Journal of Natural Gas Science and Engineering 27 (2015) 219–235

Contents lists available at ScienceDirect

Journal of Natural Gas Science and Engineering

journal homepage: www.elsevier.com/locate/jngse

3-D well path design using a multi objective genetic algorithm

Vahid Mansouri [a, *], Rassoul Khosravanian [a], David A. Wood [b], Bernt S. Aadnoy [c]

[a] Department of Petroleum Engineering, Amirkabir University of Technology, Tehran, Iran
[b] DWA Energy Limited, Lincoln, United Kingdom
[c] Department of Petroleum Engineering, Stavanger University, P.O. Box 2557, 4091, Stavanger, Norway

ARTICLE INFO

Article history:
Received 4 July 2015
Received in revised form
23 August 2015
Accepted 24 August 2015
Available online 31 August 2015

Keywords:
Multi-objective optimization
Genetic algorithm
Drilling torque
Wellbore trajectory planning
Oil and gas well design

ABSTRACT

Optimizing wellbore trajectories to reach an offset subsurface location, involving a complex combination of vertical, deviated and horizontal well components, requires the minimization of both wellbore length and frictional torque on the drill string. This is particularly the case for shallow horizontal wells which are often limited in their extent by torque. By minimizing both wellbore length and torque it is likely that a wellbore designed to reach a specific target can be drilled more quickly and cheaply than other potential trajectories. However, these two objectives are often in conflict with each other and related in a highly non-linear manner. A multi-objective genetic algorithm (MOGA) methodology is developed and applied with two objective functions, viz. wellbore length and torque, to develop a set of Pareto optimal solutions that can aid the selection of less risky/less costly well trajectory designs. The MOGA performance is compared with single-objective function studies of a specific wellbore scenario. The results indicate that the MOGA methodology outperforms single-objective function approaches leading to rapid convergence towards a set of Pareto optimal solutions. Analysis reveals that by adopting an adaptive approach that allows the behavioral parameters of the genetic algorithm (GA) to evolve as iterations progress, the MOGA proposed converges more rapidly toward better ultimate solutions than if the GA behavioral parameters are held constant over all iterations of the algorithm. Algorithm code listings for the MOGA and GA applied in the analysis presented are included as appendices.

1. Introduction

One of the most expensive operations involved in the exploration and development of oil and gas reservoirs is typically the drilling of the wellbores. In the prevailing market conditions of relatively high costs and low oil and gas prices across much of the world most oil and gas companies are particularly keen to minimize their drilling costs Khaled et al., 1999.

Previous studies indicate that the cost of drilling a horizontal well is about 1.4 times the cost of drilling a vertical well (S. D. Joshi, 2003). The attraction of drilling horizontal and directional wells is that they can contact a greater volume of the reservoir and can transect the highest quality zones more effectively than vertical wells, resulting in higher production and recovery rates. In both cases one of the most important factors affecting the cost of drilling is length of the wellbore and the time taken to drill to the reservoir target. Thus any possibility to reduce the length of the wellbore, within the constraints of acceptable curvatures and geological obstacles, typically reduces the time it takes to reach the target and thereby reduces the total drilling costs. Optimizing wellbore lengths, subject to a defined set of constraints, typically is desirable as a means of improving the economics of drilling operations.

In the recent years, optimization has been used extensively across the petroleum industry for a variety of purposes, process plant optimization, transport scheduling and to various aspects of the drilling operation (e.g. Shokir et al., 2004; Atashnezhad et al., 2014; Guria et al., 2014). With respect to wellbore trajectory planning Shokir et al. (2004) used a genetic algorithm and Atashnezhad et al. (2014) used a particle swarm optimization algorithm with the single objective function of minimizing wellbore length, subject to a number of defined constraints. A recent application of multi-objective genetic algorithm (MOGA) to drilling is provided by Guria et al. (2014) in their application of a multi-objective optimization genetic algorithm to two- and three-objective functions related to determining optimum drilling variables related to an oil

* Corresponding author. Faculty Member of Drilling Engineering of Department of Petroleum Engineering, Amirkabir University of Technology, 424 Hafez Ave, P.O. Box: 15875-4413, Tehran, Iran. Tel.: +98 (21) 64545130.
E-mail addresses: vahid.mansouri1990@gmail.com (V. Mansouri), Khosravanian@aut.ac.ir (R. Khosravanian), dw@dwasolutions.com (D.A. Wood), bernt.aadnoy@uis.no (B.S. Aadnoy).

http://dx.doi.org/10.1016/j.jngse.2015.08.051
1875-5100/

field offshore Louisiana. That work applied an elitist non-dominated sorting genetic algorithm to (i) maximize drilling depth, (ii) minimize drilling time and (iii) minimize drilling cost with fractional drill bit tooth wear as a constraint. Another application of MOGA, relevant to petroleum field operations, is proposed by Yasari et al. (2013) to apply a non-dominated sorting genetic algorithm to find optimized and robust water injection policies for three injection wells.

In this paper a new MOGA optimization model for well trajectory planning is developed for optimizing drilling operations. The model optimizes the recommended well path by taking into account two-objective functions: 1) to minimize wellbore length and trajectory to a specified sub-surface target location (i.e. length); and 2) to minimize torque on the drill string during the drilling operation (i.e., torque).

We recognise that when designing wells at specific locations there are other factors that need to be taken into account in addition to wellbore length and torque (e.g., wellbore tortuosity and its influence on the ease or difficulty in running a specific well completion design; combined drilling and completion costs associated with drilling a particular well path; dealing with problem formations above a reservoir in a certain way, i.e., setting casing above it or below it certain specified points; entering the reservoir at a certain angle and penetrating it at a certain inclination). Some of these additional factors can be dealt with as constraints that selected optimum well paths need to achieve. It would also be possible to consider these as optimization objectives in their own right in the MOGA algorithm. In this study we focus on just the two key objectives, i.e., wellbore length and torque, to prove the benefits of the MOGA concept. We will be conducting future research to expand the MOGA methodology to consider some of these additional factors, both as constraints and additional objectives.

MOGA involves a process of developing a random set of potential solutions making up a population of solutions to be tested for fitness. Each individual solution generated is typically referred to as an individual or gene. The population is subjected to a series of evolutionary iterations (i.e., developing new solutions/genes by genetic processes such as mutation and crossover with different characteristics in each generations), with the solutions being tested for fitness and ranked in each generation, and the most fit for purpose being carried forward to the next generation. This process means that each generation progresses, or converges, towards the best set of solutions. Finally, the best solutions are identified and their performances compared in relation to the multiple objective functions. In this study the MOGA is coded using MATLAB software, with the detailed code provided in Appendix A. The model is applied to the horizontal well drilling trajectory scenario (based on a real well drilled in Egypt) studied for trajectory optimization purposes by Shokir et al. (2004), using a single-objective function genetic algorithm, and Atashnezhad et al. (2014) using particle swarm optimization. We describe how our MOGA model works in detail and provide an analysis of the results for the specific, complex, wellbore trajectory selected.

The case study is based on a real well drilled in Egypt, and previously used to demonstrate wellbore trajectory optimization using a genetic algorithm by Shokir et al. (2004) applied to the deviated well trajectory calculation algorithm originally proposed by Adams and Charrier (1985), using the length of the wellbore between specified surface and bottom-hole locations as the objective function.

2. Mathematical formulation

With the dual objectives of finding optimal solutions that provide the minimum well length, honoring the wellbore constraints imposed, so that the torque on the drill string is also minimized, it is necessary to provide mathematical formulations for those two clear objective functions, i.e.: 1) the wellbore length; 2) the torque on drill string while rotating. These two objective functions are described in mathematical formulations in the following sections.

2.1. Well path length

Fig. 1 illustrates all the components involved in the calculation of the length of each curved section of a directional wellbore. There are several methods available to measure the wellbore length of a directional well. Our formulation uses the radius of curvature method. In the radius of curvature method, constant curvature between two points and the radius of the curvature is given by equations (1)–(3):

$$a = \frac{1}{\Delta m}\sqrt{(\theta_2-\theta_1)^2 \sin^4\left(\frac{\varphi_2+\varphi_1}{2}\right)+(\varphi_2+\varphi_1)} \tag{1}$$

$$r = \frac{1}{a} = \frac{180^*100}{\pi^*T} \tag{2}$$

$$\Delta m = r\sqrt{(\theta_2-\theta_1)^2 \sin^4\left(\frac{\varphi_2+\varphi_1}{2}\right)+(\varphi_2+\varphi_1)} \tag{3}$$

Based upon the wellbore trajectory illustrated in Fig. 2, with multiple curved and linear components, the total measured depth of the wellbore can be obtained from equation (4):

$$TMD = D_{KOP} + D_1 + D_2 + D_3 + D_4 + D_5 + HD \tag{4}$$

Detailed formulas to calculate the non-vertical and horizontal components of the wellbore trajectory (D_1, D_2, … D_5) are provided in Appendix A.

2.2. Torque

Torque and drag analysis in relation to wellbore tubulars has been studied in detail and is now well understood (e.g., Sheppard

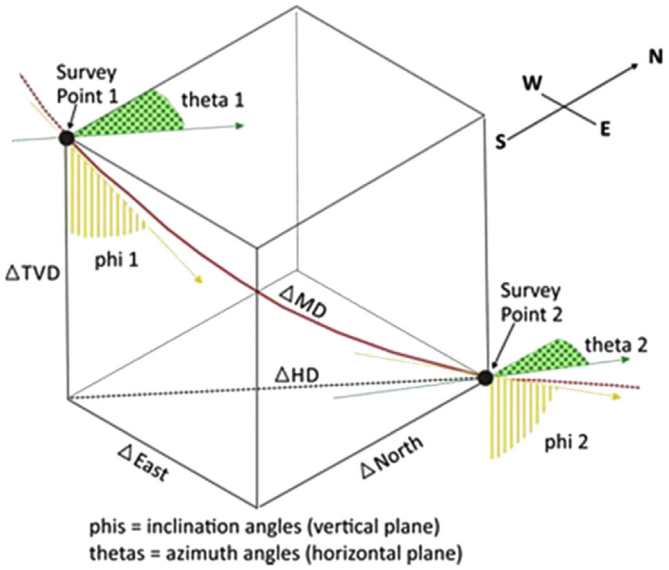

Fig. 1. Calculation of the length for a deviated section of the well trajectory after Atashnezhad et al. (2014) describes the terms used to define the different angles and components of the wellbore trajectory. MD = measured depth; TVD = true vertical depth.

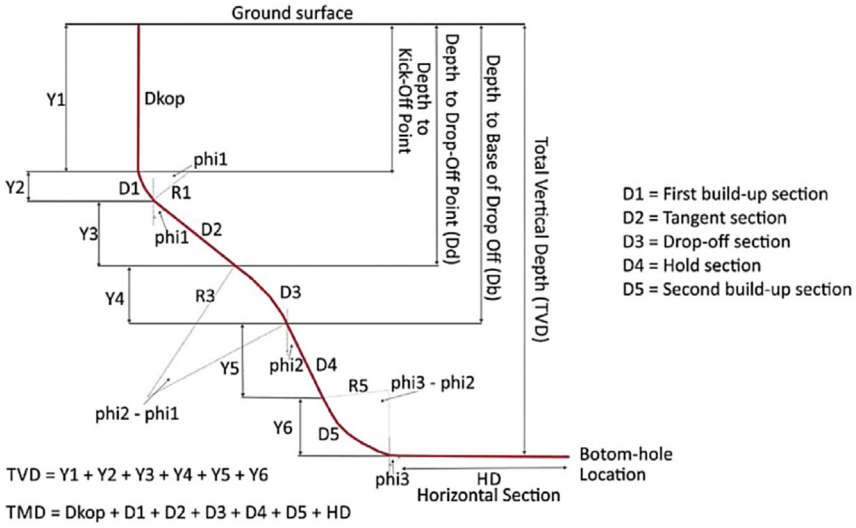

Fig. 2. The vertical plane of a horizontal well with the operational parameters from Atashnezhad et al. (2014) developed from the wellbore scenario studied originally by Shokir et al. (2004). Note that the scenario involves more than one build section and a drop-off section separating the build sections. The wellbore trajectory formulation incorporates all the sections identified in this diagram.

et al., 1987; Xie et al., 2012; Aadnøy et al., 2010). The objective is to prevent drilling and completion problems before they occur (Xie et al., 2012), and this is particularly necessary in wellbores with curved and non-vertical sections as drag and torque loss are associated with directional wells. Drag is experienced when the drill string is moved up or down the wellbore, for example when tripping in and out of the hole. The sources of drag and torque losses include: differential sticking, key seating, poor hole cleaning, and sliding wellbore friction.

Optimal well path design, from the perspective of torque and drag, should minimize all these effects (Johancsik et al., 1984; McCormick and Chiu, 2011), such that:

✓ Normal forces acting on the drill string are reduced
✓ Dynamic conditions are enhanced (static conditions minimized)

To calculate the torque on a drill string the soft string model is used as the formulation for the torque objective function. This assumes that the drill string takes the form of a heavy cable lying in the wellbore, and any tubular stiffness effects due to the drill pipes is ignored. Calculations consider only the state in which the drill string is rotating, without any axial movement up or down. Formulas are provided for two general cases: equation (5) for a straight section; and, equation (6) for a curved section.

$$F_2 = F_1 + Bw\Delta L \cos\varphi \tag{5}$$

$$T = \mu r w \Delta L \sin\varphi \tag{6}$$

In a straight section the tension along the drill pipe has no effect on the normal pipe force, and hence, no effect on friction. Straight sections are weight dominated, as only the normal weight component generates friction. Fig. 3 shows the force balance for a straight pipe. Equation (5) calculates the change in axial load, and equation (6) calculates the torque, on a straight section of drill pipe. Fig. 4.

For the curved sections, the normal contact force between string and hole depends strongly on the axial loading within the pipe. So, for the calculation of the torque, the tension in the pipe is required. Other parameters that contribute to this calculation are those angles determining the total change in direction. These angles and forces are illustrated and defined in Figs. 5 and 6. They are derived via the relationships expressed by equations (7) and (8):

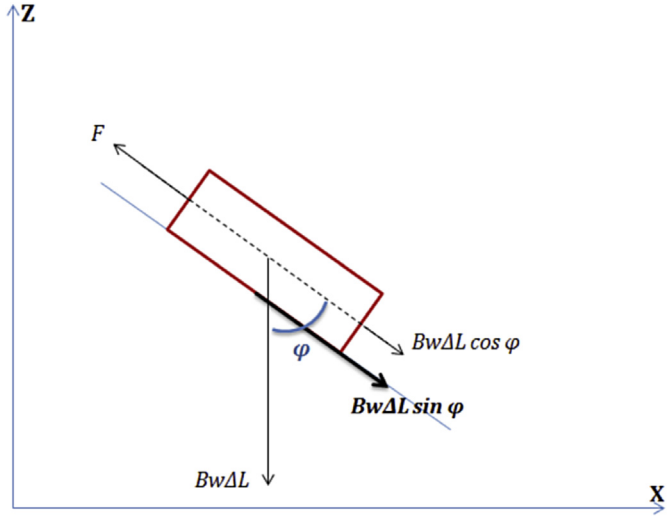

Fig. 3. Force balance for pipe pulling along a straight surface (From Aadnøy and Andersen, 2001).

$$e_1 \cdot e_2 = |e_1||e_2|\cos\beta = \cos\beta \tag{7}$$

$$\cos\beta = \sin\varphi_1 \sin\varphi_2 \cos(\theta_1 - \theta_2) + \cos\theta_1 \cos\theta_2 \tag{8}$$

The angle β is the total directional change. If both inclination and azimuth are changed, the plane that β acts in is not constrained to the horizontal or vertical plane (Fazaelizadeh, 2013). Having determined the axial loading in the drill pipe in equation (9) we are able to calculate the torque, which is expressed by equation (10):

$$F_2 = F_1 + Bw\Delta L\left(\frac{\sin\varphi_2 - \sin\varphi_1}{\varphi_2 - \varphi_1}\right) \tag{9}$$

$$T = \mu r F_1 \beta \tag{10}$$

Given that the wellbore trajectory in the scenario analyzed consists of seven parts, the torque must be calculated for each part separately, and summed to provide the total torque. The value of T

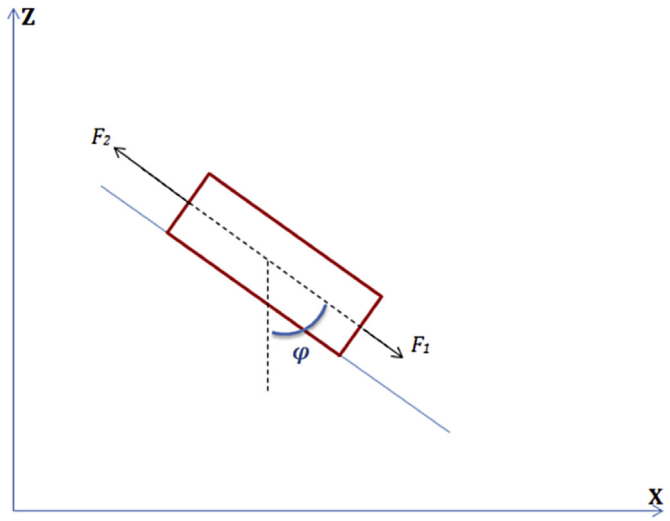

Fig. 4. Axial loading on a straight section (From Aadnøy and Andersen, 2001).

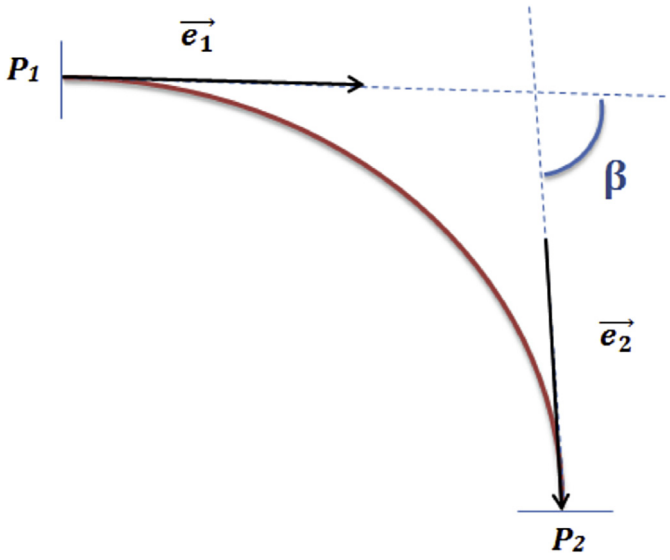

Fig. 5. Total Direction change and unit vectors e_1 and e_2 at two survey points P_1 and P_2 (from Fazaelizadeh, 2013).

from the above relation is the second objective function expressed by equation (11):

$$T = T_{vertical} + T_1 + T_2 + T_3 + T_4 + T_5 + T_{horizontal} \quad (11)$$

The calculation of the torque commences from the bottom of the drill string (i.e., when the drill string is at the wellbore's total depth (TD) then the T calculation starts at $T_{horizontal}$), and continues, stepwise, upward to the well head (i.e., last component added to the calculation is $T_{vertical}$). For the wellbore scenario evaluated the torque calculation involves the following assumptions:

1. The drill string has no axial movements (just rotation).
2. The drill string has 0.1 ft radius and 0.3 kN/ft weight.
3. The friction factor is 0.2 and the buoyancy factor is 0.7.

The detailed calculation of torque for the wellbore trajectory scenario studied is provided at Appendix A.

For the optimization calculation two types of constraints are applied to the wellbore trajectory and torque calculations: 1)

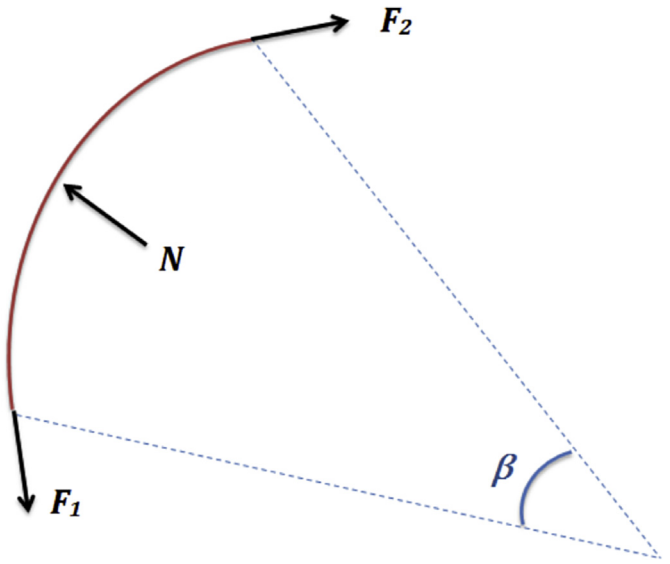

Fig. 6. Axial loading on a curved section during total change β in direction (from Fazaelizadeh, 2013).

Operational constraints, e.g., the rate of hole-angle build and rate of hole-angle drop off, and angle-hold wellbore section 2) Non-negative (logical) constraints, e.g. the measured depth and true vertical depth cannot be negative and be accepted as valid solutions to the wellbore optimization problem. Table 1 describes the optimization operational constraints imposed on the wellbore scenario evaluated. 1. The case study surveyed in this study has been extracted from "E. M. Shokir et al., 2004", so all information and limitations about it are from that paper. However the provided algorithm in this article is capable to include any other limitation such as special range for DLS in a particular formation or any constraint about the reservoir or other formations.

3. Multi-objective optimization

Multi-objective optimization is now used widely to aid decision making and selection of alternatives in many industries. Faster computation speeds, greater computer memory space and more understanding and accessibility to the algorithms is leading to expansion of their applications to solve common operational, financial and planning problems. In many real projects, there are multiple objectives that are inconsistent with each other, i.e., they act independently of each other, or in conflict with each other to an extent, in delivering high performing outcomes for the project.

Table 1
Operational restrictions.

TVD	Min. = 10850 ft. Max. = 10900 ft.
HD	2500 ft
Dogleg severity	$T_1, T_2, T_3, T_4, T_5 \leq 5°$/100 ft.
Min. value of inclination angles	$\varphi_1 = 10°$ $\varphi_2 = 40°$ $\varphi_3 = 90$
Max. value of inclination angles	$\varphi_1 = 20°$ $\varphi_2 = 70°$ $\varphi_3 = 95$
Min. value of azimuth angles	$\theta_1 = 270°$, $\theta_2 = 270°$, $\theta_3 = 270°$ $\theta_4 = 330°$, $\theta_5 = 330°$, $\theta_6 = 355°$
Max. value of azimuth angles	$\theta_1 = 280°$, $\theta_2 = 280°$, $\theta_3 = 280°$ $\theta_4 = 340°$, $\theta_5 = 340°$, $\theta_6 = 360°$
Kick-off point depth	Min. = 600 ft. Max. = 1000 ft.
Draw down point depth	Min. = 6000 ft. Max. = 7000 ft.
Third build point depth	Min. = 10000 ft. Max. = 10200 ft.
1st Casing setting depth	Min. = 1800 ft. Max. = 2200 ft.
2nd Casing setting depth	Min. = 7200 ft. Max. = 8700 ft.
3rd Casing setting depth	Min. = 10300 ft. Max. = 11000 ft.

Each objective function also typically has a number of constraints applied to it (e.g. Table 1), and this increases the complexity of the problem. Multi-objective optimization (MOO) requires simultaneous optimization of conflicting objective functions that may compete with each other in preventing the identification of a single global optimum solution (e.g., Yasari et al., 2010). In such problems, usually there is not a solution that results in all the objective functions being brought to optimum states. Rather, it is necessary for algorithms to search for solutions whereby the multiple objectives collectively occur in the most optimal state. In practice there often are multiple solutions that equally satisfy the collective "most" optimal state for the multiple objective functions evaluated, i.e., better performance in some objective functions trading off against poorer performance in other objective functions for a particular solution. Such multiple-optimum solutions are commonly referred to as Pareto optimal solutions in MOO (e.g., Haupt and Haupt, 2004).

Consider a multi-objective optimization problem with a minimization function, F(x), involving n individual objective functions ($f_1 \ldots f_n$) that is defined as follows:

$$\min F(x) = \{f_1, f_2, \ldots, f_n\}$$

where, $f_i(x) =$ ith objective function of n objective functions to be minimized

Each objective function is subjected to specific constraint functions, G(x) and H(x), with values specified:

$$G(x) < 0, H(x) = 0$$
$$x \in R$$

Each individual solution, x, is one of a population of solutions that achieves the overriding objective of the calculation (e.g. reaches the target location specified for the well bore).

The multi-objective goal is to minimize the individual objective functions, $f_1, f_2 \ldots, f_n$ simultaneously; recognising that a single solution satisfying all objective functions is unlikely to exist, an algorithm needs to be formulated to reveal the Pareto optimal solutions. To achieve this, a multi-objective algorithm must distinguish between "dominated" and "non-dominated" solutions.

If all objective functions are striving to achieve minimum solutions, an acceptable solution *j* dominates solution *i*, if and only if, for any i, $f_i(x) \leq f_j(x)$ and at least for one function $f_i(x) < f_j(x)$. If a solution is not dominated by any other solutions it is called a Pareto optimal solution or a non-dominated solution. Such a set of Pareto optimal solutions is called a Pareto optimal set. For any two objective functions the Pareto set may be displayed graphically as a Pareto frontier; for multiple objective functions the Pareto set may be displayed graphically as a Pareto surface. In some problems the set of Pareto optimal solutions can be very large (possibly infinite, i.e. all acceptable solutions in the population are non-dominated). A Pareto frontier for a two-objective optimization (minimizing) is shown schematically in Fig. 7 that clearly describe the concept of Pareto front and Pareto optimal solutions. The initial goal of a MOO is to find the Pareto optimal set; the final objective is to use that non-dominated set to inform decisions as to which non-dominated solution(s) in that set should be selected for operational purposes.

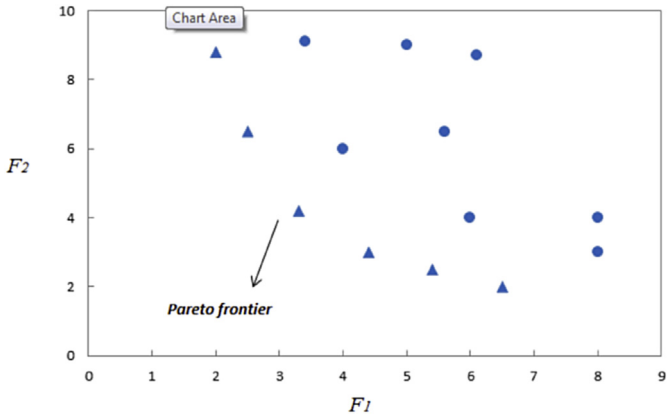

Fig. 7. Pareto optimal solutions concept for two objective functions, F_1 and F_2, to be minimized. The triangles represent non-dominated solutions, i.e. there are no solutions better than these; minimum F_2 for a given value of F_1 and vice versa. The circles represent dominated solutions, i.e. there is at least one solution that performs better than these in terms of minimizing values of F_1 in relation to F_2. (Modified after Haupt and Haupt, 2004).

4. Genetic algorithm

Genetic algorithms (GA) are stochastic search algorithms based on the mechanisms of natural selection and applying processes similar to those observed in biological genetics (e.g. Gen and Cheng, 2008). GA typically commence with an initial set of random solutions, or selected (seeded) solutions, representing a "population" of solutions that satisfy all constraints to the problem and achieving the overriding target (e.g. the total depth of the well at the specified target location) Sivanandam and Deepa, 2008. Each solution in the population is sometimes referred to as a "chromosome", maintaining the analogy with biological genetics. The population is processed through a loop that first ranks the solutions in terms of their performance with regards to the objective function(s) and then subjects some of the best performing solutions to a series of genetic processes (e.g. mutation, crossover, etc.) to derive some new solutions for the next generation to add to the best performing solutions of the current generation, the best performing solutions are ranked, and so the loop continues repeatedly generating new populations of solutions, but always preserving the best performing solutions of the previous generation. For this process to work effectively, each generation, i.e., all chromosomes (solutions) in the population need to be tested for performance by a defined fitness test scoring assignment system.

The solutions in a particular generation are ranked using their individual fitness scores. The solutions selected to be carried forward to the next generation have the highest fitness scores. The genetic algorithm loop continues through many iterations (generations) periodically finding better performing solutions (i.e., not all generations outperform previous generations, but based on fitness score selection, the highest ranking solution cannot perform worse than the previous generation). In some problems the algorithm may lead to convergence to an optimal solution, in other more complex problems, the GA may not find all solutions (i.e. there may be several isolated minima in the solution space). In the second situation a set of the highest-performing solutions are collected after a set number of iterations (GA generations), or after a specified period of computer processing time when no improved solutions are found.

The MATLAB code listing for the single-objective GA used in this study to solve deviated wellbore trajectories using the radius of curvature method is included as Appendix C.

GA is applicable to many multi-objectives optimization problems, because it is able to cope with conflicts and/or non-linear relationships among the multiple objectives, and provides a robust set of high-performing (not necessarily the best) solutions where multiple solutions exist (e.g., Yasari et al., 2013). The multi-objective genetic algorithm (MOGA) follows the same methodology as a single objective function GA, but defines its fitness score assignment system to involve all the objective functions involved. MOGA algorithms typically apply a zonal ranking scheme by firstly

identify the non-dominated solutions in each generation among all solutions in that generation. Such non-dominated solutions are assigned to rank one and are allocated the highest fitness score. All solutions in the same generation, excluding those assigned to rank one, are searched again to establish the non-dominating solutions among that set of solutions, and those identified non-dominating solutions in that sub-population of solutions are assigned to rank two and are allocated a lower fitness score than those in rank two. This process is repeated to establish multiple ranks, with the non-dominated solutions included in each successive rank allocated a lower fitness score than the previous, higher rank of solutions identified. Eventually all solutions in a particular generation will be assigned a rank and appropriate fitness score. Selection of the parents from which the next generation of solutions are generated in MOGA is based on this fitness score. All other steps in the MOGA are the same in the GA.

A pseudo coding of the MOGA used in this study is as follows:

```
Procedure: MOGA
Input: problem data, GA parameters
Output: Pareto optimal solutions
POP=Initialize(t=0);
While (not terminating condition) do
        F=Objectives( POP ) ; Calculate Objective Functions
        RANK=Evaluate( POP, F); Evaluation and Ranking by  ftness score assignment routine
        PARETO=Pareto(POP, F); Create Pareto Optimal Solutions and Pareto Frontier
        PARENTS=Select (POP, RANK); Select Parents for next generation
        POP=Xover(POP,PARENTS); Apply Crossover to select new high-performing solutions for next
        generation to replace low-ranking solutions
        POP=Mut(POP,PARENTS); Apply Mutation to generate other new solutions from the parents and replace
        the modified solutions with them
End
F=Objectives( POP ) ; Calculate Objective function values for solutions after V iterations or convergence
RANK=Evaluate( POP, F); Rank solutions according to fitness score
PARETO=Pareto(POP, F); Select rank-one solutions
Show PARETO  List rank-one solutions, Display graphically rank-one solutions for selected objective functions (i.e.
2D or 3D plots)
End
```

The MATLAB code listing for the MOGA used in this study to solve deviated wellbore trajectories using the radius of curvature method is included as Appendix B.

5. Solution and results

Considering the two objective functions for the wellbore scenario described above, and applying all specified constraints, the MATLAB codes for the two-objective optimization using MOGA are listed in Appendix B. These codes follow the sequence illustrated in the flowchart depicted in Fig. 8.

For the solution, first we need the objective function that is considered as follows:

$$OBJECTIVE\_FUNCTIONS = \{Length, Torque\}$$

Constrained to

$$C_{1,\min} < C_1 < C_{1,\max}$$

$$C_{2,\min} < C_2 < C_{2,\max}$$

$$C_{3,\min} < C_3 < C_{3,\max}$$

$$TVD_{\min} < TVD < TVD_{\max}$$

where symbol C refers to Casing setting depth.

The optimization process for the scenario described seeks to minimize the multi-objective function defined above. The MOGA code commences with initializing. A random population consisting of 100 solutions is generated that all are in feasible region (i.e. satisfy the constraints and reach the specified well target). By entering a loop, the objective functions for each solution are calculated and based on the value of these functions, a fitness score is allocated to each of them. The Pareto optimal solutions in each generation are assigned to rank one. At least some of the parent solutions for the next generation are selected. The selected parent solutions are then adjusted by a cross-over operator and new solutions are generated. The cross-over operator is designed to seek new solutions that might have higher fitness scores than some of the parents. The new improved solutions then replace low ranking solutions with low fitness scores (Low ranking solutions are replaced, so the best solutions are retained). The population then is subjected to a mutation operator generating an additional number of new solutions for the next generation. All parents undergo the mutation operator. And the new solutions based on their fitness compared to the parent fitness may be accepted or rejected. The mutation step is the last step of any iteration. In the analysis conducted for this study the process sequence or loop is repeated for 2000 iterations. Each step in an iteration described above is executed as a MATLAB function. The algorithm also includes a function to test the feasibility of each solution (i.e. does it reach the target location and satisfy the constraints). This function is required to prevent the inclusion of infeasible solutions in any generation of solutions to be ranked. This function ensures all solutions in each generation are in feasible region. Figs. 9 and 10 show the convergence of two objective functions during optimization. The results of the optimization for the defined wellbore scenario studied are listed in Table 2.

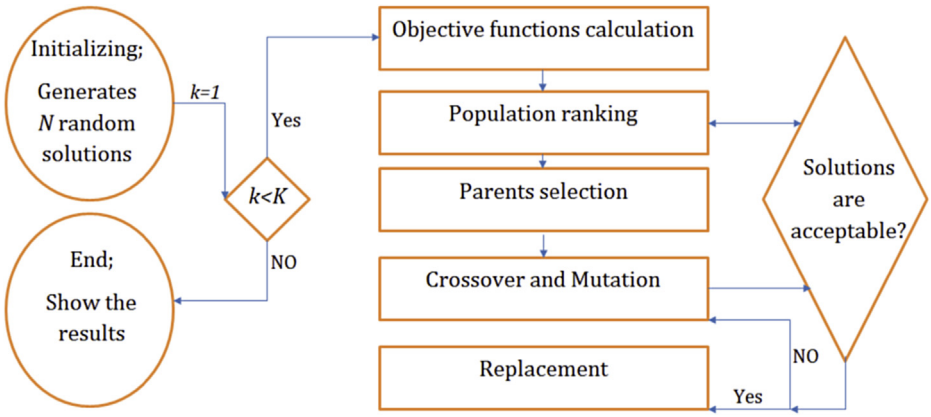

Fig. 8. Flow chart of process sequence applied in the multi-objective genetic algorithm (MOGA).

6. Discussion

The wellbore problem analyzed in this study has very complex nonlinear constraints. The objective functions themselves involve 17 variables. Maintaining all variables within their feasible regions and satisfying all constraints during the optimization requires additional coding to feed into the GA. Therefore, a customized section of coding integrated with the MOGA is necessary to obtain feasible and meaningful results. To achieve this goal, a constraint function therefore forms part of the algorithm applied. This function is called from all MOGA elements to assess whether specific solutions are feasible or not.

The initializing function ensures that the initial population is all in the feasible region and satisfies all specified constraints. As a further safeguard against infeasible solutions being propagated in the ranking function, while ranking, any potential individual solutions that are found not to satisfy all constraints, have their fitness scores decreased to zero. This ensures that infeasible solutions have no chance of selection for the next generation. The main GA operators (i.e. cross over and mutation) while producing new solution individuals are in communication with the constraint function in order to avoid producing unsatisfactory solutions.

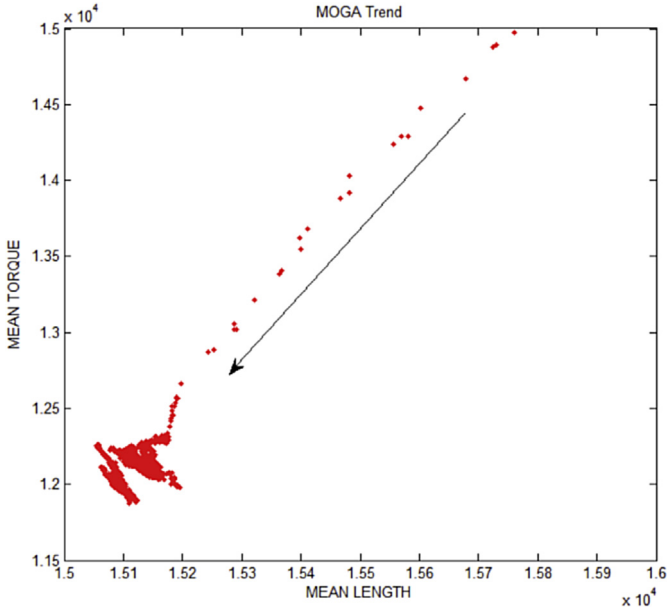

Fig. 9. Mean Torque versus Mean Length during Optimization with MOGA. The length scale is in units of ft $\times 10^{-4}$. The Torque scale is in units of N.ft $\times 10^{-4}$.

6.1. Genetic algorithm behavioral parameters

To be effective a GA needs to explore as much of the feasible space as possible (i.e. adopt a broad focus) and then target optimal zones within that space (i.e. adopt a more concentrated focus). A GA's behavioral parameters (i.e., population size of each generation, crossover probability and mutation probability) control it balance between a broad focus and a more targeted one. Selecting and modifying a GA's behavioral parameters are key issues that need to be addressed to improve the effectiveness and timing of convergence towards a useful set of optimal solutions (e.g., Gen and Cheng, 2008, Li, 2010). The initial population size (i.e., number of initial solutions that are generated at first by initializing function) is set high enough so that a wide region in the feasible solution space is encountered.

The crossover probability (P_C) determines the fraction of the population size (i.e. POP) in each iteration to produce new offspring via a cross-over mechanism. This probability controls the number (i.e., $P_C \times$ POP) of individuals in an iteration to undergo the cross-over operation. A high P_C encourages better exploration of the feasible solution space by the algorithm. This can avoid situations where the algorithm lock into a local optimum, but fails to find other better optima within the feasible solution space being searched. However, if P_C is too high it increases convergence time while the algorithm searches many sub-optimal areas of the feasible solution space.

The mutation probability (P_M) determines the fraction of the population for which new individuals are introduced via the mutation method in each iteration of the algorithm. A low P_M means that many individual characteristics that would have beneficial consequences are never evaluated. However, if P_M is too high then the next iteration differs too much in character from the previous iteration and the algorithm fails to benefit from the good evolutionary characteristics generated in previous iterations (Gen and Cheng, 2008).

Our methodology applies a parameter adaption approach to the values applied to the GA behavioral parameters mentioned above, rather than tuning the algorithm to select specific values for each of these behavioral parameters and then setting them as constant values in the algorithm Lin et al., 2003. This means that the values of the behavioral parameters applied in the algorithm vary as the GA evolves through its iterations, i.e. it follows an evolutionary process. Such variations facilitate optimum searching of the feasible

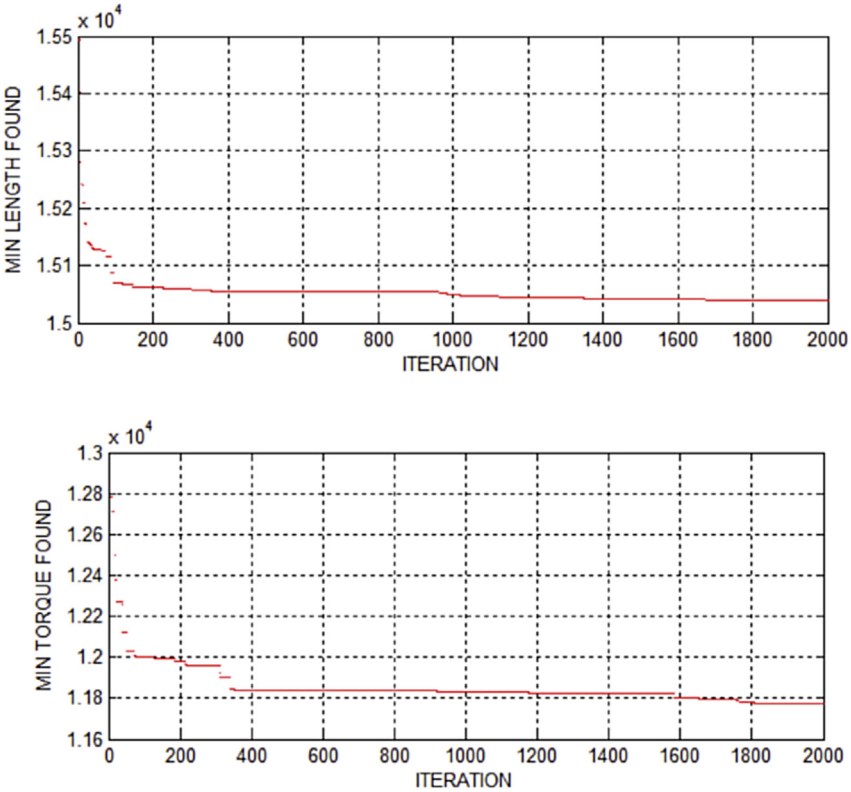

Fig. 10. Convergence of the two objective functions (Torque and Length) during MOGA optimization. Diagrams show the minimum values for torque and length found in each iteration. On the left, the torque scale is in units of N.ft $\times 10^{-4}$. On the right, the length scale is in units of ft $\times 10^{-4}$.

Table 2
Key GA behavioral parameter values applied to four runs for which the two objective function trends are illustrated in Fig. 11.

	Run 1	Run 2	Run 3	Run 4
P_C	Values adapted as	0.2	0.5	0.8
P_M	iterations progress	0.8	0.5	0.2
R_M		0.5	0.5	0.5
Minimum torque (N.ft) obtained	11,745	11,784	11,847	11,877
Minimum wellbore length (ft) obtained	15,023	15,022	15,035	15,033

space by the algorithm and a more rapid convergence trend. For example, the mutation probability in the early iterations of the GA needs to be high enough to find the zone of absolute optima and then can be gradually decreased. On the other hand, the cross-over probability should be increased to speed up the location of the absolute optima, once the algorithm has located the optimal zone in the feasible region. The parameter-adaption approach of our methodology has three simple, but efficient, layers:

1. Applying variable values for cross-over probability and mutation probability as the iterations progress. At first, P_M is set to a high value, such as 0.8 and P_C is set to a low value, such as 0.2. The high value of P_M typically will locate the optimal zone in the feasible region. P_M then is decreased gradually, whereas P_C is increased gradually to find the absolute optimal points. The values used for P_M and P_C are determined by the following relationships:

$$P_C = 0.2 + 0.7 \times (\text{ITERATION\_NUMBER}/\text{MAX\_ITERATION})$$

$$P_M = 0.8 - 0.7 \times (\text{ITERATION\_NUMBER}/\text{MAX\_ITERATION})$$

2. When no solution improvements are observed after a large number of iterations, a sudden increase of P_M is applied. When the optimization trend does not show any improvement after a large number of iterations, it is possible that the algorithm has become stuck at some local optima. A drastic change in GA behavioral-parameter values, in particular P_M, is invoked to eject the algorithm from such local minima. This is achieved in our methodology by disturbing the layer 1 rule to significantly increasing P_M in such circumstances.
3. The mutation function includes a factor that controls the extent of modification that is applied to individuals subjected to mutation. This factor, termed "mutation rate" or R_M in our methodology is a value from zero to one. The mutation rate value for the early iterations is typically set to be high, but once convergence to the optimal zone in the feasible region is achieved, a small change in an individual is more effective in finding individuals with better fitness scores. To achieve this, R_M value is adjusted in our methodology to become progressively lower as iterations progress. The mutation rate for the analysis presented here is adjusted by the following relationship as the algorithm evolves through its iterations:

$$R_M = 0.6 - 0.5 \times (\text{ITERATION\_NUMBER}/\text{MAX\_ITERATION})$$

Fig. 11 illustrates the GA evolutionary trends for the two objective functions in four runs with different GA behavioral-

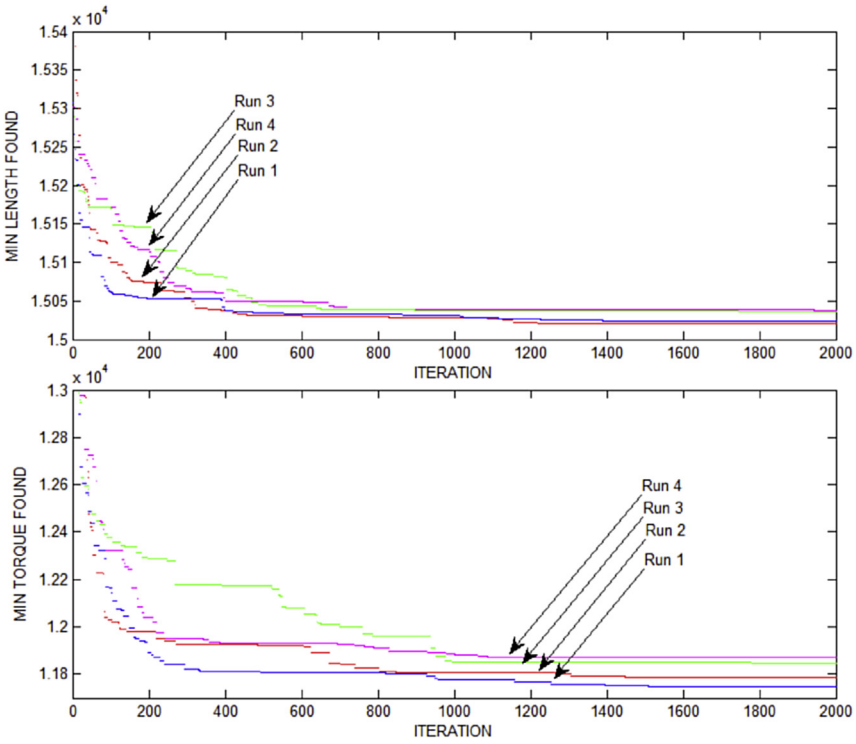

Fig. 11. Objective function trends compared for adaptive GA behavioral parameters versus constant-value GA behavioral parameters. The four runs illustrated were all conducted using the same initial population, i.e. they all begin at the same points on the left sides of the two graphs. Run 1 (adaptive parameters) shows the best convergence because of the high mutation probability applied in the initial iterations, and finds lower value optima for the objective functions because of the high cross-over probability applied in later iterations. Run 4 shows a smoother trend line and worst convergence towards its optima, and finds the least attractive optima of the four runs, because it applies the lowest mutation probability (0.2). The torque scale is in units of N.ft $\times 10^{-4}$ and the length scale is in units of ft $\times 10^{-4}$.

parameter setups. The benefit of using a parameter-adaption approach in searching the feasible space and in enhancing the convergence trend and convergence time is clear from the trends shown in Fig. 11. Table 2 specifies the values of P_C, P_M and R_M for each run. . The best performing (lower) line in Fig. 11 represents Run 1 (Table 2), the case using adaptive GA behavioral parameters. The three other lines, Runs 2 to 4, use constant values for the GA behavioral parameters, across all iterations. More rapid convergence and better results are clearly achieved using adaptive behavioral parameters.

6.2. Multi-objective function optimization results

The complexity of the well path makes it impossible to observe and characterize an explicit relationship between torque on the drill string and well path parameters. However, it seems likely that in feasible solutions with deeper kick-off (i.e. angle build-up) points, a higher dog-leg severity is required to reach to the target zone. This causes more friction torque on the drill string.

The Pareto frontier in typical two-objective optimization problems, such as the one described here, show an ascendant trend in one objective function's value versus a descendant trend in the value of the other objective function. In the case studied, the Pareto frontier established does not mean that the lowest torque is along the longest well path, and the shortest well path results in the highest torque on the drill string. The relationship between those objective functions is non-linear and the GA optimization has identified some high-performing samples that define the Pareto frontier; there are likely to be other solutions, not found by the algorithm, due to the behavioral constraints imposed, that would extend, or provide finer-detail, along that frontier.

The solutions that yield absolute minimum values of each objective function should be subset of real Pareto optimal solution set. As shown in Fig. 12, the dots corresponding to single-objective function GA optimization for the torque and the wellbore length (triangle shaped) do extend the trace of the sample of Pareto optimal solutions (spherical shaped) in both directions. This lends weight to the conclusion that the dots related to single objective function GA optimization shown in Fig. 12 do indeed represent a subset of the real (complete) Pareto optimal solution set.

By definition all the solutions along the Pareto frontier have certain optimal characteristics (i.e., they are Pareto optimal solutions). Once the Pareto frontier is defined the challenge is to

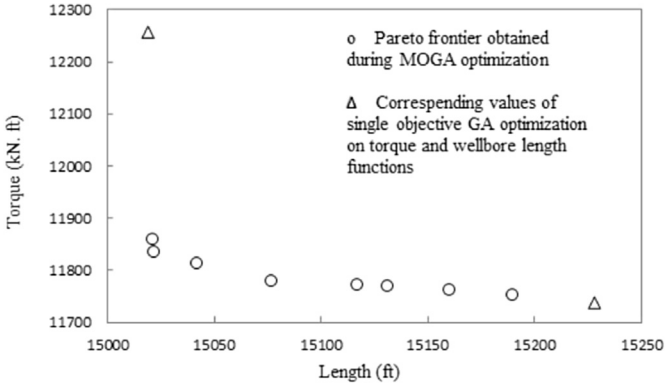

Fig. 12. Pareto frontier obtained for the two objective functions, wellbore length and torque, during MOGA, together with the results of single-objective function GA for the wellbore length and torque functions for the wellbore scenario studied.

Table 3
Results of MOGA and results of previous studies on the same wellbore trajectory scenario.

	GA design, Shokir et al. (2004)	PSO design, A. Atashnezhad et al. (2014)	This study									
			MOGA design (Pareto optimal solutions)								Single objective GA on torque	Single objective GA on length
TMD (ft)	15,496	15,023	15,131	15,190	15,117	15,022	15,042	15,021	15,160	15,077	(15,228)	**15,019**
Torque (N.ft)	–	–	11,769	11,752	11,772	11,834	11,812	11,860	11,761	11,779	**11,738**	(12,257)
TVD (ft)	–	–	10,853	11,850	10,854	10,850	10,855	10,850	10,850	10,856	10,850	10,850
D_{KOP}(ft)	987	1000	1000	1000	1000	1000	1000	1000	1000	1000	994	1000
D_D (ft)	6804	7000	6998	6998	6998	6998	6998	6998	6998	6998	6968	70,000
D_B (ft)	10,004	10,200	10,166	10,166	10,166	10,197	10,197	10,197	10,197	10,166	10,097	10,200
φ_1 (degree)	13	10	10	10	10	10	10	10	10	10	10	10
φ_2 (degree)	42	40	40	40	40	40	40	40	40	40	40	40
φ_3(degree)	90	90	92	94	91	90	90	90	90	90	92	90
θ_1(degree)	279	270	270	270	270	270	270	270	270	270	270	270
θ_2 (degree)	279	280	280	280	280	280	280	280	280	280	280	280
θ_3 (degree)	275	275	280	280	280	280	280	278	278	280	280	276
θ_4 (degree)	332	331	331	331	331	333	331	333	333	331	330	340
θ_5 (degree)	334	340	331	331	331	333	331	333	333	331	330	340
θ_6 (degree)	335	355	357	357	357	357	357	357	357	357	359	356
T_1 (°/100 ft)	1.675	0.829	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.82	0.83
T_3 (°/100 ft)	1.431	1.666	1.65	1.65	1.65	1.66	1.66	1.66	1.66	1.65	1.62	1.68
T_5 (°/100 ft)	2.413	3.243	3.23	3.23	3.23	3.38	3.38	3.38	3.38	3.23	3.00	3.31

identify which one of those Pareto optimal solutions should be selected for the well design. Analyzing the cost and time to achieve each Pareto optimum solution is one way to discriminate, but that is beyond the scope of this study.

Table 3 shows selected Pareto optimal solutions from MOGA together with the results of single-objective function GA for the wellbore length and torque functions for the wellbore scenario studied. Whereas it seems likely that in the single-objective function GA optimum solutions the lowest torque is along the well path that has the lowest dog leg severities, and the lowest length of the wellbore has the highest dog leg severities, it is not so clear cut for solutions along the Pareto frontier. There may be other geometric factors, in addition to dog-leg severity, that are influencing the wellbore length versus torque relationship.

7. Conclusion

This study has developed, applied and analyzed a two-objective function optimization model for a complex directional wellbore trajectory striving to minimize two conflicting objective functions of measured wellbore length and drilling torque, subject to multiple constraints. A customized multi-objective genetic algorithm (MOGA), with the logic clearly described and the MATLAB code transparently presented in Appendices B and C provides an impressive range of Pareto optimal solutions for this wellbore scenario. Rapid convergence and the high-performing results compared to previously-published, single-objective function studies of the same complex, deviated wellbore scenario highlight the efficiency of the MOGA proposed. The algorithm ensures that all constraints are satisfied in the selected optimal solutions. The following specific conclusions can been drawn related to the wellbore trajectory scenario studied:

- There is no explicit relationship between the wellbore length and the torque on the drill string. This means that solutions providing the minimum measured depth for the well do not necessarily provide the lowest torque on the drill string, and vice versa.
- The methodology used in this study to calculate torque on the drill string is one of several that could be used. As torque is a complex function dependent on many variables, it is clear that other methodologies for calculating torque would likely lead to different results. Such detailed analysis of torque on the drill string goes beyond the scope of this work and demands its own comprehensive study. This study is primarily concerned with demonstrating the merits of two-objective function genetic optimization.
- Considering the parameters influencing well-trajectory design, the results of the optimization scenario presented are consistent with the conclusion that torque is most significantly affected by the dog-leg severity of any specific trajectory; i.e., higher dog leg severity causes higher torque on the drill string.
- When applying well path optimization to specific target locations it is often important to consider other factors in addition to wellbore length and torque. For instance, wellbore tortuosity will influence the ability to run specific completion designs and the cost of the well completion. Minimizing combined drilling and completion costs is another common optimization target for specific operations. In some cases these additional issues may be considered as optimization constraints (e.g. setting casing above or below certain problem geological horizons). Our future research will therefore be focused on developing our proposed MOGA to incorporate additional objectives such as well tortuosity and drilling and completion costs.

7.1. Concerning the GA and MOGA

- GA and, therefore, MOGA are powerful algorithms for searching the feasible solution space defined by objective functions and for defining Pareto optimal solutions for them when more than one objective function is involved. This approach is well suited to drilling optimization challenges.

Finally for future field development for more difficult wells, the provided algorithm can be used as well as for the wellbore scenario evaluated simply by changing the objective function and constraints function only.

Nomenclature

Definitions of wellbore trajectory variables (modified after Shokir et al., 2004)

φ_1,φ_2,φ_3	First, second and third hold angles, degrees
θ_1	Azimuth angle at kick off point, degrees
θ_2	Azimuth angle at end of first build, degrees
θ_3	Azimuth angle at end of first hold section, degrees
θ_4	Azimuth angle at end of second build or drop, degrees
θ_5	Azimuth angle at end of second hold section, degrees
θ_6	Azimuth angle at end of third build portion, degrees
T_1	Dogleg severities of first build portion, °/100 feet
T_2	Dogleg severity of first hold portion, °/100 feet
T_3	Dogleg severity of second build or drop portion, °/100 feet
T_4	Dogleg severity of second hold or drop portion, °/100 feet
T_5	Dogleg severity of third build or drop portion, °/100 feet
TMD	True measured depth
TVD	True vertical depth
D_{KOP}	Depth of kick of point
D_B	True vertical depth of the well at the end of drop-off section (top of third build section), feet
D_D	True vertical depth of the well at the top of drop-off section (top of second build section), feet
HD	lateral length (horizontal length), feet
P_1, P_2	Two survey points
e_1, e_2	Unit vectors in the direction of the wellbore
F_1	Axial load at the bottom of element
F_2	Axial load at the top of element
β	Total angle change
B	Buoyancy factor
μ	Friction factor
w	Weight of unit length, kN/feet
r	Radius of the pipes, feet
ΔL	Interval length
P_C	Cross over probability
P_M	Mutation probability
R_M	Mutation rate

Appendix A

Detailed calculation of the well length and the torque on the drill string for case study of this paper.

Given the well is consist of 7 section (Vertical section, First build section, First hold, Second build, Second hold, Third build, Horizontal section), for each section, the length and the torque is separately calculated. First for the length:

$$\Delta L_1 = D_{KOP}$$

$$\Delta L_2 = D_1 = R_1\sqrt{(\theta_2-\theta_1)^2\sin^4\left(\frac{\varphi_1+\varphi_0}{2}\right)+(\varphi_1+\varphi_0)}, (\varphi_0=0)$$

$$\Delta L_4 = D_3 = R_3\sqrt{(\theta_4-\theta_3)^2\sin^4\left(\frac{\varphi_2+\varphi_1}{2}\right)+(\varphi_2+\varphi_1)}$$

$$\Delta L_6 = D_5 = R_5\sqrt{(\theta_5-\theta_6)^2\sin^4\left(\frac{\varphi_3+\varphi_2}{2}\right)+(\varphi_3+\varphi_2)}$$

$$\Delta L_3 = D_2 = D_D - D_{KOP} - D_1 \times (\sin\varphi_1 - \sin\varphi_0)/((\varphi_1-\varphi_0)\times\cos\varphi_1)$$

$$\Delta L_5 = D_4 = D_B - D_D - D_3 \times (\sin\varphi_2 - \sin\varphi_1)/((\varphi_2-\varphi_1)\times\cos\varphi_2)$$

$$\Delta L_7 = HD$$

The torque calculation starts at the bottom of the drill string and proceeds stepwise upward. First the axial loading at the bottom of each section is calculated. It is assumed that there is no axial movement (just rotating) and no weight on bit ($F_7 = 0$).

$$F_7 = 0$$

$$F_6 = F_7 + Bw\Delta L_7\cos\varphi_3 = Bw\Delta L_7\cos\alpha_3$$

$$F_5 = F_6 + Bw\Delta L_6\left(\frac{\sin\varphi_3-\sin\varphi_2}{\varphi_3-\varphi_2}\right)$$

$$F_4 = F_5 + Bw\Delta L_5\cos\varphi_2$$

$$F_3 = F_4 + Bw\Delta L_4\left(\frac{\sin\varphi_2-\sin\varphi_1}{\varphi_2-\varphi_1}\right)$$

$$F_2 = F_3 + Bw\Delta L_3\cos\varphi_1$$

$$F_1 = F_2 + Bw\Delta L_2\left(\frac{\sin\varphi_0-\sin\varphi_1}{\varphi_0-\varphi_1}\right), (\varphi_0=0)$$

Now, the torque for each section is calculated. For build and drop sections, the total angle change (β) is required.

$$T_7 = \mu rw\Delta L_7\sin\varphi_3$$

$$\cos\beta_6 = \sin\varphi_3\sin\varphi_2\cos(\theta_5-\theta_6)+\cos\theta_5\cos\theta_6$$

$$T_6 = \mu rF_6\beta_6$$

$$T_5 = \mu rw\Delta L_5\sin\varphi_2$$

$$\cos\beta_4 = \sin\varphi_2\sin\varphi_1\cos(\theta_3-\theta_4)+\cos\theta_3\cos\theta_4$$

$$T_4 = \mu rF_4\beta_4$$

$$T_3 = \mu rw\Delta L_3\sin\varphi_1$$

$$\cos\beta_2 = \sin\varphi_1\sin\varphi_0\cos(\theta_1-\theta_2)+\cos\theta_1\cos\theta_2 = \cos\varphi_1\cos\varphi_2, (\varphi_0=0)$$

$$T_2 = \mu rF_2\beta_2$$

$$T_1 = \mu rw\Delta L_1\sin\varphi_0, (\varphi_0=0)$$

In the calculations, the buoyancy factor (B), weight of unit length of the pipes (w), friction factor (μ) and radius of the pipes (r) are assumed to be 0.7, 0.3 kN/ft, 0.2, 0.1 ft, respectively.

Appendix B

MATLAB code listing for MOGA to solve deviated wellbore trajectories using the radius of curvature method.

```
function [  ] = moga(  )
    %MULTIOBJECTIVE OPTIMIZATION
    %   Detailed procedure
    %   Place the final results in the objective function(objct.m)
    close;
    clear;clc;
    GENLENGTH=17;
    POP=ini();
    F=zeros(100,3);
%%
    for k=1:100
        for i=1:100
            F(i,:)= objct( POP(i,:) );
            RIS(i)=const(POP(i,:));
        end
        plotf( POP , F , k );
        RANK = evalf( F,RIS );
        PARENTS = select( RANK );
        POP = xoverh(PARENTS,GENLENGTH,POP,RANK );
        for i=1:100
            F(i,:)= objct( POP(i,:) );
        end
        POP = Mut(PARENTS, POP );
    end
    %%
    for i=1:100
        F(i,:)= objct( POP(i,:) );
    end
    RANK = evalf( F ,RIS);
    for h=1:size(F,1)
        if const(POP(h,:))==0 && RANK(h)==1
            fprintf('length %i torque %i TVD %i \n',F(h,:))
        end
    end
    for h=1:size(F,1)
        if const(POP(h,:))==0 && RANK(h)==1
            fprintf('%i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i\n\n',POP(h,:))
        end
    end
    fprintf('THE END')
end
```

```
function POP = ini()
%INITIALIZING
%   Generates a random Population
clear,clc
POP=zeros(100,17);
for i=2:100
    RISTRICT=1;
    while RISTRICT==1
        POP(i,1)=round(rand*400)+600;
        POP(i,2)=round(rand*1000)+6000;
        POP(i,3)=round(rand*200)+10000;
        POP(i,4)=round(rand*10)+10;
        POP(i,5)=round(rand*30)+40;
        POP(i,6)=round(rand*5)+90;
        POP(i,7)=round(rand*10)+270;
        POP(i,8)=round(rand*10)+270;
        POP(i,9)=round(rand*10)+270;
        POP(i,10)=round(rand*10)+330;
        POP(i,11)=round(rand*10)+330;
        POP(i,12)=round(rand*5)+355;
        POP(i,13)=round(rand*5);
        POP(i,14)=0;
        POP(i,15)=round(rand*5);
        POP(i,16)=0;
        POP(i,17)=round(rand*5);
        RISTRICT=const(POP(i,:));
    end
end
end
```

```
function OBJ = objct( X )
% X(1:3)  ...1,2,3 th BU depth
% X(4:6)  ...1,2,3 th HOLD angle
% X(7:12) ...AZ angles
% X(13:17)...DLSs
B=0.7;W=0.3;r=0.1;U=.2;
AZ=X(7:12).*pi/180;
INC=X(4:6).*pi/180;
DLS=X(13:17).*pi/180;
R=100./(DLS);
BU=X(1:3);

%%
D(1)= R(1)*((AZ(2)-AZ(1))^2*(sin(INC(1)/2))^4+(INC(1))^2)^0.5;
D(3)= R(3)*((AZ(4)-AZ(3))^2*(sin((INC(1)+INC(2))/2))^4+(INC(1)-INC(2))^2)^0.5;
D(5)= R(5)*((AZ(6)-AZ(5))^2*(sin((INC(3)+INC(2))/2))^4+(INC(3)-INC(2))^2)^0.5;
D(2)= (BU(2)-BU(1)-D(1)*sin(INC(1))/INC(1))/cos(INC(1));
D(4)= (BU(3)-BU(2)-D(3)*(sin(INC(2))-sin(INC(1)))/(INC(2)-INC(1)))/cos(INC(2));
%%
V(1)=D(1)*(sin(INC(1)))/(INC(1));
V(3)=D(3)*(sin(INC(2))-sin(INC(1)))/(INC(2)-INC(1));
V(5)=D(5)*(sin(INC(3))-sin(INC(2)))/(INC(3)-INC(2));
```

```
V(2)=D(2)*(cos(INC(1)));
V(4)=D(4)*(cos(INC(2)));
%%
TETA(1)= acos(cos(INC(1))*1);
TETA(2)= acos(sin(INC(1))*sin(INC(1))*cos(AZ(2)-AZ(3))+cos(INC(1))*cos(INC(1)));
TETA(3)= acos(sin(INC(2))*sin(INC(1))*cos(AZ(3)-AZ(4))+cos(INC(1))*cos(INC(2)));
TETA(4)= acos(sin(INC(2))*sin(INC(2))*cos(AZ(4)-AZ(5))+cos(INC(2))*cos(INC(2)));
TETA(5)= acos(sin(INC(3))*sin(INC(2))*cos(AZ(5)-AZ(6))+cos(INC(3))*cos(INC(2)));

%%
F(5)=0;
F(4)=F(5)+B*W*D(5)*abs((sin(INC(3))-sin(INC(2)))/(INC(3)-INC(2)));
F(3)=F(4)+B*W*D(4)*cos(INC(2));
F(2)=F(3)+B*W*D(3)*abs((sin(INC(1))-sin(INC(2)))/(INC(1)-INC(2)));
F(1)=F(2)+B*W*D(2)*cos(INC(1));
%%
T=U*r.*F.*abs(TETA);
%%
OBJ(1)=sum(D)+BU(1)+2500;
OBJ(2)=sum(T)*1000;
OBJ(3)=sum(V)+BU(1); % third objective function is TVD as a constraint
end
```

```
function PARENTS = select( RANK )
%SELECTION
%   select the parents
PARENTS=zeros(1,80);
wheel=cumsum(RANK)/sum(RANK);
m=1;
while  m<81
    r=rand;
    n=0;
    for j=1:length(wheel)
        if r<wheel(j)
            for k=1:m
                if j==PARENTS(k)
                  n=n+1;
                end
            end
            if n==0
                PARENTS(m)=j;
                m=m+1;
                break
            end
        end
    end
end
end
```

```
function pop = Mut( pop,rank)
%MUTATION
x=pop;
range=[600 6000 10000 10 40 90 270 270 270 330 330 355 0 0 0 0 0
      1000 7000 10200 20 70 95 280 280 280 340 340 360 5 5 5 5 5];
for i=1:size(x,1)
    if rank(i)==1
        continue
    end

    while RISTRICT==1
        child=x(i,:);
        mutpoints=find(round(rand(1,length(child)))<.01);
        lover=range(1,:);
        uper=range(2,:);
        span=uper-lover;
        child(mutpoints) = lover(mutpoints) + round(rand(1,length(mutpoints)) .*
span(mutpoints));
        x(i,:)=child;
        RISTRICT=const(x(i,:));
    end
end
 pop=x;
```

```
end
```

Appendix C

MATLAB code listing for single objective GA to solve deviated wellbore trajectories using the radius of curvature method.

```
function [  ] = ga(  )
    %SINGLEOBJECTIVE OPTIMIZATION
    %   Detailed procedure
    %   Place the final results in the objective function(objct.m)
    close;
    clear;clc;
    GENLENGTH=17;
    POP=ini();
    F=zeros(100,2);
    nF = input('Which do U want to optimize? Length(1) or Torque(2)? Enter 1 or 2: \n');
    if nF~=1
        if nF~=2
            fprintf('WRONG VALUE ENTERED\n')
            nF = input('Which do U want to optimize? Length(1) or Torque(2)? Enter 1 or
2: \n');
            ga;
        end
    end
    %%
    for k=1:2000
        for i=1:100
            F(i,:)= singleobjct( POP(i,:),nF );
            RISTRICT(i)=const(POP(i,:));
        end
        for i=1:100
            if F(i,1)==min(F(RISTRICT==0,1))
                MIN=POP(i,:);
            end
        end
        plotf( POP , F , k );
        RANK = evalf( F,RISTRICT );
        PARENTS = select( RANK );
        POP = xover(PARENTS,GENLENGTH,POP,RANK,k,nF );
        %POP = Mut( PARENTS,POP,k );
        a=randi(size(POP,1));
        POP(a,:)=MIN;
    end
    %%
    for i=1:100
        F3(i,:)= objct( POP(i,:) );
        F(i,:)= singleobjct( POP(i,:) ,nF);
    end
    RANK = evalf( F ,RISTRICT);
    for h=1:100
        if const(POP(h,:))==0 && RANK(h)==1
            fprintf('length %d torque %d TVD %d \n',F3(i,1),F3(i,2),F3(i,3))
        end
    end
    for h=1:100
        if const(POP(h,:))==0 && RANK(h)==1
            fprintf('%i %i %i %i %i %i %i %i %i %i %i %i %d %d %d %d
%d\n\n',round(POP(h,1:12)),round(100.*POP(h,13:17)))
        end
    end
    fprintf('THE END')
end
```

```
function RANK = evalf( F ,RIS)
%%
n=1;
RANK(1:size(F,1))=0;
while sum(1./F)~=0
    for i=1:size(F,1)
        m=0;
        if RIS(i)==1
            F(i,:)=inf;
            RANK(i)=0;
            continue
        end
        for j=1:size(F,1)
            if j==i
                continue
            end
            if (F(j,1)<F(i,1))
                m=m+1;
                break
            end
        end
        if m==0
            RANK(i)=1/n;
            n=n+1;
        end
    end
    F((RANK~=0),:)=inf;

end


end
```

```
function pop = reproduct( parents,genlength,pop,rank,iter,nF )
%XOVER
%   2point xover
Pc=0.2+ 0.7*iter/2000;
if abs(iter-1000)<500
    Pc=0.1;
end

xnkids=round(Pc*length(parents));
if rem(xnkids,2)==1
    xnkids=xnkids+1;
end
xoverkids=zeros(xnkids,genlength);
index=1;
xparents_num=randi(length(parents),[1,xnkids]);
xparents=parents(xparents_num);

    for i=1:xnkids/2
            %get parents
            xparent1 = pop(xparents(index),:);
            index=index+1;
            xparent2=pop(xparents(index),:);
            index=index+1;

            sz = length(xparent1) - 1;
            xpoint1 = ceil(sz * rand);
            xpoint2 = ceil(sz * rand);
            while(xpoint2 == xpoint1)
                xpoint2 = ceil(sz * rand);
            end
            if xpoint1<xpoint2
                left=xpoint1;
                right=xpoint2;
            else
                left=xpoint2;
                right=xpoint1;
            end
            %xover
            a=[xparent1(1:left),xparent2(left+1:right),xparent1(right+1:end)];
            b=[xparent2(1:left),xparent1(left+1:right),xparent2(right+1:end)];
                xoverkids(2*i-1,:)=a;
                xoverkids(2*i,:)=b;
                parents(xparents_num(2*i-1))=0;
                parents(xparents_num(2*i))=0;

    end
%Replacement
k=1;
```

```
    for i=1: size(pop,1)
        m=0;
        for j=1:size(pop,1)
            if j== i
                continue
            end
            if rank(i)>rank(j)
                m=m+1;
                break
            end

        end

        if m==0
            pop(i,:)=xoverkids(k,:);
            rank(i)=inf;
            k=k+1;
            if k==xnkids+1
                break
            end

        end
    end
%mutation
mparents=parents(parents~=0);
f=zeros(100,2);
for i=1:100
    f(i,:)= singleobjct( pop(i,:),nF );
end
P=0.5;%-0.2*iter/2000;
if iter>500
    P=0.2;
end
if iter>1000
    P=0.1;
end
range=[600 6000 10000 10 40 90 270 270 270 330 330 355 0 0 0 0 0
      1000 7000 10200 20 70 95 280 280 280 340 340 360 5 5 5 5 5];
for i=1:length(mparents)
    RISTRICT=1;
    while RISTRICT==1
        child=pop(mparents(i),:);
        mutpoints=find((rand(1,length(child)))<P);
        lover=range(1,:);
        uper=range(2,:);
        span=uper-lover;
        child(mutpoints) = lover(mutpoints) + (rand(1,length(mutpoints)) .*
span(mutpoints));
        RISTRICT=const(child);
    end
    newf=singleobjct(child,nF);
    if newf(1)< f(mparents(i),1)
        pop(mparents(i),:)=child;
    end
end

end
```

```
function F=singleobjct(X,nF)
OBJ=objct(X);
F(1)=OBJ(nF);
F(2)=OBJ(3);
end
```

```
function PARENTS = select( RANK )
%SELECTION
%
PARENTS=zeros(1,80);
wheel=cumsum(RANK)/sum(RANK);
m=1;
while  m<81
```

```
    r=rand;
    n=0;
    for j=1:length(wheel)
        if r<wheel(j)
            for k=1:m
                if j==PARENTS(k)
                  n=n+1;
                end
            end
            if n==0
                PARENTS(m)=j;
                m=m+1;
                break
            end
        end
    end
end
end
```

```
function POP = ini()
%INITIALIZING
%   Generates a random Population
clear,clc
POP=zeros(100,17);
for i=1:100
    RISTRICT=1;
    while RISTRICT==1
        POP(i,1)=round(rand*400)+600;
        POP(i,2)=round(rand*1000)+6000;
        POP(i,3)=round(rand*200)+10000;
        POP(i,4)=round(rand*10)+10;
        POP(i,5)=round(rand*30)+40;
        POP(i,6)=round(rand*5)+90;
        POP(i,7)=round(rand*10)+270;
        POP(i,8)=round(rand*10)+270;
        POP(i,9)=round(rand*10)+270;
        POP(i,10)=round(rand*10)+330;
        POP(i,11)=round(rand*10)+330;
        POP(i,12)=round(rand*5)+355;
        POP(i,13)=round(rand*5);
        POP(i,14)=0;
        POP(i,15)=round(rand*5);
        POP(i,16)=0;
        POP(i,17)=round(rand*5);
        RISTRICT=const(POP(i,:));
    end
end
end
```

References

Aadnøy, B.S., Andersen, K., 2001. Design of oil wells using analytical friction models. J. Pet. Sci. Eng. 32, 53–71.

Aadnøy, B.S., Fazaelizadeh, M., Hareland, G., 2010. A 3-D analytical model for wellbore friction (SPE paper 141515). J. Can. Pet. Technol. 49 (10), 25–36.

Adams, J.N., Charrier, T., 1985. Drilling Engineering: a Complete Well Planning Approach. PennWell Publishing Company, Tulsa, Oklahoma, pp. 342–345.

Atashnezhad, A., Wood, D.A., Fereidounpour, A., Khosravanian, R., 2014. Designing and optimizing deviated wellbore trajectories using novel particle swarm algorithms. J. Nat. Gas Sci. Eng. 21, 1184–1204.

Fazaelizadeh, M., 2013. Real Time Torque and Drag Analysis during Directional Drilling (PhD thesis). University of Calgary, Alberta, Canada.

Guria, C., Goli, K., Pathak, A., 2014. Multi-objective optimization of oil well drilling using elitist non-dominated sorting genetic algorithm. J. Pet. Sci. Eng. 11, 97–110.

Gen, M., Cheng, R., 2008. Network Models and Optimization. Springer, p. 692.

Haupt, R.L., Haupt, S.E., 2004. Practical Genetic Algorithms, second ed. Wiley Interscience.

Johancsik, C.A., Friesen, D.B., Dawson, R., 1984. Torque and drag in directional wells–prediction and measurement. J. Pet. Technol. 987–992. June 1984.

Joshi, S.D., 2003. Cost/benefits of horizontal wells, 2003 (SPE paper 83621). In: SPE/AAPG Joint Meeting at Long Beach, California, U.S.A., 19–24 May 2003.

Khaled, A/El, Fattah, S., El-Tayeb, A., Dahab, A., Khalaf, F., 1999. 3-D well design using computer optimization model. In: 4th Computer Conference, SPE Egypt Section, Cairo, Egypt.

Li, A., 2010. The operator of genetic algorithms to improve its properties. J. Mod. Appl. Sci. 4 (3), 60–62.

Lin, W.Y., Lee, W.Y., Hong, T.P., 2003. Adapting crossover and mutation rates in genetic algorithms. J. Inf. Sci. Eng. 19, 889–903.

McCormick, J.E., Chiu, T.F., 2011. The practice and evolution of torque and drag reduction: theory and field results (SPE conference paper 147100). In: SPE Annual Technical Conference and Exhibition, 30 October-2 November, Denver, Colorado, USA.

Shokir, E.M., Emera, M.K., Eid, S.M., Wally, A.W., 2004. A new optimization model for 3-D well design. Emir. J. Eng. 9 (1), 67–74.

Sheppard, M.C., Wick, C., Burgess, T., 1987. Designing well paths to reduce torque and drag (SPE Paper 15463). SPE Drill. Eng. 2 (4), 344–350.

Sivanandam, S.N., Deepa, S.N., 2008. Introduction to Genetic Algorithms. Springer, p. 442.

Xie, L., Moran, D., Yan, L., et al., 2012. Sophisticated software analysis system and use of torque/drag modeling for complex well operations increase operational efficiency (SPE Paper 152056). In: Presented at the Western Regional Meeting, Bakersfield, USA, 21–23 March.

Yasari, E., Pishvaie, M.R., Khorasheh, F., Salahshoor, K., Kharrat, R., 2013. Application of multi-criterion robust optimization in water-flooding of oil reservoir. J. Pet. Sci. Eng. 74, 147–153.