new sample is drawn from the conditional Q density, and it is kept if it has likelihood greater than the current sample likelihood. Algorithm 9 explains in detail the Metropolis-Hastings scheme.

Algorithm 8 Naive Sampling

```

1: procedure NAIVE_SAMPLING(pop_size)
2:   pop_size_counter  $\leftarrow$  0
3:   repeat
4:     sample new individual  $\mathbf{x}$ 
5:     if  $\mathbf{x}$  is feasible then
6:       pop_size_counter  $\leftarrow$  pop_size_counter + 1
7:   until pop_size_counter = pop_size

```

Algorithm 9 Metropolis Hastings Sampling

```

1: procedure METROPOLIS_HASTINGS_SAMPLING(pop_size)
2:   pop_size_counter  $\leftarrow$  0
3:    $t \leftarrow 0$ 
4:    $\mathbf{x}_0 \leftarrow$  sample from  $\Pi(\mathbf{x})$ 
5:   repeat
6:     draw sample  $\mathbf{Y}$  from  $Q(\mathbf{x}|\mathbf{x}_t)$ 
7:      $a_1 \leftarrow \Pi(\mathbf{x})/\Pi(\mathbf{x}_t)$ 
8:      $a_2 \leftarrow Q(\mathbf{x}_t|\mathbf{x})/Q(\mathbf{x}|\mathbf{x}_t)$ 
9:      $a \leftarrow a_1 \cdot a_2$ 
10:    if  $a \geq 1$  then
11:       $\mathbf{x}_{t+1} \leftarrow \mathbf{Y}$ 
12:    else
13:       $\mathbf{x}_{t+1} \leftarrow \begin{cases} \mathbf{Y} & \text{with probability } a \\ \mathbf{x}_t & \text{with probability } 1 - a \end{cases}$ 
14:     $t \leftarrow t + 1$ 
15:    if  $\mathbf{x}_{t+1}$  is feasible then
16:      pop_size_counter  $\leftarrow$  pop_size_counter + 1
17:  until pop_size_counter = pop_size

```

We use the algorithm 9 to sequentially draw samples from the distribution $\Pi(\mathbf{x})$ until the required number of feasible individuals is found, similarly to the case of the random sampling mentioned earlier. However, since our method designs the distribution $\Pi(\mathbf{x})$ so that it has higher probabilities within feasible subsets of the search space, this proposed scheme tends to be faster and more reliable than the naive solution. The next few paragraphs explain in detail how to design the function $\Pi(\mathbf{x})$ based on the maximum well length constraint of Eq. (9) and the minimum pairwise wells distance constraint of Eq. (10).

Hence, we write the density function $\Pi(\mathbf{x})$ as a product of two density functions, namely $\Pi_d(\mathbf{x})$ and $\Pi_l(\mathbf{x})$:

$$\Pi(\mathbf{x}) = \Pi_d(\mathbf{x}) \cdot \Pi_l(\mathbf{x}) \quad (11)$$

The density Π_d does not value individuals that have pairs of wells too close to each other. More clearly, if the minimum allowed pairwise wells distance is d_{min} , then:

$$\Pi_d(\mathbf{x}) = \prod_{(r,s)} \Pi_{dw}(w_r, w_s)$$

$$\Pi_{dw}(w_r, w_s) = \begin{cases} A \tanh(\zeta(\lambda - 1)) + 1 & \lambda \leq 5 \\ \nu e^{5-\lambda} & \lambda > 5 \end{cases} \quad (12)$$

where

$$\begin{aligned} \nu &= 1.9 \\ \zeta &= 10.0 \\ A &= \frac{\nu - 1}{\tanh(4\zeta)} \\ \lambda &= \frac{d(w_r, w_s)}{d_{min}} \end{aligned} \quad (13)$$

Thus, as the wells distance of a particular pair of wells of the individual \mathbf{x} approaches 0, the value of $\Pi_d(\mathbf{x})$ decreases, as expected. The same happens if the distance between a pair of wells approaches infinity because the function $\Pi_d(\mathbf{x})$ should be integrable. If all pairs of wells of the individual \mathbf{x} satisfy $1 \leq \lambda \leq 5$, the value of $\Pi_d(\mathbf{x})$ is high and almost constant, as expected (see Fig. 2).

The next function is $\Pi_l(\mathbf{x})$, which should be high for chromosomes whose wells are large, but smaller than the maximum well size. The proposed function is:

$$\Pi_l(\mathbf{x}) = \prod_{r=1}^N \Pi_{lw}(w_r)$$

$$\Pi_{lw}(w_r) = \begin{cases} \left[1 + \left(\frac{l(r)}{L} - 1 \right)^2 \right]^{-1} & l(r) \leq L \\ \exp -\zeta \left(\frac{l(r)}{L} - 1 \right) & l(r) > L \end{cases} \quad (14)$$

where $l(r)$ is the length of the well w_r and L is the maximum allowed length of a well.

As the length of the well increases towards the maximum allowable size L , the likelihood approaches the maximum value 1. Furthermore, as the well length increases above its maximum allowed value, the value of $\Pi_l(\mathbf{x})$ vanishes, which guarantees both the low probability of having large sized wells and the integrability of $\Pi_l(\mathbf{x})$. The Fig. 3 shows how Π_{lw} behaves.

In this work, we compared the naive approach to the proposed Metropolis-Hastings generator. The Sect. 4 specifies the results in detail.

The Evolutionary Algorithms

The article [3] explains the classical genetic algorithm (GA). It uses the Genocop III (Genetic Algorithm for Numerical Optimization of Constrained Problems III) technique [19] to handle black-box constraints. Additionally, for generating an

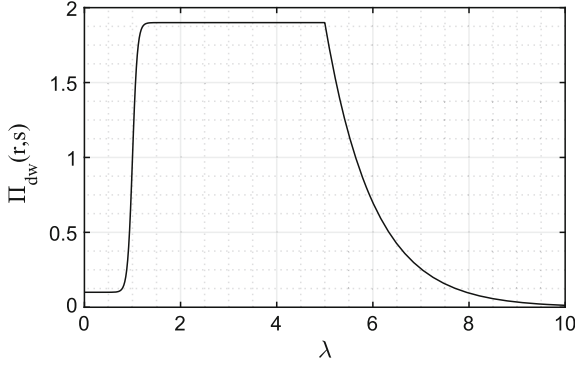


Fig. 2. Likelihood function of the wells distance

Algorithm 10 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:        $parent_1 \leftarrow$  random neighbor of  $ind$ 
6:        $parent_2 \leftarrow$  random neighbor of  $ind$ 
7:        $child \leftarrow$  RECOMBINATE( $parent_1, parent_2$ )
8:        $ind_{new} \leftarrow$  MUTATE( $child$ )
9:       EVALUATE( $ind_{new}$ )
10:      if  $ind_{new}$  evaluation  $\geq$   $ind$  evaluation then
11:        Replace  $ind$  by  $ind_{new}$ 
12:   until  $gen \geq gen_{max}$ 
13:   return Best Solution

```

initial population, the algorithm randomly creates chromosomes until it finds a sufficient number of feasible individuals.

On the other hand, the cellular genetic algorithm (CGA) is a variation of the standard GA that enforces a unique geographic location for every individual. As a result, the selection operation takes the individuals locations into account, so the algorithm restricts the recombination to neighbour 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 [23] make a comparison between classical genetic algorithms and cellular genetic algorithms. Additionally, [10] explains in detail how CGA works. The Fig. 4 and the algorithm 10 illustrates the CGA workflow.

For handling black-box constraints, we propose in the next few paragraphs a variation of Genocop III suitable for the concept of geographic locations the CGA uses.

The Genocop III works by maintaining two separate populations: a search population and a reference population. The reference population is generated at the start of the optimisation during the initial population sampling, and it should not contain infeasible individuals. Conversely, the search population consists of individuals used by the optimisation. Whenever an invalid individual appears, it goes through a repair process that converts it into a feasible one.

Repairing infeasible individuals consists of selecting one of the reference individuals and applying arithmetic crossover between the infeasible and the reference individuals until a new feasible individual is known. Equivalently, the individuals can be interpreted as points so that \mathbf{r} is the reference point and \mathbf{s} is the search point. Thereby, the process creates a segment between \mathbf{s} and \mathbf{r} and chooses a random point \mathbf{z} where $\mathbf{z} = a\mathbf{s} + (1 - a)\mathbf{r}$ and a is a random number between 0 and 1. If \mathbf{z} is infeasible, then the process is repeated until a valid \mathbf{z} is known. After that, \mathbf{z} replaces the infeasible point \mathbf{s} (see Fig. 5).

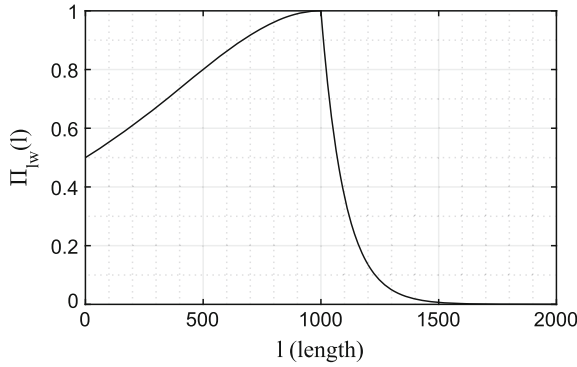


Fig. 3. Likelihood function of the well length

The CGA, unlike the traditional GA, arranges each individual to a determined geographic location. The purpose of the geographic locations is to maintain a healthy population diversity during the evolution process. Therefore, using a random reference individual to repair the infeasible individuals would completely disregard the locations of the individuals, since the reference individual of the Genocop III technique does not emphasise position. Hence, we propose to discard the reference population and to employ as the reference the last known feasible individual in the same location. This scheme maintains the property that each individual is influenced only by its nearby neighbours.

3.3 Algorithm Adaptation to Problems with Uncertainties

To model well placement problems having uncertainties in the geophysical parameters, we use the concept of *geological scenarios*. Geological scenarios are versions of the geological model constructed from samples of the geophysical parameters distribution.

Typical geophysical parameters are modelled as random variables, usually uniformly distributed from a possible range of values. These variables include the reservoir porosity and the reservoir permeability. The geological scenarios are constructed from samples of the porosity and permeability samples, and each scenario is a concrete reservoir model ready to simulate.

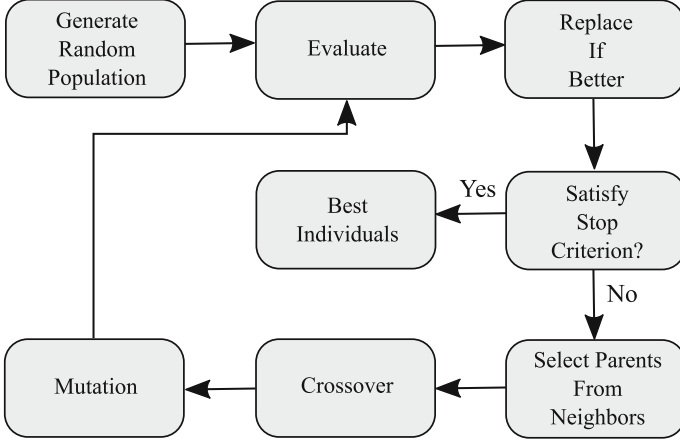


Fig. 4. Sketch of the CGA algorithm

Therefore, the uncertainty in a specific parameter is modelled as a set of geological scenarios. Since each scenario has its specific net present value and its specific probability of occurrence, the all-scenario net present value is defined as the expected value of the NPV calculated as:

$$E[NPV] = \sum_{i=1}^N p_i \cdot NPV_i^s, \quad (15)$$

where NPV_i^s is the NPV of the geological scenario i and p_i is its probability of occurrence. The value p_i is provided beforehand.

Consequently, the fitness evaluation now consists of not only one, but many simulations, because we need many geological scenarios to represent the reservoir. Additionally, the number of NPV computations is also multiplied by the number of geological scenarios, since we need to calculate a NPV for each scenario and the compute Eq. (15). After these modifications, the algorithms proposed in Sect. 3.2 are still valid and their result is now an optimal wells placement alternative that is robust to geological uncertainties.

4 Experiments and Results

This section splits the experiments into two parts: the initialisation part and the optimisation part. The initialisation experiments concern on comparing the

performance of the naive and the enhanced initialisation algorithms described in Sect. 3.2, whereas the optimisations compare the effect of the optimisation models on the final solution.

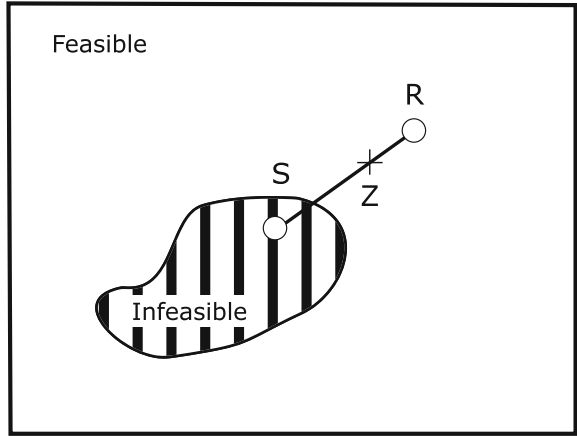


Fig. 5. Sketch of the Genocop III heuristic

4.1 Population Initialisation Experiments

To compare the performance of the Metropolis-Hastings method to the performance of the naive method, we generated 1000 individuals using both algorithms and counted how many of them were feasible. The model used was the UNISIM-I [25], from UNICAMP. We varied the minimum wells distance from 100 to 500 meters and the maximum wells length from 500 to 2000 meters, and we set the number of wells of each individual to 20.

Table 1. Count of feasible individuals found after 1000 samples using the naive and the proposed algorithms for a given maximum allowed well length and minimum allowed pairwise wells distance

Max. length (m)	Min. distance (m)	Naive method	Proposed method
500	100	1	77
2000	100	108	649
500	500	13	381
2000	500	46	302

Table 1 shows the results of the experiment. As the minimum wells distance increases (second column), it becomes harder to find an alternative that respects the minimum wells distance for every pair of wells (there are 190 pairs of wells). However, the Metropolis-Hastings approach still can find a significant number of feasible alternatives, whereas the naive approach cannot.

Moreover, the same pattern is observable when the maximum well length (first column) is constrained to small values. For example, after 1000 samples, the naive approach found one feasible individual for a maximum well length of 500 m (row 1), while the Metropolis-Hastings approach found 77.

We conclude that the proposed technique is more efficient to find the initial population when it requires that all individuals be feasible. From now on, all the following experiments use the proposed technique.

4.2 Optimisation Experiments

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 maximise the NPV of exploring the UNISIM synthetic reservoir model [25]. This model has two flavours: the black-oil version, which simulates in the IMEX simulator [11], and the compositional model version, which simulates in the GEM simulator. The typical simulation time ranges from 2 to 10 min. Therefore, possible optimisations 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 optimisation plug-in. This way, we were able to focus solely on the scientific aspects of the experiments, namely the evolutionary algorithms and the optimisation 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 utilised 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 [10] 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 avoided convergence to local maxima and maximised the chances of finding the global maximum of the problem.

Table 2 summarises the parameters used in each experiment. In particular, the CGA needs the threshold ϵ , 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 2. 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 ϵ	–	0.05
Max. wells (N)	20	20
Wells radii	0.0762 m	0.0762 m

In addition to the optimisation parameters, a typical economic scenario is shared for all experiments. Specifying herein all the constants of the Eqs. (5)–(7) would be impractical, so Table Table 3 exhibits only a few of them.

The Figs. 6, 7 and 8 illustrate the results, respectively, of the classical GA with orderless and ordered chromosomes and the CGA with ordered chromosome. All experiments executed 10 times, and the plots 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

Table 3. Economic scenario parameters

Quantity	Unit	Value
Platform cost (C_p)	billion US\$	1.48
Oil price (P_o)	US\$/m ³	250.00
Gas price (P_g)	US\$/m ³	0.05
Oil prod. cost (O_{pc})	US\$/m ³	40.00
Water prod. cost (W_{pc})	US\$/m ³	2.00
Water inj. cost (W_{ic})	US\$/m ³	2.00
Gas inj. cost (G_{ic})	US\$/m ³	0.002
Gas prod. cost (G_{pc})	US\$/m ³	0.002
Tax rate (I)	%	34.00
Discount rate (D)	%	9.00

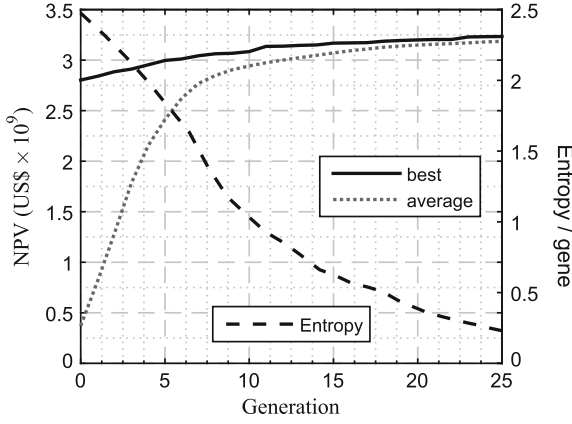


Fig. 6. Average of 10 runs of the CGA algorithm using the order-aware representation

method used an entropy based control of diversity, the entropy is a secondary axis in the plot in Fig. 6.

On average, the proposed representation using order-aware chromosomes reaches an optimum with roughly 6 to 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. 6 and 8. 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

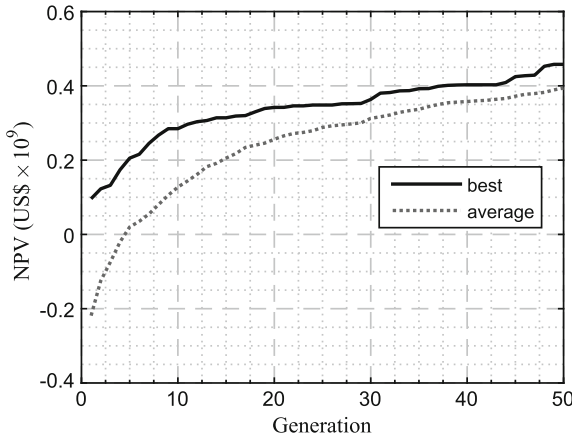


Fig. 7. Average of 10 runs of the GA algorithm using the orderless representation

enforces the relative wells order is more efficient than the traditional orderless representation.

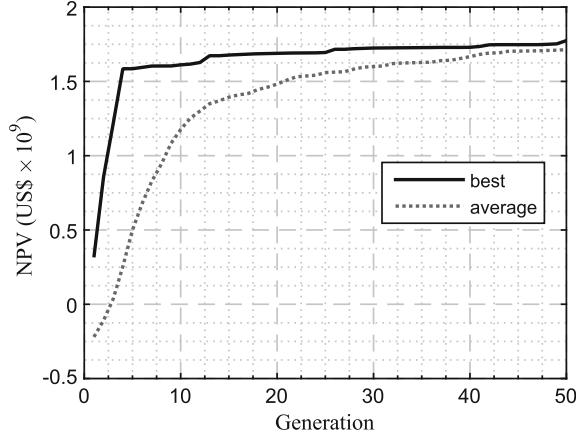


Fig. 8. Average of 10 runs of the GA algorithm using the order-aware representation

Table 4 compares the final results of the three optimisation models. The “relative” results use the formula $[NPV(end) - NPV(1)] / |NPV(1)|$, where $NPV(end)$ 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\$ 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, these findings reflect the consequences of integer optimisation with smaller search spaces.

When comparing conventional to cellular GA with the proposed representation, we concluded that the CGA is better than the standard 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. 6 and 8. In theory, this behaviour 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.

Finally, we compared the performance of the proposed method and the conventional GA for a typical optimisation under uncertainty scenario. Our experiment used the UNISIM-I oil field with 5 different geological scenarios, with probabilities of occurrence equal to 10%, 10%, 20%, 20%, and 50%. Therefore, there were 2 low probability scenarios, 2 medium probability scenarios, and a high probability one. The expected NPV was calculated using Eq. (15).

Table 4. 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 $NPV(US\$)$	0.45786×10^9	1.7747×10^9	3.2341×10^9
Relative best	377%	462%	15%
Average NPV	0.39411×10^9	1.7137×10^9	3.1861×10^9
Relative average	280%	879%	760%

Table 5. Comparison among the three models after 10 runs for the multi-scenario optimisation. The rows “relative” show the relative improvement of the algorithm

	GA (orderless)	GA (ordered)	CGA (ordered)
Best $NPV(US\$)$	0.33572×10^9	1.28813×10^9	1.48871×10^9
Relative best	402%	382%	69%
Average NPV	0.39411×10^9	1.7137×10^9	3.1861×10^9
Relative average	280%	488%	255%

The Table 5 shows that the CGA with order-aware chromosome finds the solution with highest expected NPV . Since the multi-scenario experiment has some poor geological scenarios, the maximum expected NPV is smaller than the NPV on Table 4. We conclude that the proposed CGA + order-aware model is the best option for optimising the wells placement problem, even in the presence of uncertainties.

5 Conclusions

This work presented two new approaches for the wells placement and type optimisation problem using evolutionary algorithms. These are the CGA algorithm with an adapted version of the Genocop III algorithm and the space-efficient order-aware chromosome model. We also presented a new efficient way to find a random feasible initial population based on Monte-Carlo sampling.

The Sect. 2 depicted the fundamental problem approached, explaining its discrete nature, the need for a reservoir simulator and the adopted idea of maximising the net present value of the reservoir under analysis. Then, the Sect. 3 proposed the order-aware representation, in contrast to the classical chromosome utilised for representing the wells alternatives. Next, the text describes the conventional and cellular genetic algorithms, emphasising the CGA is a new approach to this kind of optimisation problem. Finally, the Sect. 4 presented the experiments in two parts.

The first part compared the new population initialisation model to the traditional naive initialisation. The proposed Monte-Carlo based initialisation per-

formed better than the traditional naive approach, especially for highly constrained optimisations. Therefore, we conclude that it should be used to find a feasible initial population on the experiments to follow.

Then, we presented three optimisation experiments: classical GA with the traditional representation and with the proposed order-aware representation and the CGA with the proposed order-aware representation. The findings showed that the order-aware chromosome is better, for the experiments that used it converged to higher *NPV*. We believe that this better behaviour is due to reduced search space since the proposed order-aware chromosome reduces the redundancy of individuals because there is only one possible representation of 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 from the classical GA. Hence, the CGA features a smaller probability of hanging in local maxima of the fitness function than the standard GA.

The presented model is widely applicable beyond the area of oil field optimisation. In particular, the concept of an order-aware chromosome that is space efficient is relevant for any integer optimisation problem using evolutionary algorithms. Additionally, being able to efficiently find random feasible initial populations on highly constrained optimisation problems is always a challenge in the field of evolutionary algorithms, and our model based on Metropolis-Hastings sampling worked fairly well. Finally, it is important to notice that any improvement in the area of oil field optimisation increases the economic viability of reservoirs and is of particular concern to top oil companies in the world. Therefore, we consider this work highly relevant for the oil & gas industry.

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: OTC Brasil. Offshore Technology Conference (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. Jesmani, M., Bellout, M.C., Hanea, R., Foss, B.: Well placement optimization subject to realistic field development constraints. *Comput. Geosci.* **20**(6), 1185–1209 (2016)
8. Siavashi, M., Tehrani, M.R., Nakhaee, A.: Efficient particle swarm optimization of well placement to enhance oil recovery using a novel streamline-based objective function. *J. Energy Resour. Technol.* **138**(5), 052903 (2016)
9. Al Dossary, M.A., Nasrabadi, H.: Well placement optimization using imperialist competitive algorithm. *J. Pet. Sci. Eng.* **147**, 237–248 (2016)
10. Bernabe Dorronsoro, E.A.: *Cellular Genetic Algorithms*. Springer (2008)
11. Three-Phase, black-oil reservoir simulator, CMG (Computer Modeling Group Ltd.) (2015). https://www.cmgl.ca/uploads/files/pdf/SOFTWARE/2015ProductSheets/IMEX_Technical_Specs.15-IM-04.pdf
12. Gaspar, A.T., Avansi, G.D., dos Santos, A.A., von Hohendorff Filho, J.C., Schiozer, D.J.: Unisim-id: Benchmark studies for oil field development and production strategy selection. *Int. J. Model. Simul. Pet. Ind.* **9**(1) (2015)
13. Trangenstein, J.A., Bell, J.B.: Mathematical structure of the black-oil model for petroleum reservoir simulation. *SIAM J. Appl. Math.* **49**(3), 749–783 (1989)
14. Rankin, R., Riviere, B.: A high order method for solving the black-oil problem in porous media. *Adv. Water Res.* **78**, 126–144 (2015)
15. Kozlova, A., Li, Z., Natvig, J.R., Watanabe, S., Zhou, Y., Bratvedt, K., Lee, S.H., et al.: A real-field multiscale black-oil reservoir simulator. *SPE J.* (2016)
16. Thiele, M.R., Batycky, R.P., Blunt, M.J., et al.: A streamline-based 3d field-scale compositional reservoir simulator. In: *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers (1997)
17. Coats, K.H., et al.: An equation of state compositional model. *Soc. Pet. Eng. J.* **20**(05), 363–376 (1980)
18. Qiao, C., Khorsandi, S., Johns, R.T., et al.: A general purpose reservoir simulation framework for multiphase multicomponent reactive fluids. In: *SPE Reservoir Simulation Conference*. Society of Petroleum Engineers (2017)
19. Michalewicz, Z., Nazhiyath, G.: Genocop iii: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In: *1995, IEEE International Conference on Evolutionary Computation*, vol. 2, pp. 647–651. IEEE (1995)
20. Griffin, J.E., Walker, S.G.: On adaptive metropolis–hastings methods. *Stat. Comput.* **23**(1), 123–134 (2013). <http://dx.doi.org/10.1007/s11222-011-9296-2>
21. Yildirim, I.: *Bayesian inference: metropolis-hastings sampling*. Department of Brain and Cognitive Sciences, Univ. of Rochester, Rochester, NY (2012)
22. Eberly, D.: *Robust computation of distance between line segments*. Geometric Tools, LLC, Technical report (2015)
23. 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)
24. Zhao, Y., Chen, L., Xie, G., Zhao, J., Ding, J.: Gpu implementation of a cellular genetic algorithm for scheduling dependent tasks of physical system simulation programs. *J. Comb. Optim.* 1–25 (2016)
25. Avansi, G.D., Schiozer, D.J.: Unisim-i: Synthetic model for reservoir development and management applications. *Int. J. Model. Simul. Pet. Ind.* **9**(1) (2015)

Intelligent Systems and Applications

Extended and Selected Results from the SAI Intelligent
Systems Conference (IntelliSys) 2016

Bi, Y.; Kapoor, S.; Bhatia, R. (Eds.)

2018, XI, 476 p. 255 illus., 195 illus. in color., Hardcover

ISBN: 978-3-319-69265-4