Comparison between Genetic Algorithms and Differential Evolution for Solving the History Matching Problem

Conference Paper *in* Lecture Notes in Computer Science · June 2012

DOI: 10.1007/978-3-642-31125-3_48

CITATIONS
14

READS
2,350

4 authors, including:

Elisa Portes dos Santos Amorim
The University of Calgary
14 PUBLICATIONS **129** CITATIONS
SEE PROFILE

Carolina Ribeiro Xavier
Federal University of São João del-Rei
71 PUBLICATIONS **302** CITATIONS
SEE PROFILE

Ricardo Silva Campos
Federal University of Juiz de Fora
37 PUBLICATIONS **95** CITATIONS
SEE PROFILE

Comparison between Genetic Algorithms and Differential Evolution for solving the history matching problem

Elisa P. dos Santos Amorim[1], Carolina R. Xavier[2,3], Ricardo Silva Campos[4] and Rodrigo W. dos Santos[4]

[1] Dept. of Computer Science, University of Calgary
[2] Departamento de Ciência da Computação - UFSJ
[3] COPPE - UFRJ
[4] Programa de Pós Graduação em Modelagem Computacional - UFJF
epdamori@ucalgary.ca

Abstract. This work presents a performance comparison between Differential Evolution (DE) and Genetic Algorithms (GA), for the automatic history matching problem of reservoir simulations. The history matching process is an inverse problem that searches a set of parameters that minimizes the difference between the model performance and the historical performance of the field. This model validation process is essential and gives credibility to the predictions of the reservoir model. Four case studies were analyzed each of them differing on the number of parameters to be estimated: 2, 4, 9 and 16. Several tests are performed and the preliminary results are presented and discussed.

1 Introduction

Oil is one of the most important sources of energy in the world today. It is usually located inside porous rocks, called reservoirs, hundreds of meters bellow the surface. Its extraction is a very complex and costly process that involves a great amount of decisions that need to be as precise as possible in order to better explore the reservoir potential. Reservoir engineers are directly involved in this process of making decisions and they often rely on reservoir simulators to do so. Reservoir simulators are computational tools based on mathematical models that describe the fluid flow inside the reservoir rock. They are used to predict reservoir behavior under different oil exploration scenarios allowing engineers to test a variety of strategies before deciding which should be applied on the real field. The simulator models are highly dependent on parameters that describe the physical properties of reservoirs, such as permeability and porosity, and the reliability of predictions depends on good estimations for such properties. Unfortunatelly, it is not possible to perform a direct measurement of these properties for all reservoir extensions. Direct measurements are only provided on well locations which are usually hundreds of meters apart from each other.

An alternative for model validation is the estimation of the relevant properties by History Matching (HM)[1]. A log with information regarding production data, such as oil rate and bottomhole pressure measured on wells, is kept for reservoirs that have been in operation for some time. This log is often called *history* of the reservoir. History Matching process is an inverse problem that utilizes reservoir simulation to find a set of parameters that minimizes the difference between the simulated performance and the history data of the field. The problem is modeled as a minimization problem and there are different approaches for solving it.

Traditional Newton-like methods have been used before in [2]. Other work, based on singular value decomposition (SVD) was shown in [3]. In addition, free-derivative methods based on Genetic Algorithms were proposed in [4] and were compared to derivative methods in [5]. This work presents a performance comparison between Genetic Algorithms (GA) and Differential Evolution (DE) for solving the history matching inverse problem. Both algorithms belong to the class of Evolutionary Algorithms, which are population-based optimization techniques. More details about each method will be provided in Section 4.

This work is organized as follows: Section 2 introduces the direct problem formulation. Section 3 introduces the history matching theory. Section 4 presents the theory behind Evolutionary Algorithms and details concerning GA and DE. Section 5 presents the numerical experiments performed in this work. Section 6 presents results and Sections 7 presents the conclusion of this work.

2 Reservoir Simulator: The Forward Problem

The problem treated in this paper is a two dimensional two-phase (water/oil) immiscible and compressible porous media flow in an environment with gravity. The system of partial differential equations which governs this flow is derived from the *law of mass conservation* and the *Darcy Law*. The law of mass conservation for both phases is written as

$$\phi \partial_t(\rho_\alpha s_\alpha) + \nabla.(\rho_\alpha v_\alpha) = Q_\alpha, \tag{1}$$

where $\alpha = w$ denotes the water phase, $\alpha = o$ denotes the oil phase, ϕ is the porosity of the porous medium, and ρ_α, s_α, v_α and Q_α are, respectively, the density, saturation, volumetric velocity and flow rate in wells of the α-phase. The volumetric velocity (v_α) is given by the Darcy law and can be written as:

$$v_\alpha = \frac{K k_{r\alpha}(s_\alpha)}{\mu_\alpha}(\nabla p_\alpha - \rho_\alpha g \hat{D}), \tag{2}$$

where K is the effective permeability of the porous medium, $k_{r\alpha}$ is the relative permeability of α-phase, which is a function that depends on saturation, μ_α and p_α are, respectively, viscosity and pressure of the α-phase, g is the gravity acceleration and $\hat{D} = (0, 0, -1)$. We also have that

$$s_w + s_o = 1. \tag{3}$$

We introduce the phase mobility and transmissibility functions, respectively:

$$\lambda_\alpha(s) = \frac{k_{r\alpha}(s)}{\mu_\alpha}, \tag{4}$$

$$T_\alpha(s) = K\lambda_\alpha, \tag{5}$$

where $s = s_w$ from now on. The volumetric velocity can then be written as $v_\alpha = -T_\alpha \nabla p_\alpha$.

The capillary pressure is modeled by

$$p_w = p_o - cp(s_w). \tag{6}$$

Since the saturation and pressure of one phase is determined by the value of the same unknowns of the other phase, the problem will be solved with regard to water saturation (s_w) and oil pressure (p_o).

We assume that the density of the phases and the porosity of the reservoir rock are dependent on the oil pressure. Finally, we consider no flow boundary condition, $v_\alpha.\nu = 0, x \in \partial\Omega$, where ν is the outer unit normal to the boundary $\partial\Omega$ of the domain Ω, and the initial condition is given by $s(x,0) = s_0(x), x \in \Omega$.

The final system of partial differential equations used in this work to model the fluid flow on porous media is given by

$$\begin{cases} \partial_t \phi \rho_\alpha s_\alpha + \nabla.(\rho_\alpha v_\alpha) = q_\alpha \rho_\alpha, \\ v_\alpha = -T_\alpha(s_w)(\nabla p_\alpha - \rho_\alpha g \hat{D}), \\ s_o + s_w = 1, \\ p_w = p_o - p_c(s_w), \\ \phi = \phi(p_o), \\ \rho_l = \rho_l(p_o), \\ v_\alpha.\nu = 0, x \in \partial\Omega, \\ s(x,0) = s_0(x), x \in \Omega. \end{cases} \tag{7}$$

There are different numerical methods that can be used to solve system (7). The simulator used in this work implements a fully implicit method.

3 History Matching

History matching is a well known inverse problem in the oil industry [6] . In the forward problem, the physical properties of the reservoir are known and a

simulator is used to calculate the production behavior of the reservoir. In the inverse problem (history matching), the goal is to estimate plausible reservoir physical properties, given the observed production data of a real reservoir. The forward and inverse problem schemes can be exemplified by Figure 1.

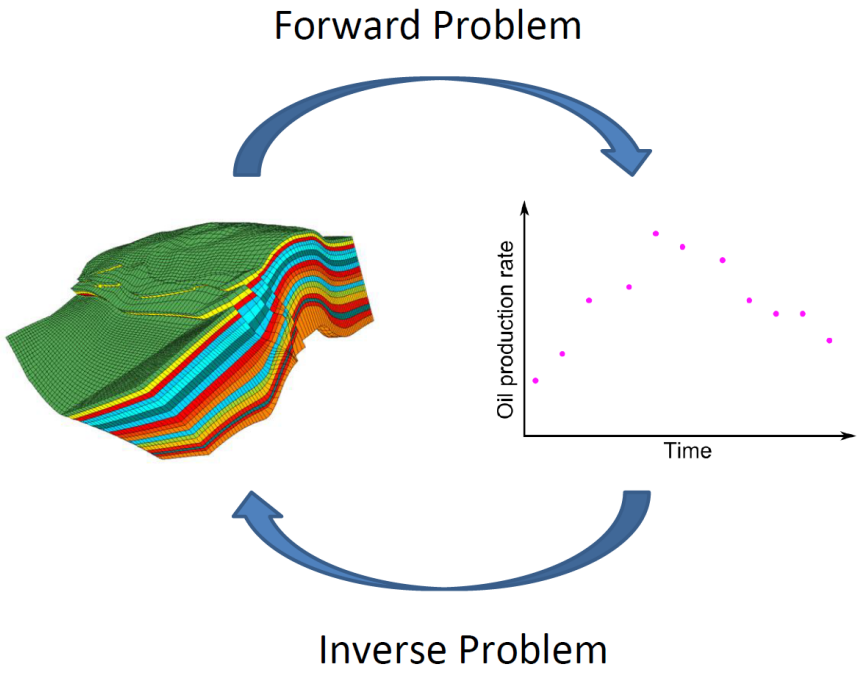


Fig. 1. History Matching: an inverse problem

Since physical properties of the reservoir cannot be directly measured in all extensions of the reservoir, the history matching process is used to estimate such properties. The estimated properties are used as the simulator's parameters with the goal of predicting the reservoir behavior under different production scenarios.

In this work, the inverse problem proposed aims to estimate the absolute permeability field of a reservoir by history-matching its production data, which is given by the oil rate and bottom-hole pressure measure at well locations from time to time. We denote by K the vector of permeability to be determined, by $S(K)$ the vector of simulated data given the parameter K and by $\bar{O}$ the vector of observed data. The problem consists on searching K that minimizes the least square formulation

$$f(K) = \|S(K) - O\|^2. \tag{8}$$

The problem to solve is then formulated as a minimization problem

$$\min_{K} f(K). \tag{9}$$

In the context of Evolutionary Algorithms, $f(K)$ is called *fitness function* on dealing with Evolutionary optimization algorithms. Its importance will be elucidated in the following sections. In this work we transform the fitness function in a relative error measurement, as follows

$$f(K) = \frac{\|S_o(K) - O_o\|^2}{\|O_o\|^2} + \frac{\|S_p(K) - O_p\|^2}{\|O_p\|^2}, \tag{10}$$

where subscripts o and p denote oil rate and bottom-hole pressure observations, respectively.

The following sections will introduce EAs and, more specifically, Genetic Algorithms and Differential Evolution, the methods used in this work to solve the minimization problem (9).

4 Evolutionary Algorithms

Evolutionary Algorithms (EA) are stochastic optimization methods inspired on Darwin's Evolution Theory and Natural Selection [7]. There are many different variants of EAs and the common underlying idea behind all these techniques is the same: given a population of individuals, the environmental pressure causes natural selection (survival of the fittest), which causes a rise in the fitness of the population. Given a quality function to be maximized, we can randomly create a set of candidate solutions, i.e., elements of the function's domain, and apply the quality function as an abstract fitness measure - the higher the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (the so-called parents) and results one or more new candidates (the offspring). This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached [8]. This process is exemplified by the pseudo-code 1.

Algorithm 1 EA pseudocode

```
Initialize population;
Evaluate individuals;
while stop criteria not met do
    Select individuals;
    Recombine individuals;
    Mutate individuals;
    Evaluate individuals;
end while
```

Selection, recombination and mutation are the three basic operators that steer EAs and the way they are implemented is what usually differ the variants of EAs techniques. The following sections will present a more detailed discussion about these operators on GA and DE.

4.1 Genetic Algorithms

Genetic algorithm was initially introduced by Holland as a means of studying adaptive behaviors [9]. However they have largely been considered as optimization methods since then. GAs follow the EA scheme presented previously. Traditionally, individuals are represented using a bit string, what requires a function to map each string to the parameter set to be optimized. However, since GA's invention, different approaches have been proposed to represent the individuals. The approach applied in this work is a floating point representation. It is an alternative representation in optimization problems with real-valued continuous variables, as is the case in this work. With this representation, there is no need for an explicit encoding mechanism. Each member of each population in the genetic algorithm is a floating-point vector. The genetic operators (mutation and crossover) in this case do not handle bit strings and are defined in a different manner.

Different ways of performing selection, mutation and recombination have also been proposed so far. The following sections will describe the operators used in this work.

Roulette Wheel Selection In the roulette wheel selection approach an individual of the current population has a probability of being selected that is proportional to its fitness value. Thus, a more fitted individual is more likely to be selected, but a bad individual still has its chance, what is important to prevent the algorithm to quickly converge to local minima.

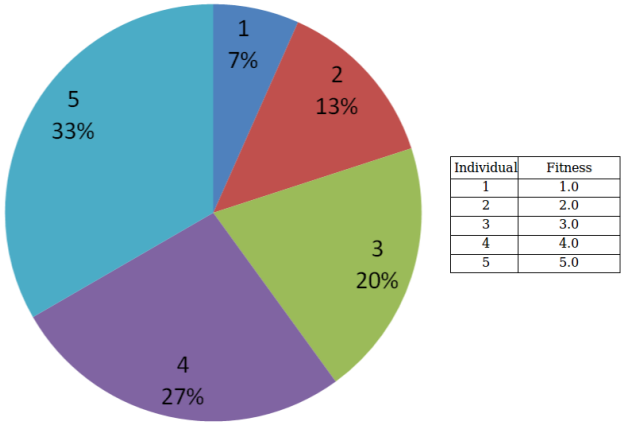
The probability of selecting a particular individual i in a maximization problem is given by

$$p_i = \frac{f_i}{\sum_{j=1}^{Npop} f_j}, \tag{11}$$

where f_α is the fitness of individual α and $Npop$ is the size of the population. When dealing with minimization problems, which is the case of this work, instead of using f_j we use $\frac{1}{f_j}$.

This selection process can be compared as spinning a roulette wheel where its sectors are set equal to the probability of each individual. Figure 2 shows an example of this selection using 5 individuals.

In this work we used $\alpha = 0.36$.



Individual	Fitness
1	1.0
2	2.0
3	3.0
4	4.0
5	5.0

Fig. 2. Roulette wheel selection example

Blend Crossover The Blend-crossover (BLX-α) [10] generates two offspring from two individuals (parents). It randomly picks values that lie between two points that contain the two parents, but may extend equally on either side determined by a user specified parameter α, as shown on Figure 3.

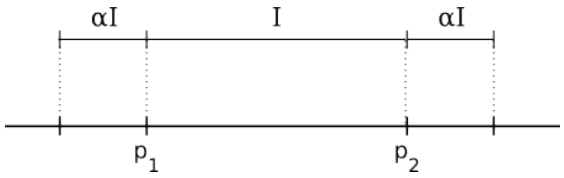


Fig. 3. Blend-crossover space. (p_1 and p_2 are the two parents)

Algorithm 2 presents a pseudo-code of the Blend-crossover.

Algorithm 2 Blend-crossover

Input: parents p_1 and p_2
Output: offspring o^1 and o^2
for each parameter i **do**
  $d_i = \|p_i^1 - p_i^2\|$
  $X_i^1 = min(p_i^1, p_i^2) - \alpha d_i$
  $X_i^2 = max(p_i^1, p_i^2) + \alpha d_i$
  $o_i^1 = rand(X_i^1, X_i^2)$
  $o_i^2 = rand(X_i^1, X_i^2)$
end for

Mutation The mutation used in this work consists on perturbing a parameter p_i of an individual in the following way

$$p_i = (1 + a) * p_i, \tag{12}$$

where $a = rand(-0.5, 0.5)$.

Elitism When only offspring vectors are allowed to advance, there is no guarantee that the best-so-far solution will not be lost. Retaining the best-so-far solution is known as *elitism* and it plays an important role on the successful convergence of the algorithm. In this strategy, the best individual(s) of the previous generation is kept in the next generation if none of the offspring is better than it. In this work this strategy is applied and the number of best individuals kept is a parameter of the algorithm.

4.2 Differential Evolution

Differential Evolution (DE) was first introduced by Price and Storn [11] as a new EA technique. Since its invention it has been applied to numerous optimization problems and has presented very satisfactory results. Like nearly all EAs, DE is a population-based optimizer that begins with a randomly chosen initial population, whose parameters lie within preset parameter bounds. The operates through the same computational steps as employed by standard EAs presented previously. However, unlike traditional EAs, DE employs difference of the parameter vectors to explore the objective function landscape. This characteristic appears on the *mutation* operator.

The next sections will be devoted to explaining how mutation, recombination and selection are defined on DE.

Mutation: the Heart of DE De mutates and recombines the population to produce a population of N_p mutant vectors. In particular, *differential mutation* adds a scaled, randomly sampled, vector difference to a third vector. Equation (13) shows how to combine three different, randomly chosen vectors to create a mutant vector, v_i

$$v_i = x_{r0} + F.(x_{r1} - x_{r2}). \tag{13}$$

The scale factor $F \in (0, 1^+)$, is a positive real number that controls the rate at which the population evolves. Where there is no upper limit on F, effective values are usually smaller than 1.0.

The base vector in $r0$ can be determined in a variety of ways, but in this work it was chosen randomly, with $r0 \neq i$. Except for being distinct from each other and from both i and $r0$, the difference vector indices $r1$ and $r2$ are also randomly selected [12].

Figure 4 illustrates the differential mutation in a 2D space.

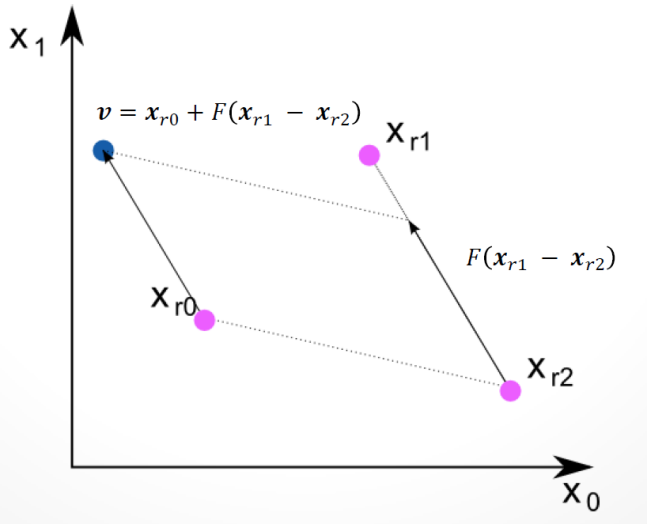


Fig. 4. Differential Mutation

Recombination To complement the differential mutation, DE also employs *uniform crossover/recombination*, which builds trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector of the current population with a mutant vector as follows

$$u_i = \begin{cases} v_{i,j}, & if(rand_j(0,1) \leq C_r) \\ x_{i,j}, & otherwise, \end{cases} \tag{14}$$

that is, a new individual will be a mixture between parameters on the mutant vector and on the current population vector. The crossover probability $C_r \in [0, 1]$, is a user defined value that controls the fraction of parameter values that are copied from the mutant.

Selection If the trial vector u_i has an equal or lower objective function value (for minimization problems) than that of its target vector x_i, it replaces the target vector in the next generation; otherwise, the target retains its place on the population for at least one more generation, as presented on Eq. (15).

$$x_{i,g+1} = \begin{cases} u_{i,g}, & if(f(u_{i,g}) \leq f(x_{i,g})) \\ x_{i,g}, & otherwise. \end{cases} \tag{15}$$

5 Numerical Experiments

The reservoir simulation we consider in this work is the classical five-spot configuration with 4 producer wells in the corners of the reservoir and one injection well in its center (see Figure 5). The reservoir is a square of sizes equal to $200m$ and $20m$ of depth. Each producer well produces a total of $100m^3$ per day. The reservoir's history is given by the oil production rate and bottom-hole pressure measured on the producer wells and on all wells, respectively, on 300 days of simulation. The other parameters of the model are: porosity (0.2), relative permeability, given by the Corey curve, irreducible water saturation ($s_{iw} = 0.2$) and residual oil saturation ($s_{ro} = 0.2$). The spatial discretization used was $\Delta x = \Delta y = 7.7m$.

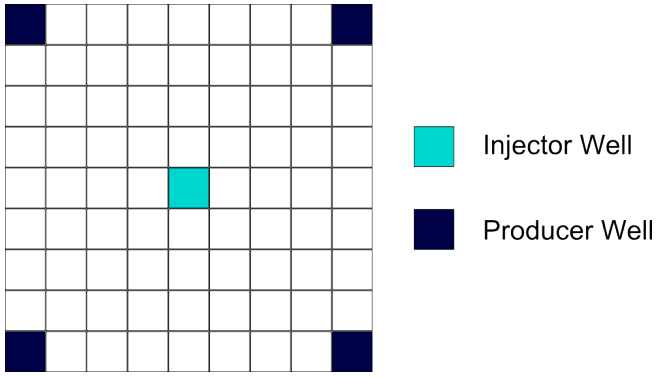


Fig. 5. 5-spot configuration

The history data was produced synthetically using the simulator model described in Section 2. Four different synthetic histories were generated differing on the number of different permeability regions on the reservoir. The number of these regions are 2, 4, 9 and 16 and the configuration of each model is shown on Figure 6.

For each case both GA and DE were applied. The algorithms have some parameters to be tuned. Parameters in common for both GA and DE are *population size* and *crossover rate* which were set to 30 and 0.90, respectively. Exclusive GA parameters are *mutation rate* and *elitism size*, set to 0.10 and 5 respectively. Parameter F of DE was set to 0.5. Initial populations were generated for each synthetic problem and the algorithms started from the same initial population. Stopping criteria are the maximum number of generations, 100, and the minimum fitness value, 1.e-6, whichever comes first.

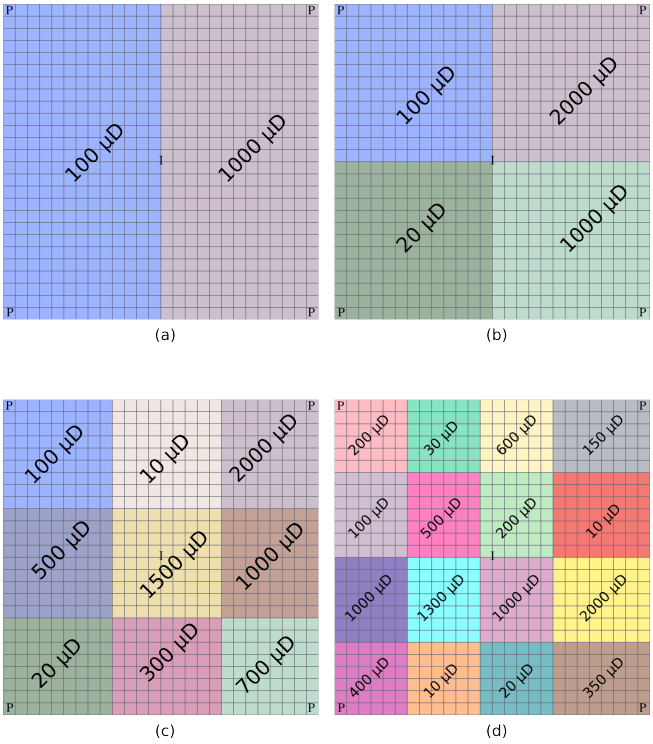


Fig. 6. Permeability field that generated 4 different histories. (a) 2 parameters, (b) 4 parameters, (c) 9 parameters and (d) 16 parameters.

6 Results

This section presents the results of the experiments described on Section 5. Figure 7 shows the fitness evolution for 2, 4, 9 and 16 parameters optimization problems. Both algorithms were executed 4 times for each experiment, and the best fitness in each generation of every execution is displayed on the graphs. Table 1 presents the best found fitness for each example.

Table 1. Best fitness after 100 generations

# parameters / **Method**	2	4	9	16
GA	9.05e-5	2.41e-2	3.43e-2	1.77e-1
DE	0.0	4.26e-7	1.37e-3	6.35e-2

We can observe that DE outperformed GA in every experiment presented in this work. In the problem of estimating 2 parameters, DE achieved a perfect

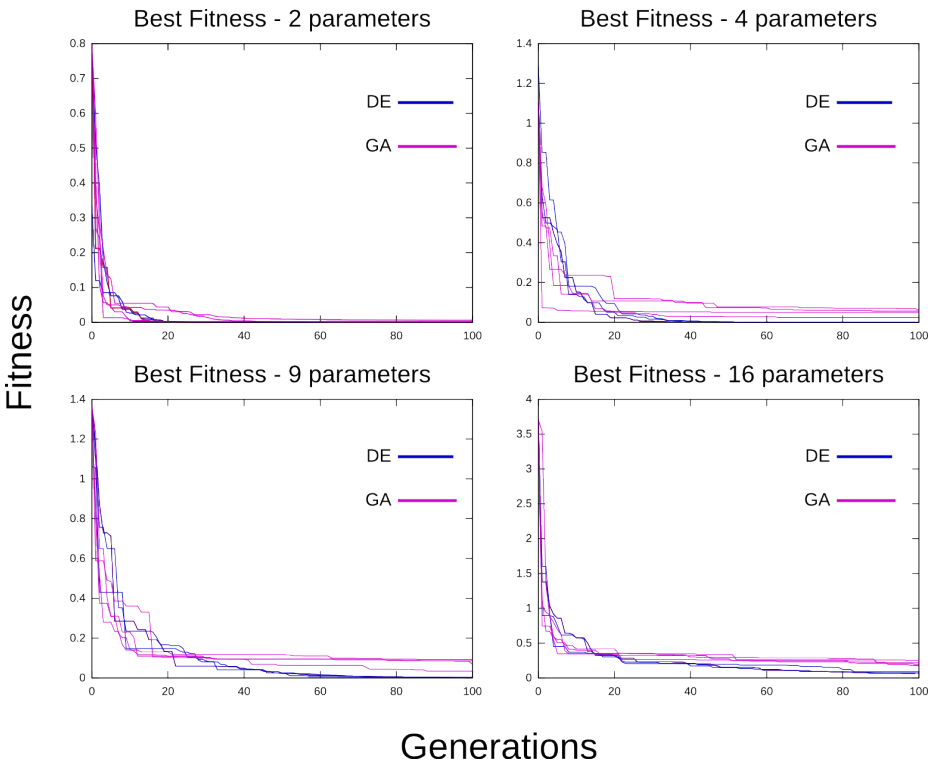


Fig. 7. Best fitness evolution throughout the generations. 4 runs of both GA and DE are shown for each case.

matching (fitness = 0.0) before the 60^{th} generation for every experiment, while the best run of GA got a fitness of $9.05e-5$. The superiority of DE was also obvious in the problem with 4 parameters, where the best individual of DE was more than 5×10^4 better than the best solution found by GA. Although this advantage was not that prominent on the cases with 9 and 16 parameters, DE still achieved better results than GA.

Despite the unquestionable superiority of DE, it is interesting to notice by Figure 7 that GA was more efficient in the early iterations than DE, that is, GA converged faster to a better solution in the beginning of the optimization process. However, once close to the region of a good solution, GA was not successful in refining this possible solution. On the other hand, DE was more consistent and would continue to refine the solution even when it was very close to the optimum.

Tables 2, 3, 4 and 5 present the best parameter set found by GA and DE. TP means *target parameters*, that is, the parameters used to generate the history and, consequently, the optimum known set of parameters.

The relative error between the target parameter vector and the parameters found by the algorithms is presented in Table 6.

Table 2. 2 Parameters

TP	100	1000
GA	99.9922	999.7467
DE	99.9999	999.9997

Table 3. 4 Parameters

TP	100	20	1000	2000
GA	98.8882	20.0627	1289.6820	2368.3789
DE	100.0000	19.9999	1000.0007	2000.0050

Table 4. 9 Parameters

TP	100	500	20	10	1500	300	2000	1000	700
GA	100.2492	366.4448	20.0922	12.4617	1710.3032	455.0980	2980.9579	1213.4672	1203.6730
DE	99.93170	520.9520	19.9958	10.0148	1451.7166	298.6878	2044.6256	966.92989	713.73589

Table 5. 16 Parameters

TP	200	100	1000	400	30	500	1300	10
GA	143.1093	70.8814	1355.3798	391.0537	191.8145	997.6883	2492.9848	335.9915
DE	163.4741	37.4271	92.3983	382.8372	299.3884	202.9935	1374.7098	890.7038
TP	600	200	1000	20	150	10	2000	350
GA	2622.9554	174.7876	429.8657	381.0928	164.7305	5.5504	7.1481	1111.63595
DE	140.5125	349.5064	234.4274	1282.0209	165.7923	16.3574	8.3219	493.5829

Table 6. Relative Error between target and found parameters

Method \ **# parameters**	2	4	9	16
GA	2.5e-4	2.0e-1	4.0e-1	1.12e+0
DE	3.0e-7	2.3e-6	2.7e-2	9.6e-1

7 Conclusion

This work presented a comparison between Genetic Algorithms and Differential Evolution for solving the history matching problem, a well known reservoir engineering inverse problem. DE outperformed GA in every experiment presented, proving to be a very powerful optimization method and a good alternative for applying to history matching. It is interesting to note that GA performed better than DE on the initial generations, that is, the fitness value of the best individuals dropped much faster in the beginning of GA than DE. But GA was unsuccessful on refining the solution. On the other hand, DE could refine very well the solutions and achieved impressive fitness values. DE is a little bit simpler to implement than GA. It could be interesting to combine both methods, creating a hybrid optimization system, that would employ GA techniques in the beginning of the process and DE would follow on refining the solution.

Acknowledgments

This work was been partially funded by CNPq, Petrobras and Foundation CMG (Alberta Innovates).

References

1. Watson, A. T., W.J.G., E., E.R.: Parameter and system identification for fluid flow in underground reservoirs. Proceedings of the Conference, Inverse Problems and Optimal Design in Industry (1994)
2. Brun, B., G.O.B.J.: Use of prior information in gradient-based history-matching. SPE Reservoir Simulation Symposium (2001) 13–23
3. dos Santos Amorim, E.P., Goldfeld, P., Dickstein, F., dos Santos, R.W., Xavier, C.R.: Automatic history matching in petroleum reservoirs using the tsvd method. In: ICCSA 2010. (2010) 475–487
4. Soleng, H.: Oil reservoir production forecasting with uncertainty estimation using genetic algorithms. Evolutionary Computation (1999)
5. Santos, E.P., Xavier, C.R., Goldfeld, P., Dickstein, F., Weber Dos Santos, R.: Comparing genetic algorithms and newton-like methods for the solution of the history matching problem. In: Proceedings of the 9th International Conference on Computational Science: Part I. ICCS '09, Berlin, Heidelberg, Springer-Verlag (2009) 377–386
6. Oliver, D.S., Reynolds, A.C., Liu, N.: Inverse Theory for Petroleum Reservoir Characterization and History Matching. 1 edn. Cambridge University Press (2008)
7. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press (1998)
8. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing (Natural Computing Series). Springer (2008)
9. Holland, J.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. (1992; First edition: 1975 The University of Michigan.)
10. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. Foundation of Genetic Algorithms 2 (1993) 187 – 202
11. Storn R, P.K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces (1995)
12. Kenneth Price, R.M.S., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). Springer (205)