Journal of Petroleum Exploration and Production Technology (2018) 8:1487–1503
https://doi.org/10.1007/s13202-018-0447-2

ORIGINAL PAPER - EXPLORATION ENGINEERING

# A comparative study of several metaheuristic algorithms for optimizing complex 3-D well-path designs

**Rassoul Khosravanian[1] · Vahid Mansouri[1] · David A. Wood[2] · Masood Reza Alipour[1]**

Received: 3 September 2017 / Accepted: 9 February 2018 / Published online: 10 March 2018

## Abstract

Considering the importance of cost reduction in the petroleum industry, especially in drilling operations, this study focused on the minimization of the well-path length, for complex well designs, compares the performance of several metaheuristic evolutionary algorithms. Genetic, ant colony, artificial bee colony and harmony search algorithms are evaluated to seek the best performance among them with respect to minimizing well-path length and also minimizing computation time taken to converge toward global optima for two horizontal wellbore cases: (1) a real well offshore Iran; (2) a well-studied complex trajectory with several build and hold sections. A primary aim of the study is to derive less time-consuming algorithms that can be deployed to solve a range of complex well-path design challenges. This has been achieved by identifying flexible control parameters that can be successfully adjusted to tune each algorithm, leading to the most efficient performance (i.e., rapidly locating global optima while consuming minimum computational time), when applied to each well-path case evaluated. The comparative analysis of the results obtained for the two case studies suggests that genetic, artificial bee colony and harmony search algorithms can each be successively tuned with control parameters to achieve those objectives, whereas the ant colony algorithm cannot.

**Keywords** Metaheuristic algorithms · Well-path designing · Well-path optimization · Genetic algorithm · Harmony search · Artificial bee colony

## List of symbols

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

| | |
|---|---|
| $\varphi_1, \varphi_2, \varphi_3$ | First, second and third hold angles, ° |
| $\theta_1$ | Azimuth angle at kickoff point, ° |
| $\theta_2$ | Azimuth angle at end of first build, ° |
| $\theta_3$ | Azimuth angle at end of first hold section, ° |
| $\theta_4$ | Azimuth angle at end of second build or drop, ° |
| $\theta_5$ | Azimuth angle at end of second hold section, ° |
| $\theta_6$ | Azimuth angle at end of third build portion, ° |
| $T_1$ | Dogleg severities of first build portion, °/100 ft |
| $T_2$ | Dogleg severity of first hold portion, °/100 ft |
| $T_3$ | Dogleg severity of second build or drop portion, °/100 ft |
| $T_4$ | Dogleg severity of second hold or drop portion, °/100 ft |
| $T_5$ | Dogleg severity of third build or drop portion, °/100 ft |
| TMD | True measured depth |
| TVD | True vertical depth |

✉ Rassoul Khosravanian
khosravanian@aut.ac.ir

Vahid Mansouri
vahid.mansouri1990@gmail.com

David A. Wood
dw@dwasolutions.com

Masood Reza Alipour
mass0ood@gmail.com

[1] Department of Petroleum Engineering, University of Amirkabir, Tehran, Iran

[2] DWA Energy Limited, Lincoln, UK

| | |
|---|---|
| $D_{\mathrm{KOP}}$ | Depth of kickoff point |
| $D_{\mathrm{B}}$ | True vertical depth of the well at the end of drop-off section (top of third build section), ft |
| $D_{\mathrm{D}}$ | True vertical depth of the well at the top of drop-off section (top of second build section), ft |
| HD | Lateral length (horizontal length), ft |

## Introduction

Optimizing the wellbore length of a multi-directional well requires algorithms that not only find the minimum length, but do so quickly and effectively, taking into account operational limitations in design. Hence, the execution time of algorithms should be taken into account in addition to their ability to find the optimum well trajectory in a reproducible manner. This study compares several established metaheuristic optimization algorithms in terms of both their ability to achieve an acceptable objective function value (i.e., wellbore length subject to operational constraints applied) and the algorithms' computer execution times.

Heuristic algorithms seek "acceptable" solutions to optimization challenges by "trail-and-error" taking a "reasonable" amount of computational timing. What constitutes "acceptable" and "reasonable" is clearly subjective and to an extent depends upon the nature of the optimization task addressed and the context and urgency in which an optimal solution is being sought. Heuristic methods generally do not guarantee finding the best or global optimum, i.e., their solutions often could be improved upon if more computational time were dedicated to the task. However, in many applied operational applications a "good/acceptable" solution may be a rapidly determined local optima situated close to the global optimum. Hence, heuristic algorithms that search local regions of a feasible solution space detecting local optima form important components of metaheuristic methods.

The term metaheuristic refers to higher-level heuristic algorithms (e.g., Bianchi et al. 2009; Yang 2009) that typically combine several lower-level heuristic processes in achieving their higher-level strategic optimization objectives. High-performing metaheuristic optimization algorithms efficiently search a feasible solution space which is too large to be completely sampled in a reasonable time. As Yang (2009) explains, "intensification" and "diversification" are two key attributes of modern metaheuristics: "*For an algorithm to be efficient and effective, it must be able to generate a diverse range of solutions including the potentially optimal solutions so as to explore the whole search space effectively, while it intensifies its search around the neighborhood of an optimal or nearly optimal solution.*" The metaheuristic optimization algorithms evaluated, in terms of wellbore designs of a complex trajectory subject to constraints, are: genetic algorithm (GA), artificial bee colony (ABC), ant colony optimization (ACO) and harmony search (HS). The results are compared to previously published work on the particle swarm optimization algorithm applied to one of the case studies evaluated (Atashnezhad et al. 2014).

*Genetic algorithms (GA)* involve an evolutionary process, typically starting with a random set of feasible solutions, followed by steps of evolution, i.e., successive iterations that aim to improve their performance in terms of the objective function by modifying a number of some genetic operators, using mechanisms akin to those operating in biological evolutionary processes (Sivanandam and Deepa 2008). GAs have been successfully applied to many nonlinear and non-smooth types of optimization challenges across many industries (Gallagher and Sambridge 1994).

*Artificial bee colony algorithm (ABC)* was created based on swarm intelligence and specifically the food-foraging strategies of bee colonies (Karaboga and Basturk 2007). It has been demonstrated that algorithms based upon bee colony behaviors can solve NP-hard optimization problems (Karaboga and Basturk 2007). Using ABC algorithms, many highly constrained and complex models, as well as models that cannot be solved with deterministic functions (i.e., are probabilistic in nature), can be solved (Karaboga and Basturk 2007). ABC algorithms typically involve three distinct types of bees: employed bees, onlooker bees and scout bees. The location of each potential food source (or meal) is considered as a possible solution, with the objective function looking to minimize the distance/time taken to access the identified target location. In the first step of ABC algorithms, a number of random routes are selected between the hive and the specified food source (i.e., target for optimization). The shorter routes provide the employed bees with more foraging time at the food source enabling them to return to the hive with more nectar than those employed bees that have spent longer traveling. On their return to the hive, the employed bees exchange information with the onlooker bees that observe the amount of nectar each bee has collected (in nature this is achieved by a dancing ritual in the hive). The routes between hive and the target location taken by the bees returning with the most nectar are selected with higher preferences for the routes selected by the employed bees for subsequent iterations. In addition, scout bees select a new set of random routes to the specified target to avoid the algorithm becoming stuck at local optima solutions. The onlooker bees accumulate information imparted by employed and scout bees in successive iterations of the ABC algorithm

(Karaboga and Basturk 2007). According to the principles of swarm intelligence, eventually the optimal route between the hive and the specified target is found. ABC algorithms have been successfully applied to solve a number of nonlinear, non-smooth optimization challenges (Karaboga and Ozturk 2009).

*Ant colonies optimization (ACO)* is also based on the behavior of social communities of insects, which operate as distributed systems that despite the functional simplicity of the individual members of the colony, a complex social organization is created by the accumulated knowledge of multiple individuals which leads to modified behavior trending toward optimal patterns (Dorigo and Stutzle 2004; Darquennes 2005). Different components of ACO algorithms are inspired by different aspects of ant colony behavior in finding optimum routes between the colony's nest and an identified food source. Initially the ants follow random trails, but leave pheromone (or scent) along each path. The highest pheromone deposit is located on the shortest route and that encourages ants to follow that route on future journeys between the nest and the food source, reinforcing the pheromone. Overtime the pheromone deposit evaporates at a specified rate resulting in the infrequently followed routes being collectively "forgotten." As with ABC algorithms, scout ants can be employed to select some random routes in each iteration to avoid the colony becoming stuck in local optima and missing a better overall of global optimum solution (i.e., the shortest distance between nest and target location). Traditional ACO algorithms focus upon discrete optimization problems, but require modification and hybridization with other metaheuristics to efficiently solve continuous optimization problems (Hu et al. 2008).

*Harmony search (HS)* algorithms apply the principles employed by musicians and composers when playing existing musical scores and striving to achieve the best combination of musical notes to produce a harmonious outcome (Lee and Geem 2005). Musicians typically take combinations of three distinct approaches when attempting to improve on musical scores through improvisation, these are: (1) playing parts of the original score as initially written, (2) playing the sections of the piece in a close but slightly different combination of notes to the original score and (3) creating sections of the piece through random substitution of notes. Improved scores resulting from the combination of these processes are stored in a matrix known as harmony memory (HM) which is used to converge to the optimum solution (Yang 2009).

The metaheuristic evolutionary algorithms are widely applied to many complex and NP-hard problems that cannot be readily solved analytical models. For example, in the drilling industry GA are applied in many areas, e.g., drilling optimization, well placement, well design, anti-collision problems and ROP modeling. GA are also applicable in multi-objective problems involving several conflicting objectives, such as simultaneously optimization of drilling parameters (Guria et al. 2014) that focused on drilling depth, drilling time and drilling cost. The conflicting objectives together with nonlinear constraints and large numbers of variables make such problems difficult to solve with conventional methods. Some of the evolutionary algorithms are very simple to implement. For example, HS can be readily adapted to solve complex problems such as fluid injection (Khalili et al. 2013) or well placement (Afshari et al. 2013). Also, ABC and PSO are demonstrated to be efficient at solving continuous problems, such as well placement area (Nozohour and Fazelabdolabadi 2016). Despite their simplicity, these algorithms are robust and very fast in terms of computational running time. Among the algorithms evaluated, ACO is more suited to discrete and network optimization problems. For example, the application of ACO is gas allocation in gas lift operations (Zerafat et al. 2009).

All the above-mentioned evolutionary optimization algorithms have been coded in MATLAB to solve 3-D well-path design optimization problems. Each optimization algorithm includes some key parameters that require tuning (i.e., selecting optimum values that lead to better or faster performance). This tuning process is described for each algorithm applied and results are presented for various values of the key parameters for each algorithm, from which the best, or optimally tuned values, are selected. Results obtained by each optimization algorithm applied to the same complex well-path design are presented and compared. A discussion of the pros and cons associated with each algorithm applied to the well-path design optimization is also provided. MATLAB codes were all run on a PC computer with the following specifications: Intel Core i5 2430 M 2.4 GHz, 4 GB DDR3 Memory.

## Well-path design problem used to test optimization algorithms

The gas or oil well-path design to be optimized by the optimization algorithms studied involves determining the combined wellbore length of a complex well involving multiple straight and curved sections of various inclinations and orientations. The objective function is to minimize the combined well bore length subject to a number of specified constraints. As measure depth drilled typically is directly proportional to the drilling cost, it follows that the shortest overall wellbore design is likely to be the cheapest, although other factors such as torque and casing design also play important roles requiring multiple objectives to be optimized (e.g., Mansouri et al. 2015). The particular well-path targets and constraints applied are those used by Shokir et al. (2004) and further utilized by Atashnezhad et al. (2014) to illustrate the performance of tuned-PSO algorithms and Mansouri et al. (2015) to illustrate the multi-objective optimization performance of GA. The lengths of the curved sections of the wellbore in the example used are calculated by the radius-of-curvature method based on the curves being achieved at constant rates of curvature (Fig. 1). The curvatures of the curved wellbore sections are achieved using the following formulas (Shokir et al. 2004):

$$a = \frac{1}{\Delta m}\sqrt{(\theta_2 - \theta_1)^2 \sin^4(\frac{\varphi_2 + \varphi_1}{2}) + (\phi_2 + \phi_1)} \tag{1}$$

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

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

Two wellbore scenarios are evaluated in this study. The first Eq. (4) consists of five-component sections, and the second Eq. (5) consists of seven component sections constituting the complete well path, as illustrated in Figs. 2 and 3. The overall wellbore length is therefore calculated by summing the lengths of the all component sections that are calculated separately for each well-path design considered.

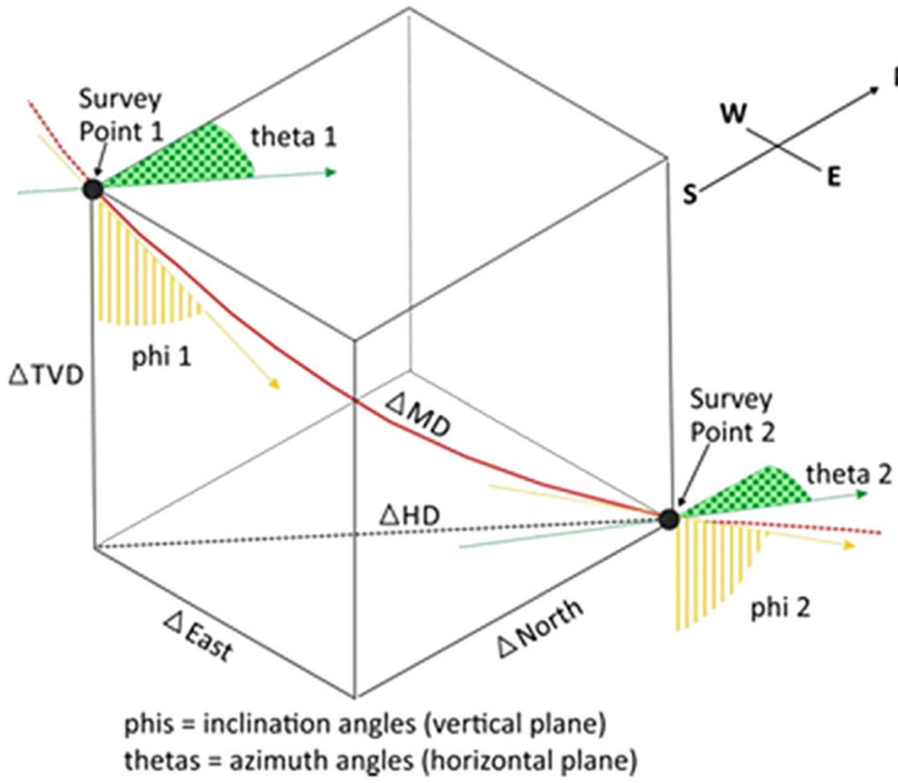
$$\mathrm{TMD}_1 = D_{\mathrm{KOP}} + D_1 + D_2 + D_3 + D_H \tag{4}$$

$$\mathrm{TMD}_2 = D_{\mathrm{KOP}} + D_1 + D_2 + D_3 + D_4 + D_5 + \mathrm{HD} \tag{5}$$

Symbols and abbreviations are explained in Figs. 1, 2 and 3 and in the nomenclature section.

Operational limitations should also be taken into account, e.g., torque and drag (T&D), wear and fatigue of the drill string. For example, a deep kickoff point (KOP) reduces T&D compared to a shallow KOP. There is, therefore, a need to establish a balance between several different factors influencing a well path's design. These factors include T&D, stuck pipe, damage and wear of the drill string, wellbore cleaning and stability. For example, a carefully selected KOP may enable a well to be drilled in less time and at lower cost, but problem formations or shallow reservoir targets can place constraints on where the KOP can be located. Wellbore stability issues related to factors such as wellbore inclination and azimuth also need to be considered. Generally, wells drilled parallel to the direction of least in situ stress demonstrate better stability. Also, in cases where the difference between the maximum and minimum horizontal stress is high, wellbore sections with lower inclination are more stable. Such cases must be considered in determining the inclinations and azimuth well. Dogleg severity (DLS) should also be limited in order to lower the risks of drill pipe failure, casing fatigue and damage, etc. An increase in DLS results in an increase in lateral force that will cause damage and wear. To limit lateral force in order to prevent tool joint damage, Lubinski (1961) recommended a limitation of 2000 lb. DLS proportional to the amount of force can be calculated using Eq. (6) (Devereux 1998).

$$\mathrm{DLS} = \frac{108,000F}{\pi LT} \tag{6}$$



Survey Point 1
theta 1
N
W
E
S
phi 1
ΔTVD
ΔMD
Survey Point 2
theta 2
ΔHD
phi 2
ΔEast
ΔNorth
phis = inclination angles (vertical plane)
thetas = azimuth angles (horizontal plane)

**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

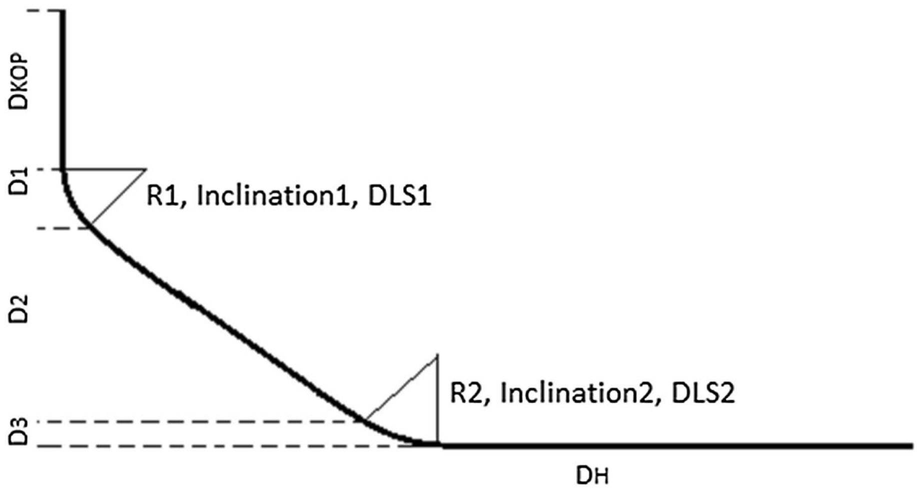


**Fig. 2** The vertical plane of a horizontal well (first case study) with the operational parameters. Note that the scenario involves two build sections

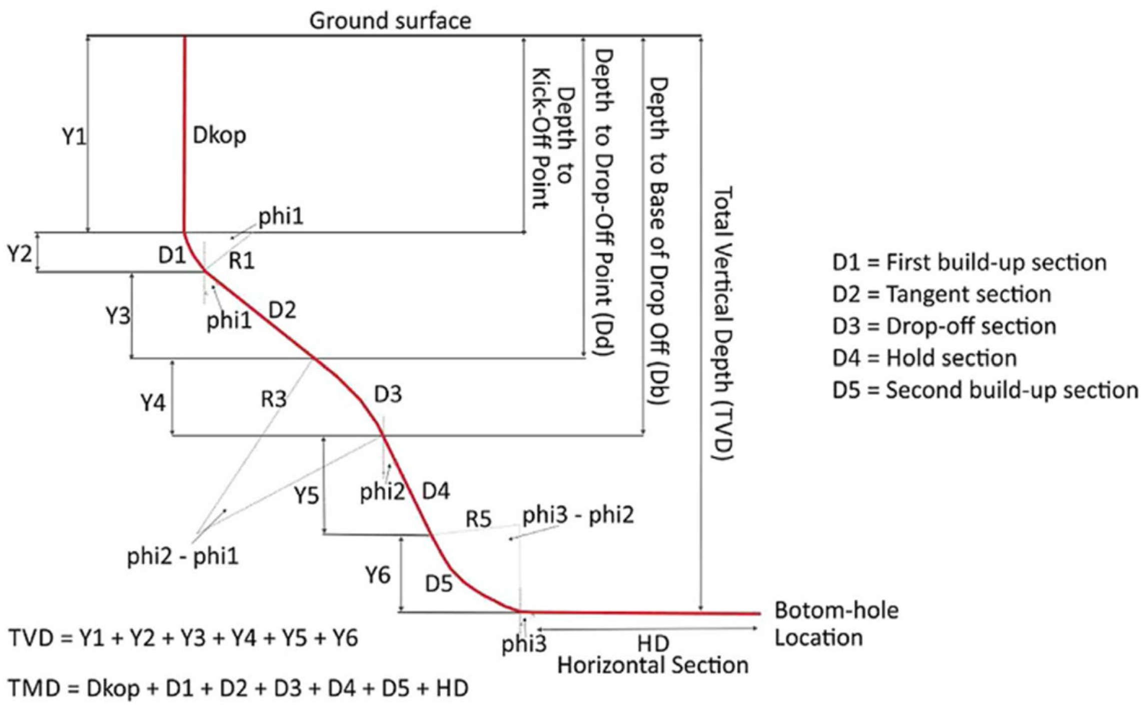

Ground surface
Y1
Dkop
Depth to Kick-Off Point
Depth to Drop-Off Point (Dd)
Depth to Base of Drop Off (Db)
Total Vertical Depth (TVD)
phi1
Y2
D1
R1
Y3
phi1
D2
D1 = First build-up section
D2 = Tangent section
D3 = Drop-off section
D4 = Hold section
D5 = Second build-up section
Y4
R3
D3
Y5
phi2
D4
R5
phi3 - phi2
phi2 - phi1
Y6
D5
Botom-hole Location
TVD = Y1 + Y2 + Y3 + Y4 + Y5 + Y6
phi3
HD
Horizontal Section
TMD = Dkop + D1 + D2 + D3 + D4 + D5 + HD

**Fig. 3** The vertical plane of a horizontal well (second case study) 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

where $F$ is the lateral force, $L$ is half length of the drill pipe and $T$ is tensile force on the pipe at depth of interest. Thus, in the upper part of a well the amount of DLS desirable is limited. In practice, there are typically points along the trajectory where the DLS may exceed the desirable design limits and constraints need to be applied. For example, if 2° is the maximum DLS is desired, the design should consider values less than 2°.

## Case studies

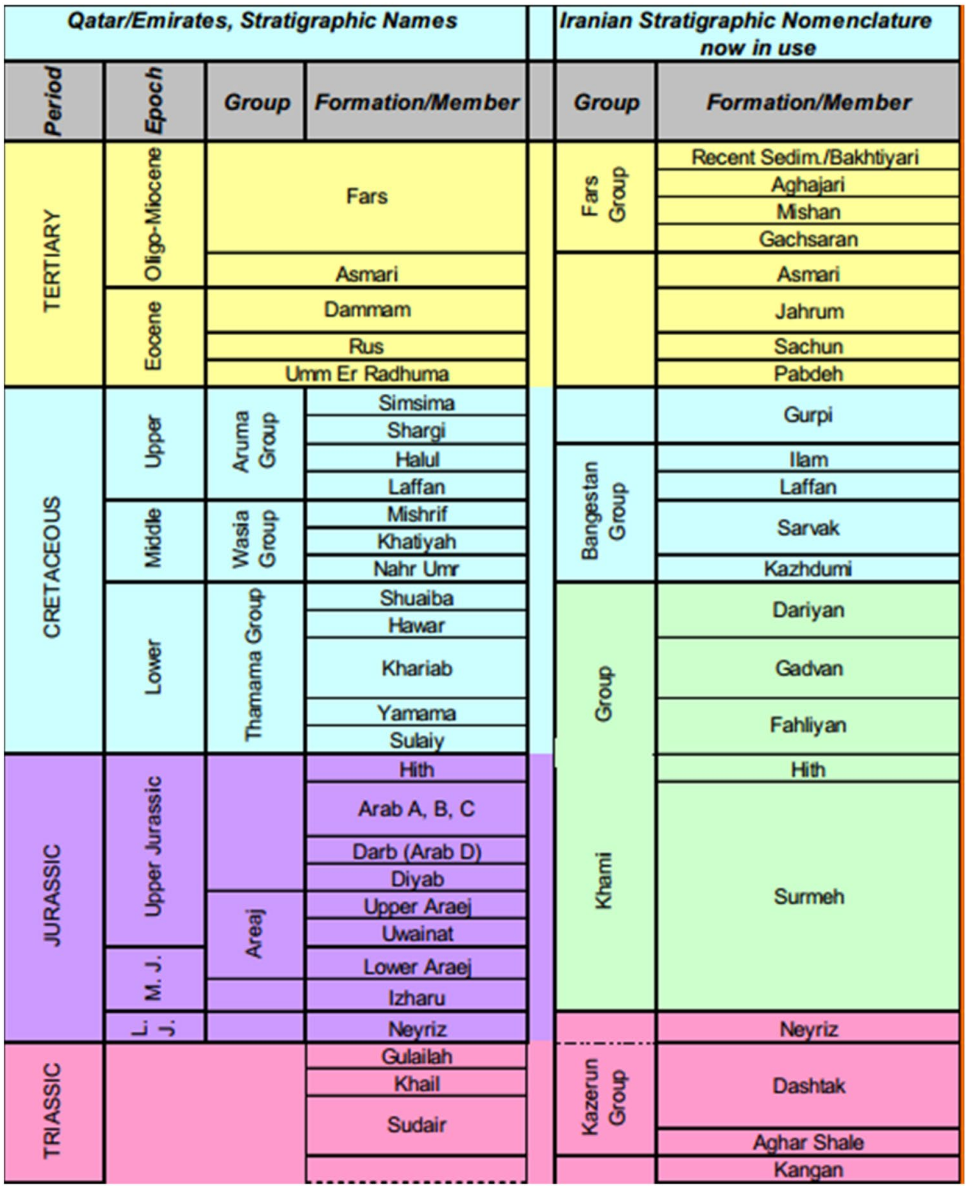
We considered two case studies for examining the performance of the trajectory optimization algorithms. The first case study (Case 1) is for a producing well offshore Iran. The well is a horizontal well and has two build sections. The reservoir section is located at a TVD of 1200 m from the rig's kelly bushing (RKB) and the wellbore involves a horizontal section of 1470 m. The plan view of the well is illustrated in Fig. 2 with operational constraints listed in Table 1. Figure 4 shows the sequence of geological formations encountered in drilling this first case study well.

**Table 1** Constraints applied to the example well path (Case 1—offshore Iran)

| | |
|---|---|
| TVD | 3936 ft (1200 m) |
| $D_H$ | 4821 ft (1470 m) |
| Max DLS for 1st build | 6°/100 ft |
| Inclination angles | $\varphi_1 = 30–50$ |
| | $\varphi_2 = 90°$ |
| Azimuth angle | $\theta = 250°$ |
| Kickoff point depth | 203–688 ft (62–210 m) |
| 2nd build point depth | 3214–3542 ft (980–1080 m) |

In the original plan for the well, the KOP is set at 125 m, in Aghajari formation, but its TVD can range from 62 to 210 m in that formation. The build section should end before the high-pressured Gachsaran formation is encountered in order to set a casing point. The second buildup should be started in the Ilam formation in order to maintain an acceptable lateral force and end before entering the reservoir zone. A long-radius design is applied in order to build angle, as this can achieve a greater offset from the surface location (Carden and Grace 2007).

A 2000-lb maximum lateral force constraint for DLS, already discussed, is applied. Considering a typical bottom-hole assembly (BHA) and drill pipe grade E for directional



| Qatar/Emirates, Stratigraphic Names | | | |
|---|---|---|---|
| Period | Epoch | Group | Formation/Member |
| TERTIARY | Oligo-Miocene | Fars | |
| | | Asmari | |
| | Eocene | Dammam | |
| | | Rus | |
| | | Umm Er Radhuma | |
| CRETACEOUS | Upper | Aruma Group | Simsima |
| | | | Shargi |
| | | | Halul |
| | | | Laffan |
| | Middle | Wasia Group | Mishrif |
| | | | Khatiyah |
| | | | Nahr Umr |
| | Lower | Thamama Group | Shuaiba |
| | | | Hawar |
| | | | Khariab |
| | | | Yamama |
| | | | Sulaiy |
| JURASSIC | Upper Jurassic | | Hith |
| | | | Arab A, B, C |
| | | | Darb (Arab D) |
| | | | Diyab |
| | | Areaj | Upper Araej |
| | | | Uwainat |
| | M. J. | | Lower Araej |
| | | | Izharu |
| | L. J. | | Neyriz |
| TRIASSIC | | | Gulailah |
| | | | Khail |
| | | | Sudair |

| Iranian Stratigraphic Nomenclature now in use | |
|---|---|
| Group | Formation/Member |
| Fars Group | Recent Sedim./Bakhtiyari |
| | Aghajari |
| | Mishan |
| | Gachsaran |
| | Asmari |
| | Jahrum |
| | Sachun |
| | Pabdeh |
| | Gurpi |
| Bangestan Group | Ilam |
| | Laffan |
| | Sarvak |
| | Kazhdumi |
| Khami Group | Dariyan |
| | Gadvan |
| | Fahliyan |
| | Hith |
| | Surmeh |
| | Neyriz |
| Kazerun Group | Dashtak |
| | Aghar Shale |
| | Kangan |

**Fig. 4** Sequence of geological formations encountered by drilling (Case 1—offshore Iran)

drilling, about 60,000 lbs can be imposed on the drill string as the maximum tension in drill string at the depth of the first build section, which results in the maximum DLS permissible for this section of about 6°/100 ft. The drill pipe at the depth of second build section experiences less tension and therefore can tolerate a higher DLS. The DLS for this second build section needs to be less than about 9°/100 ft in order to prevent drill pipe fatigue, but the long-radius build method employed results in the DLS value actually being less than 6°/100 ft. The tangent section-inclusive (TSI) method is applied to the design of the horizontal section of the trajectory. The TSI method enables TVD adjustments for differences in DLS between the overall well plan and actual well path (Garden and Grace 2007). Results from previously drilled offset wells in the field suggest that a hold angle of 30° to 50° is an appropriate inclination angle to maintain stability of the well and we apply this limitation as the geomechanical constraint for our design example.

The second case study (Case 2) has been extracted from (Shokir et al. 2004) and also evaluated by Atashnezhad et al. (2014). The related information and constraints have been summarized in Table 2. The plane view of this well has been presented in Fig. 3.

## Genetic algorithm (GA)

Using a genetic algorithm that follows the main five steps, the optimization has been applied on two wellbore scenarios as mentioned before. The main steps are:

1. Initialize a set of random feasible solutions (called a sample population).
2. Calculate the objective function and then rank the solutions (i.e., best solution is rank#1).

**Table 2** Constraints applied to the example well path (Case 2, after Shokir et al. 2004)

| | |
|---|---|
| TVD | 10,850–10,900 ft (3307–3323 m) |
| HD | 2500 ft (762 m) |
| Dogleg severity | $T_1$, $T_2$, $T_3 \leq 5°/100$ 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 | $\theta_1 = 270°$, $\theta_2 = 270°$, $\theta_3 = 270°$ |
| Max. value of azimuth angles | $\theta_1 = 280°$, $\theta_2 = 280°$, $\theta_3 = 280°$ |
| Kickoff point depth | 600–1000 ft (182–304 m) |
| Draw down point depth | 6000–7000 ft (1829–2134 m) |
| Third build point depth | 10,000–10200 ft (3048–3109 m) |
| 1st casing setting depth | 1800–2200 ft (548–670 m) |
| 2nd casing setting depth | 7200–8700 ft (2195–2652 m) |
| 3rd casing setting depth | 10,300–11000 ft (3140–3353 m) |

3. Assemble high-ranking selected solutions and some randomly generated solutions to act as parents for the next generation (i.e., iteration of the GA).
4. Apply cross-over and mutation on various parent solutions and to create some new solutions to compare with the high-ranking solutions of the previous generation.
5. Rank the new generation: reject the lowest-rank solutions; retain the highest ranking solutions to perpetuate the next generation.
6. Repeat steps 2–5 until the solution converges to an optimum value or a specified number of iterations or computational time has elapsed (i.e., termination criteria are met).

The GA evaluated in this study adopts a process sequence illustrated by the flowchart shown in Fig. 5. For coding the GA for the cases considered, a string of five-component solution for the first case study representing five variables in the trajectory design (KOP, second build depth, inclination angle and DLSs) and 12-component solution representing the 12 variables in the second case study (KOP, second and third build depth, inclination angles, azimuth angles and DLSs). This type of code construction is applied to all the algorithms considered except in ACO.

The main GA operators controlling its behavior are cross-over and mutation, which involve key behavioral parameters for the algorithms. Those behavioral parameters are: (1) cross-over probability ($P_c$); (2) mutation probability ($P_m$); and (3) mutation rate. These three parameters should be set to appropriate so that the GA functions appropriately achieve the best result (Gen and Cheng 2008; Lin et al. 2003). Such tuning of the key behavioral parameters avoids premature convergence (i.e., getting stuck at local optima) and encourages convergence toward the global optimum as quickly as possible (i.e., in the fewest iterations). Multiple runs of the GA for the example well path (see Fig. 6), applying a range of key behavioral parameter values, reveal that the GA performs better when the values of these parameters vary as the generations (iterations) progress rather than maintaining constant values for these parameters across all generations. The best sequence of variations applied to the behavioral parameters was found to be: (1) $P_m$ commencing at a higher value than $P_c$ and then gradually decreasing as the generation evolve; and (2). Whenever the algorithm becomes locked into a local minima, $P_m$ is abruptly increased in an attempt to release the algorithm from that local optima and continue its search for the global optimum.

Table 3 lists the different values of GA behavioral parameters applied in each of the four runs shown in Fig. 6 together with the key performance results of each run.

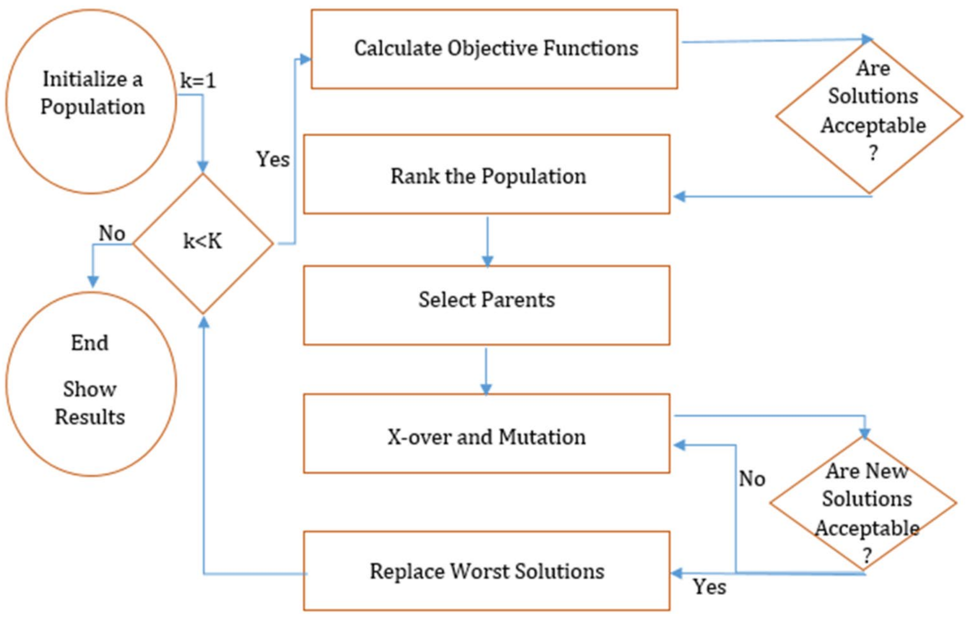


**Fig. 5** Flowchart of GA used for optimizing well bore length of the example well bore. Initially a set of N solutions is generated randomly and their objective function values are calculated and ranked. From the solution set, some solutions are selected as “parents” with the probability of selection proportional to the rank of their objective functions. The parents are combined and some of them are mutated or crossed over to obtain a new solution set potentially involving some solutions with better fitness/objective function values. This process is repeated for multiple iterations up to the maximum number of specified iterations

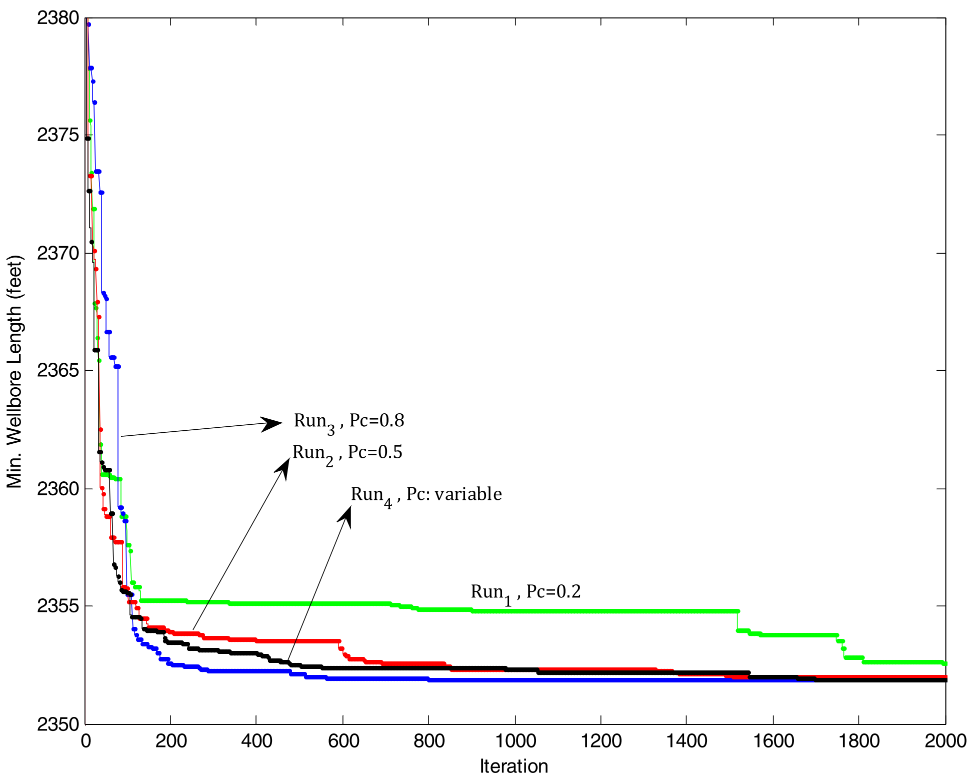


**Fig. 6** Objective function trends compared for variable GA behavioral parameters versus constant-value GA behavioral parameters. The four runs (Case 1) illustrated were all conducted using the same initial population, i.e., the trends all begin with the optimum for an initial random sample population at the left side of the graph. Run 4 (i.e., variable behavioral parameters) shows better performance than other runs (i.e., constant behavioral parameters). The better performance of Run 4 is characterized by it exiting local minima more rapidly than the other runs. Note that the mutation probability in the GA is obtained by the relationship $P_m = 1 - P_c$

## Ant colony optimization (ACO)

Ant colony optimization (ACO), based upon the natural behavior of ants in finding the shortest path from the nest to a specific and identified source of food, can be exploited to find the shortest path among a discrete number of alternative routes distributed in 2D or 3D feasible solution space, honoring defined constraints (Guan et al. 2016). The ACO algorithm applied here to wellbore trajectory optimization is illustrated in Fig. 7.

The selection probability of a path is proportional to the following relation:

$$P_i = \left[ \frac{F_i^n}{\sum_i F_i^n} \right] \tag{7}$$

**Table 3** Key GA behavioral parameters values applied to four runs and the optimal objective function value found after 2000 iterations

| | Run 1 | Run 2 | Run 3 | Run 4 |
|---|---|---|---|---|
| $P_m$ | 0.8 | 0.5 | 0.2 | Variable |
| $P_c$ | 0.2 | 0.5 | 0.8 | |
| Optimal solution found—TMD (m) | 2352 | 2352 | 2354 | 2352 |

The optimal objective function value for each iteration of each run is illustrated as trends in Fig. 6 (Case 1)

The pheromone update rule is calculated as (Guan et al. 2016):

$$F_i^n = C_{\text{evp}} F_i^{n-1} + T_i^n \tag{8}$$

where $P$ is probability of selection for each path, $F$ is pheromone intensity on each path, $C_{\text{evp}}$ is evaporation constant and $T$ is the incremental pheromone concentration to be added at each iteration. Suffix $i$ refers to the $i$th path and $n$ refers to the $n$th iteration. In each iteration, paths selected more frequently will establish, via Eqs. (6) and (7), higher pheromone concentrations. $C_{\text{evp}}$ refers to the effect of pheromone evaporation that prevents excessive accumulation of pheromone concentrations. Higher values of $C_{\text{evp}}$ result shorter run times for the algorithm, but also increase the probability of becoming trapped at a local optima. Table 4 show the results of optimization for various values of $C_{\text{evp}}$. A variable $C_{\text{evp}}$ that decreases as the iterations increase produced an acceptable result with a compromise between algorithm run times and objective function values achieved. The value of $T$, Eq. (8), should be selected so as not to increase the pheromone density rapidly. In this study, $T$ is set to 2 with the initial pheromone density set at 1, and $C_{\text{evp}}$ is able to vary between a range of 0.2 and 1.

**Table 4** Results of optimization for various values of $C$evp in 500 iterations (Case 1)

| $C_{\text{evp}}$ | Running time (s) | Objective function value (m) |
|---|---|---|
| 0.95 | 41 | 2381 |
| 0.75 | 123 | 2377 |
| 0.5 | 249 | 2361 |
| Variable | 165 | 2371 |

**Table 5** ABC algorithm behavioral parameter variations and their impact on the algorithm's performance for the well-path example (Case 1) over 50 iterations

| | Run 1 | Run 2 | Run 3 | Run 4 |
|---|---|---|---|---|
| $L$ | 100 | 500 | 750 | 1000 |
| $a$ | 5 | 5 | 5 | 5 |
| Number of bees in population | 50 | 50 | 50 | 50 |
| Optimal solution found—TMD (ft) after 50 iterations | 2352.65 | 2352.56 | 2352.79 | 2352.82 |
| Computational time (s) | 0.65 | 0.6 | 0.55 | 0.45 |

In order to apply an ACO algorithm to the well trajectory design cases evaluated, it was necessary to modify the decision space. Traditional ACO algorithms are designed to optimize discrete decision spaces, and on other hand, the well trajectory design is composed of a complex and continuous decision space of various parameters such as inclinations, azimuths, doglegs and vertical depths. To convert this

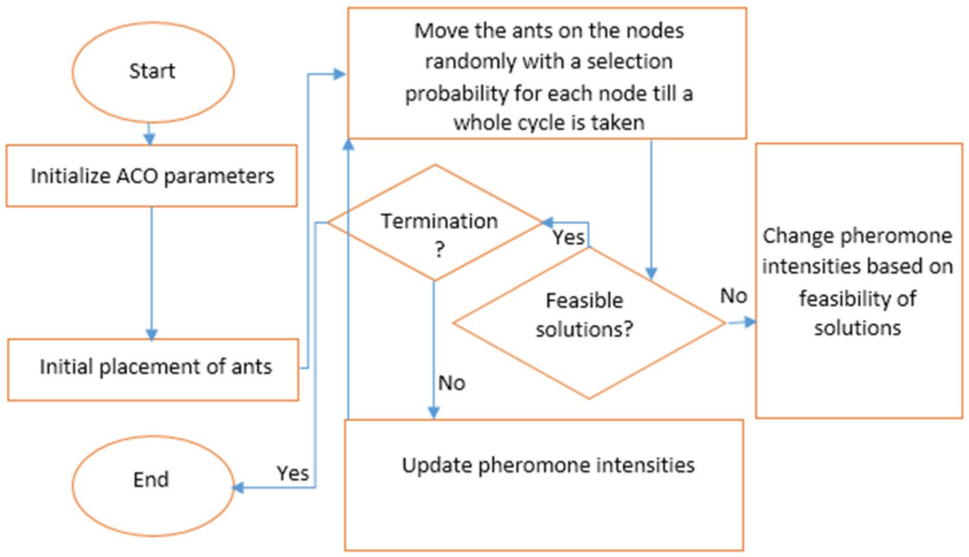

Start
Initialize ACO parameters
Initial placement of ants
End
Yes
Move the ants on the nodes randomly with a selection probability for each node till a whole cycle is taken
Termination ?
Yes
Feasible solutions?
No
No
Update pheromone intensities
Change pheromone intensities based on feasibility of solutions

**Fig. 7** Flowchart of a general ACO algorithm used for optimizing well bore length of the example well bore. The entire feasible range allowable for each variable is divided into subsections, such that each subsection represents a node. Initially each ant is placed at a node and the ants then move through the all nodes in order to complete the well path. The lowest-well-path length dictates the highest pheromone density, which in turn results in higher probability of selection for the next iteration

continuous space into a discrete space, the permissible range of each parameter was divided into smaller sections, with each subsection considered as a node. This modification creates a large number of nodes that slows down the speed of optimization and results in the algorithm consuming high computation time. In addition, large number of nodes necessitate the deployment of larger numbers of ants, which again increases the computation time. Figure 8 shows the trend of optimization of the objective function by dividing the solution variables into sections (nodes) and deploying 10 ants. Increasing the number of ants yields better results in terms of objective function values located, but increases the computation time vice versa.

It is clear that the ACO algorithm applied may produce good results for the well-path optimization problem studied here, but the running time will be very high rather than that obtained by GA in previous section. Actually ACO algorithm is structured and best suited to discrete optimization (Dorigo and Stutzle 2004) and best suited to solving problems with discontinuous decision spaces in the form of specified nodes in the space (Blum 2005, Hatampour et al. 2013). Although ACO can be applied to continuous domains (Socha and Doriga 2008), it is necessary to deform the decision space of the well-path optimization problem studied here into a discontinuous space as described above. Because of high computation time associated with that approach, the ACO algorithms were not developed further. To improve the performance of ACO, one approach would be to hybridize it with other metaheuristics better adapted to deal with continuous solution spaces (e.g., Hu et al. 2008).

## Artificial bee colony (ABC) optimization

Artificial bee colony (ABC) is a more recently developed metaheuristic optimization algorithm than ACO or GA approaches (Karaboga 2005; Karaboga and Basturk 2007, 2008; Karaboga and Ozturk 2009). ABC is a simple evolutionary algorithm using only common control parameters such as colony size and maximum cycle number. This algorithm provides a population-based search procedure described above and expressed by the following equations summarized from Karaboga and Gorkemli (2014):

### Initialization phase

All the vectors of the population of food sources, $X_m$'s, are initialized ($m = 1 \ldots SN$, $SN$: population size) by scout bees and control parameters are set. Since each food source, $X_m$, is a solution vector to the optimization problem, each $X_m$ vector holds n variables, ($X_{mi}$, $i = 1 \ldots n$), which are to be optimized so as to minimize the objective function.

The following definition is used for initialization purposes:

$$X_{mi} = L_i + \text{rand} \times (U_i - L_i) \tag{9}$$

where $L_i$ and $U_i$ are the lower and upper bound of the parameter $X_{mi}$, respectively. And rand is a random number between 0 and 1.

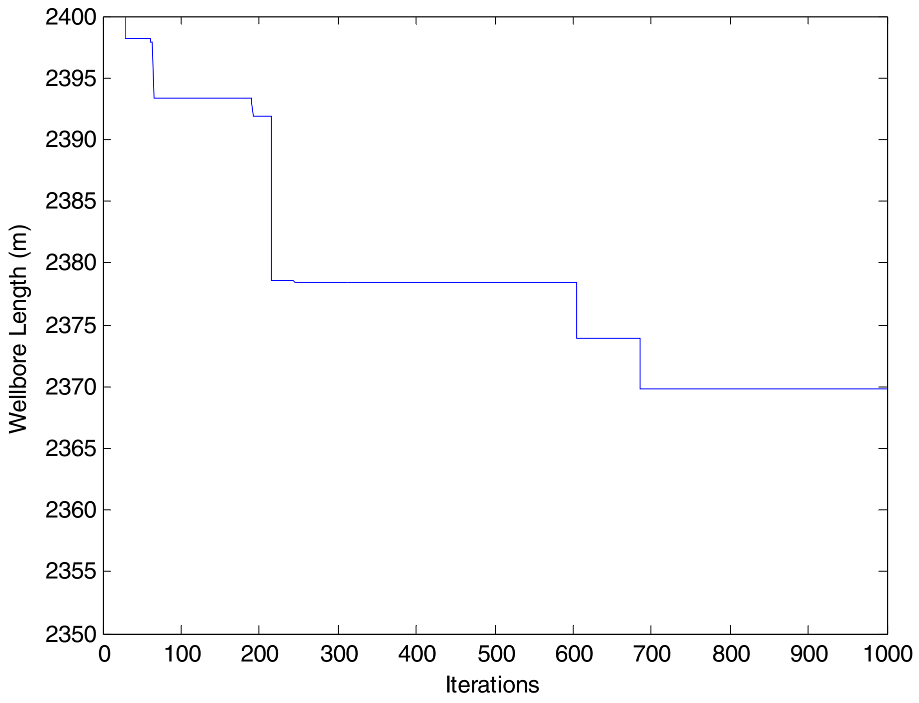


**Fig. 8** The ACO algorithm behavior with 10 ants and 1000 iterations that takes about 150 s to complete (Case 1)

## Employed bees phase

Employed bees search for new food sources ($V_m$) having more nectar within the neighborhood of the food source ($X_m$) in their memory. They find a neighbor food source and then evaluate its profitability (fitness). For example, they can determine a neighbor food source $V_m$ using the formula given by Eq. (10):

$$V_{mi} = X_{mi} + \emptyset_{mi} \times (X_{mi} - X_{ki}) \tag{10}$$

where $X_k$ is a randomly selected food source, $i$ is a randomly chosen parameter index and $\varnothing_{mi}$ is a random number within the range [− a, a]. After producing the new food source $V_m$, its fitness is calculated and a greedy selection is applied between $V_m$ and $X_m$.

The fitness value of the solution, $\mathrm{fit}_m(X_m)$, might be calculated for the problem using the following formula (11):

$$\mathrm{fit}_m(X_m) = \begin{cases} \frac{1}{1+f(X_m)} & ,f(X_m) > 0 \\ 1 + \mathrm{abs}(f(X_m)), & f(X_m) < 0 \end{cases} \tag{11}$$

where $f(X_m)$ is the objective function value of solution $X_m$.

## Onlooker bees phase

Unemployed bees consist of two groups of bees: onlooker bees and scouts. Employed bees share their food source information with onlooker bees waiting in the hive and then onlooker bees probabilistically choose their food sources depending on this information. In ABC, an onlooker bee chooses a food source depending on the probability values calculated using the fitness values provided by employed bees. For this purpose, a fitness-based selection technique can be used, such as the roulette wheel selection method (Goldberg 1989).

The probability value pm with which $X_m$ is chosen by an onlooker bee can be calculated by using the expression given in Eq. (12):

$$P_m = \frac{\mathrm{fit}(X_m)}{\sum_{m=1}^{\mathrm{SN}} \mathrm{fit}(X_m)} \tag{12}$$

After a food source $X_m$ for an onlooker bee is probabilistically chosen, a neighborhood source $V_m$ is determined by using Eq. (10), and its fitness value is computed. As in the employed bees' phase, a greedy selection is applied between $V_m$ and $X_m$. Hence, more onlookers are recruited to richer sources and positive feedback behavior is perpetuated.

## Scout bees phase

The unemployed bees who choose their food sources randomly are called scouts. Employed bees whose solutions cannot be improved through a predetermined number of trials, specified as the "limit" or "abandonment criteria," become scouts and their solutions from previous iterations are abandoned. The converted scouts start to search for new solutions, randomly. For instance, if solution $X_m$ has been abandoned, the new solution discovered by the scout who was the employed bee of $X_m$ can be defined by Eq. (9). Hence, those sources which are initially poor or have failed to be improved by exploitation are abandoned and negative feedback behavior is used to trigger new exploration efforts.

ABC algorithms combine local search processes, carried out by employed and onlooker bees, with global search processes, managed by onlookers and scouts, to achieve a balance between exploration and exploitation efforts. A general ABC algorithm involves the following steps (Karaboga and Gorkemli 2014):

1. Initialize the food source positions, i.e., the target locations (note in the wellbore path problems there is typically one bottom-hole target location).
2. Employed bees identify new food sources within a site of specified dimensions and exploits the best food source within that site.
3. Onlooker bees select a site depending upon the quality of the performance observed from other bees returning to the hive; they detect new food sources within the selected site and exploits the best food source located in that site.
4. Determine which sites should be abandoned and designate the employed bees visiting it to become scout bees to searching randomly for new food sites.
5. Memorize the best food sources found so far within the sites visited.
6. Repeat steps 2–5 until the solution converges to an optimum value or a specified number of iterations or computational time has elapsed (i.e., termination criteria are met).

In ABC algorithms, two parameters play key roles:

1. *Acceleration coefficient* (*a*)—determines the amplitudes of a random number (phi) used to create a new site to explore for bees; and
2. *Abandonment limit parameter* (L)—typically a linear function of several variables including the population size. The abandonment limit establishes an upper boundary of another parameter that records how many times bees encounter a food source at a specific search area. If the abandonment parameter value exceeds the abandon-

ment limit parameter ($L$), that search area (site) must be abandoned and scout bees are instead sent out to search for new promising sites. The purpose of the abandonment limit is to prevent searches becoming trapped at local optima.

Other parameters, such as the number of bees in the population, are effective at impacting the computational time required by ABC algorithms to locate global optimum values of the objective function. The behavioral parameters, $a$ and $L$, are used here to tune the ABC algorithm (Table 5).

With regard to behavioral parameter "$L$," lower values result in greater computational time and more iterations being required to find the global optimum of the objective function. Better outcomes were achieved for the well-path optimization example by systematically varying the value of behavioral parameter "$a$" as the ABC algorithm progresses through its iterations. An initial value for "$a$" (e.g., 1 in the first iteration) multiplied by a linear coefficient (e.g., 1.0005), increasing incrementally from one iteration to the next (according to Eq. 13), combined with higher values for parameter "$L$" were found to produce the best outcomes and fastest convergence for the well-path design cases evaluated.

$$a_{i+1} = a_i \times c \tag{13}$$

where "$c$" is a constant value (e.g., $c = 1.0005$).

## Harmony search (HS) optimization

HS is a relatively recently developed optimization algorithm that has been successfully applied (e.g., Yang 2009) and adapted to solve various optimization challenges (Chakraborty et al. 2009; Daham et al. 2014). It is not though without its critics (e.g., Weyland 2010; Padberg 2012) who conclude that it is similar to other evolutionary algorithms in searching feasible solutions spaces and lacks efficiency by repeatedly retracing previously travelled pathways. The solutions to an optimization problem derived by an HS algorithm are progressively enhanced as a harmony is improved by refining individual improvisations by musicians while the music is being played (Yang 2009). The HS algorithm consists of five distinct steps:

1. Initializing the problem.
2. Initializing the harmony memory (HM).
3. Creating a new harmony (solution).
4. Updating HM.
5. Repeat steps 2–4 until: (a) the solution converges to an optimum value; (b) a specified number of iterations; or (c) computational time has elapsed (i.e., termination criteria are met).

HM is a matrix of N × M dimensions that stores N solutions each consisting of M components or variables. A new solution can be produced either by a random amendment to an element selected from the whole range of available variables, or by a small incremental change to an existing solution from the HM to explore the regions surrounding that known solution. Comparing a random number between 0 and 1 and the harmony memory considering rate (HMCR) forms the basis of deciding the manner in which a new solution will be calculated. If HMCR is set to 0.7 and if the random number generated in each iteration is lower than 0.7, the new solution is created using as a starting point a solution already available in the HM. However, if the random number generated is greater than the specified HMCR value, then the new solution is created randomly from the whole range of available variables (Yang 2009).

Using the HM, the new solution is created using the following equations:

$$X_{\text{NEW}} = X_{\text{OLD}} + B \times \varepsilon \tag{14}$$

$$B = \text{UB} - \text{LB} \tag{15}$$

$X_{\text{OLD}}$ is chosen randomly from the HM, UB and LB are the upper and lower boundaries of the variables. And the ε is a small number between − 1 and +1 causing the new solution to be close to the old one. (Yang 2009).

The key HS behavior controllers are the number of solutions that are stored in the HM, known as harmony memory size (HMS), and the rate of change that is applied to the solutions taken from HM known as harmony memory considering rate (HMCR) (Weyland 2010). A larger harmony memory (i.e., higher HMS) results in a more thorough search of the feasible solution space, but also results in higher computational time. The HS algorithm evaluated in this study adopts a process sequence illustrated by the flowchart shown in Figs. 9, 10.

The effect of the HMCR value on the progress of the HS algorithm is shown in Fig. 11. Higher HMCR values tend to yield the better optimal objective function values as the algorithm progresses. In the HS algorithm applied here, we set HMCR as a variable so that whenever the algorithm becomes trapped at local minima, the HMCR value is abruptly decreased, enabling more as-yet-untested solutions from the entire solution space to enter the HM. Such an approach increases the chances of the algorithm escaping from local minima in which it has become trapped or locked into. Figure 11 compares the HS algorithms performance with different constant HMCR values and compares that performance with higher performance with variable HMCR values.

Another key behavioral parameter of HS algorithms is the mutation rate ($R_m$), which determines what proportion of the variables contributing to the existing HM solutions should

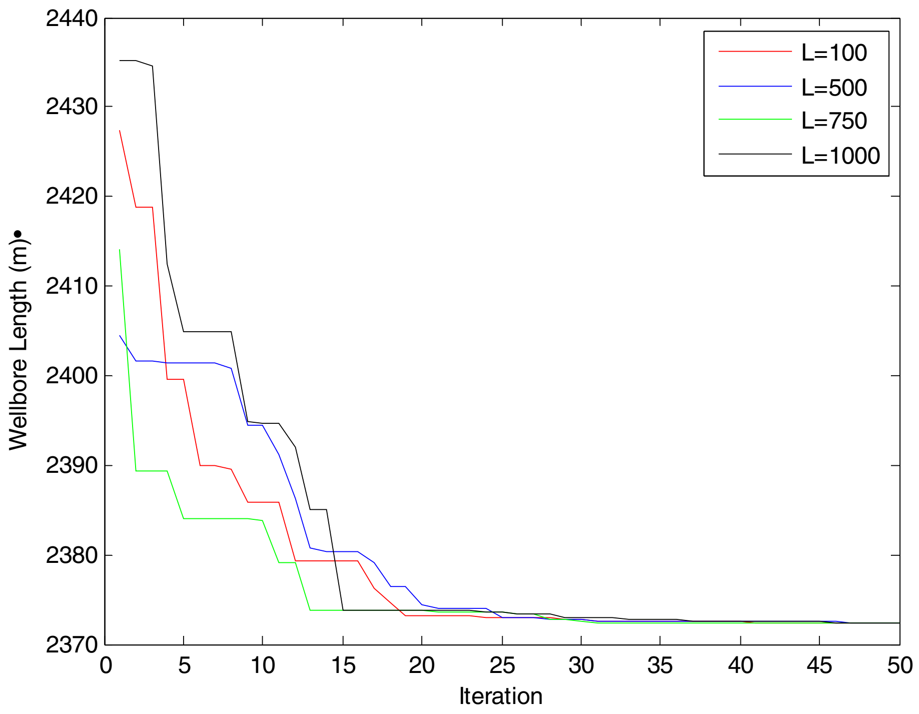


**Fig. 9** Performance of the ABC algorithm for different values of "L" (abandonment limit) applied to the wellbore trajectory (Case 1)

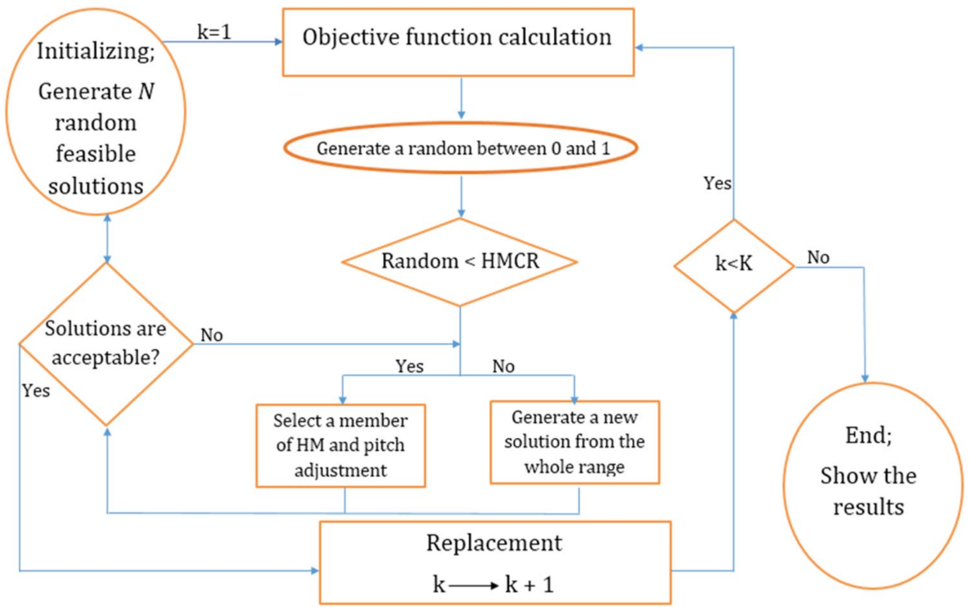

Initializing; Generate N random feasible solutions
k=1
Objective function calculation
Generate a random between 0 and 1
Random < HMCR
Yes
No
Select a member of HM and pitch adjustment
Generate a new solution from the whole range
Solutions are acceptable?
No
Yes
Replacement
k ⟶ k + 1
k<K
Yes
No
End; Show the results

**Fig. 10** Flowchart of HS applied to the wellbore path optimization problem. As for the GA, there is a string of components representing a solution. Each component is responsible for a variable. Initially, a set of solutions is generated randomly and assembled as the harmony memory (HM). Based on their fitness, established from their respective objective function values, the HM solutions are ranked. Applying a probabilistic selection method (a random number from the range 0–1, but closer to 1) a solution with high fitness from the HM is selected and a slight change is applied to it to generate a new solution. That new solution replaces the lowest-ranking solution in the HM. Alternatively, the new solution may be generated independently from the whole range of solutions in the HM dependent on a predetermined threshold for the random number generated. This cycle is repeated over a significant number of iterations until the specified maximum iteration is reached

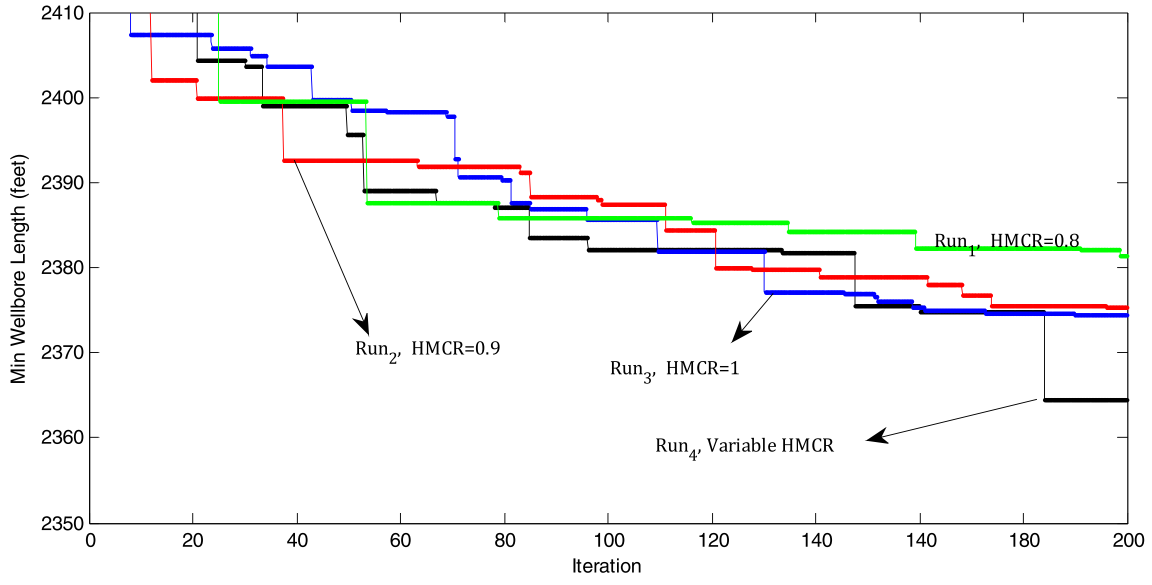


**Fig. 11** Performance comparison of tuned HS versus non-tuned constant HMCR–HS for well-path Case 1. Variable HMCR runs tend to yield better optimum values of the objective function because the algorithm remains trapped at local minima for less iterations than for runs applying fixed-HMCR values. The mutation rate in all runs shown is set at 0.2

be changed to generate new slightly modified solutions to be evaluated. $R_m$ should be as low as possible to make a slight change in good solutions and to explore the solution space around them. A high value of $R_m$ can abruptly disturb the searching process. $R_m$ of 0.05 is applied in this study. A potentially negative feature of the HS algorithm evaluated is that it requires a large number of iterations to achieve an acceptable "optimum" value for the objective function; however, each iteration can be conducted with relatively short computational time.

## Comparative results applying tuned metaheuristic algorithms

Each of the metaheuristic algorithms applied is tuned for the wellbore path optimization Cases 1 and 2 (i.e., as defined in Figs. 1 and 3 and Tables 1 and 2) and then evaluated and compared for performance (i.e., objective function value and computational time consumed). Tables 6 and 7 list the results obtained for the optimization of wellbore Cases 1 and 2 of each algorithm evaluated. It is important to bear in mind that these results were obtained on a PC computer with specifications "Intel Core i5 2430 M 2.4 GHz., 4 GB DDR3 Memory." Attempts to run these algorithms on different computer systems are likely to result in different computation times, but their relative performance order will be the same.

From Fig. 12 and Tables 6 and 7, its apparent that all the algorithms evaluated except ACO are very fast solving these complex cases involving continuous nonlinear solution spaces. GA, ABC and HS showed that in very complex problems, for which there may not be single exact analytical solutions, they can be employed to find an acceptable solution near to global optima. The modified-discrete ACO algorithm developed for this study was found not to be an appropriate solver for wellbore trajectory optimization problems, i.e., involving continuous solution space and large numbers of variables. This finding is perhaps not surprising as the original ACO algorithm was developed to address discrete optimization problems (Dorigo and Stutzle 2004). Clearly A the ACO algorithm is better suited to solve problems with discontinuous decision spaces in the form of specified nodes in the space (Hatampour et al. 2013).

**Table 6** Comparative results of tuned metaheuristic algorithms applied to the wellbore optimization Case 1

| | GA | ACO | ABC | HS |
|---|---|---|---|---|
| Optimal solution found—TMD (m) | 2353 | 2370 | 2352 | 2352 |
| Required number of iterations | 1000 | 2000 | 100 | 2000 |
| Computational time (s) | 1.02 | 300 | 0.46 | 1.50 |

## Discussion

A summary of the converged optima achieved by each metaheuristic algorithm applied to the wellbore path optimization Cases 1 and 2 is provided in Tables 8 and 9, respectively.

**Table 7** Comparative results of tuned metaheuristic algorithms applied to the wellbore optimization Case 2

| | GA | ACO | ABC | HS | PSO from Atashnezhad et al. (2014) |
|---|---|---|---|---|---|
| Optimal solution found TMD (ft) | 15,023 | 15,239 | 15,023 | 15,024 | 15,023 |
| Required number of iterations | 1000 | 2000 | 100 | 2000 | 100 |
| Computational time (s) | 1.60 | 1350 | 5.02 | 2.8 | 2.2 |

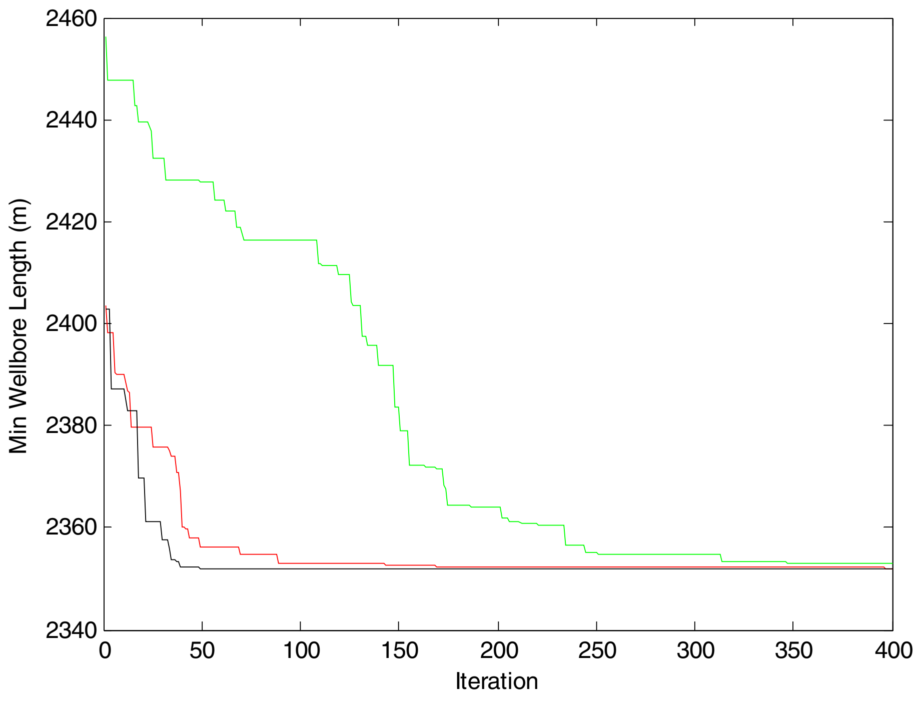


**Fig. 12** Performance trends for GA, ABC and HS algorithms applied to the wellbore optimization Case 1 for 400 iterations. GA and HS may require more iterations to be fully converged

**Table 8** Comparison of the best results achieved by each tuned metaheuristic algorithms applied to the wellbore optimization Case 1

| | GA | HS | ABC | ACO |
|---|---|---|---|---|
| Optimal solution found TMD (m) | 2352 | 2352 | 2352 | 2370 |
| KOP (ft) | 62 | 62 | 62 | 65 |
| 2nd BU depth (ft) | 980 | 980 | 980 | 987 |
| Inclination (°) | 30 | 30 | 30 | 30 |
| 1st DLS (°/100 ft) | 3.76 | 3.76 | 3.76 | 4.07 |
| 2nd DLS (°/100 ft) | 3.90 | 3.90 | 3.90 | 3.90 |

The minimum total measured depth (TMD) located by all studied algorithms is similar (except for ACO), but the computation times and number of iterations taken by each one are different. The GA and HS algorithms require more iterations than ABC and PSO to converge toward the global optima. The same TMD achieved by the algorithms suggest that all have reached good solutions, but still we cannot say these are the absolute global optima.

An important consideration in this study is that for all algorithms evaluated, a strategy of non-constant key-control parameters is applied. For example, in tuning the GA, high cross-over probability typically leads to better performance than high mutation probability. This is applied to the GA in such a way that when the GA becomes trapped around local minima, mutation probability is abruptly increased until an improvement in the GA trend is observed. The same approach has been applied to the HS algorithm such that by default a high value for harmony memory considering rate (HMCR) is applied. However, when the HS becomes trapped around local minima, HMCR is decreased to promote exploration of new solutions spaces. This strategy is applied to all the algorithms evaluated with successful results; multiple runs of each algorithm with the flexibility to vary key parameters have proven that non-constant parameters result in significant improvements in their performance.

**Table 9** Comparison of the best results achieved by each tuned metaheuristic algorithms applied to the wellbore optimization Case 2

| | GA | HS | ACO | ABC | PSO from Atashnezhad et al. (2014) |
|---|---|---|---|---|---|
| Optimal solution found TMD (ft) | 15,023 | 15,023 | 15,251 | 15,023 | 15,023 |
| KOP (ft) | 1000 | 1000 | 749 | 1000 | 1000 |
| 2nd BU depth (ft) | 7000 | 7000 | 6769 | 7000 | 7000 |
| 3rd BU depth (ft) | 10,200 | 10,200 | 10,060 | 10,200 | 10,200 |
| 1st inclination | 10 | 10 | 10 | 10 | 10 |
| 2nd inclination | 40 | 40 | 40 | 40 | 40 |
| 3rd inclination | 90 | 90 | 90 | 90 | 90 |
| 1st azimuth | 273 | 270 | 270 | 270 | 270 |
| 2nd azimuth | 280 | 276 | 273 | 273 | 271 |
| 3rd azimuth | 335 | 340 | 335 | 240 | 332 |
| 1st DLS | 0.83 | 0.82 | 0.82 | 0.84 | 0.77 |
| 2nd DLS | 1.58 | 1.58 | 1.38 | 1.68 | 1.67 |
| 3rd DLS | 4.26 | 4.7 | 3.55 | 3.24 | 3.46 |

An important achievement of the algorithms developed and tuned for this study is that they can solve a nonlinearly constrained problem in less than 2 s (e.g., ABC, GA and HS in Case 1).

The wellbores studied in this paper are evaluated to minimize wellbore length (TMD). Other functions that could be set as objective functions include torque and drag on the drill string. The algorithms could also be examined for multi-objective purposes, but this requires a separate study which is beyond the scope of this work. Another consideration that can be developed by means of metaheuristic algorithms is the anti-collision constraint in a multi-well drilling program. This can represent a very complicated limitation and is the subject of work in progress by the authors.

## Conclusions

Evaluation of a suite of metaheuristic evolutionary algorithms applied to complex well-path optimization provides insight to their relative performance and how they might best be tuned to optimize their performance. Key insights gained from this study are:

1. GA, HS, ABC and PSO are fast-convergence algorithms that can be successfully applied and tuned to solve complex and time-consuming wellbore trajectory design problems.
2. HS is a simple algorithm that typically requires a high number of iterations to converge toward the global optima, but does so in low computation times.
3. For each algorithm evaluated, a set of key-control parameters can be tuned to optimize their performance when applied to specific wellbore trajectory problems. Constant values applied to these parameters did not result in optimal performance. Rather, it was found that changing these parameters progressively within certain ranges through successive iterations resulted in the best performance of each algorithm studied.
4. With the exception of ACO, all algorithms evaluated in this study demonstrate their ability to solve complex wellbore trajectory problems rapidly and locate acceptable solutions close to global optima.

## References

Afshari S, Aminshahidi B, Pishvaei MR (2013) Application of an improved harmony search algorithm in well placement optimization using streamline simulation. J Pet Sci Eng 78(3–4):664–678
Atashnezhad A, Wood DA, Fereidounpour A, Khosravanian R (2014) Designing and optimizing deviated wellbore trajectories using novel particle swarm algorithms. J Nat Gas Sci Eng 21:1184–1204
Blum C (2005) Ant colony optimization: introduction and recent trends. Phys Life Rev 2:353–373
Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ (2009) A survey on metaheuristics for stochastic combinatorial optimization. Nat Comput 8(2):239–287
Carden RS, Grace RD (2007) Directional horizontal drilling manual. Petroskills, Tulsa
Chakraborty P, Roy GG, Das S, Jain D, Abraham A (2009) An Improved Harmony Search Algorithm with Differential Mutation Operator. Fundam Inf 95:1–261
Daham BFA, Mohammed NM, Mohammed KS (2014) Parameter controlled harmony search algorithm for solving the four-color mapping problem. Int J Comput Inf Technol 3(6):13981402
Darquennes D (2005) Implementation and applications of ant colony algorithms. Master thesis, The University of Namur
Devereux S (1998) Practical well llanning and drilling manual. PennWell Books, Houston
Dorigo M, Stutzle T (2004) Ant colony optimization. MIT Press, USA
Gallagher K, Sambridge M (1994) Genetic algorithm: a powerful tool for large-scale nonlinear optimization problems. Comput Geosci 20(7/8):1229–1236
Gen M, Cheng R (2008) Network models and optimization. Springer, Berlin
Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Professional, ISBN: 0201157675

Guan Z-C, Liu Y-M, Liu Y-W, Xu Y-X (2016) Hole cleaning optimization of horizontal wells with the multidimensional ant colony algorithm. J Nat Gas Sci Eng 28:347–355

Guria C, Goli KK, Pathak AK (2014) Multi-objective optimization of oil well drilling using elitist non-dominated sorting genetic algorithm. J Pet Sci 11(1):97–110

Hatampour A, Razmi R, Sedaghat MH (2013) Improving performance of a neural network model by artificial ant colony optimization for predicting permeability of petroleum reservoir rocks. Middle-East J Sci Res 13(9):1217–1223

Hu XM, Zhang J, Li Y (2008) Orthogonal methods based ant colony search for solving continuous optimization problems. J Comput Sci Technol 23(1):2–18

Karaboga D (2005) An Idea Based on Honey Bee Swarm for Numerical Optimization. Erciyes University, Engineering Faculty

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8:687–697

Karaboga D, Gorkemli B (2014) A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Appl Soft Comput 23:227–238

Karaboga D, Ozturk C (2009) A novel clustering approach: artificial Bee Colony (ABC) algorithm. Appl Soft Comput 11(2011):652–657

Khalili M, Kharrat R, Salahshoor K, Haghighat Sefat M (2013) Fluid injection optimization using modified global dynamic harmony search. Iran J Oil Gas Sci Tech 2(3):57–72

Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Method Appl Mech Eng 194 (36–38):3902–3933

Lin WY, Lee WY, Hong TP (2003) Adapting crossover and mutation rates in genetic algorithms. J Inf Sci Eng 19:889–903

Lubinski A (1961) Maximum permissible dog legs in rotary boreholes. J Pet Technol Dallas 13(2):256–275

Mansouri V, Khosravanian R, Wood DA, Aadnoy BS (2015) 3-D well path design using a multi objective genetic algorithm. J Nat Gas Sci Eng 27(1):219–235

Nozohour LB, Fazelabdolabadi B (2016) On the application of Artificial Bee Colony (ABC) algorithm for optimization of well placements in fractured reservoirs; efficiency comparison with the Particle Swarm Optimization (PSO) methodology. J Pet 2(1):79–89

Padberg M (2012) Harmony Search Algorithms for binary optimization problems. In: Klatte D, Lüthi HJ, Schmedders K (eds) Operations Research Proceedings 2011. Springer, Berlin, Heidelberg

Shokir EM, Emera MK, Eid SM, Wally AW (2004) A new optimization model for 3-D well design. Emir J Eng 9(1):67–74

Sivanandam SN, Deepa SN (2008) Introduction to genetic algorithms. Springer, Berlin

Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155–1173

Weyland D (2010) A rigorous analysis of the harmony search algorithm how the research community can be misled by a "novel" methodology. Int J Appl Metaheur Comput 1–2(April-June):50–60

Yang XS (2009) Harmony search as a metaheuristic algorithm. Stud Comput Intell, Springer, Berlin 191:1–14

Zerafat MM, Ayatollahi S, Roosta AA (2009) Genetic algorithms and ant colony approach for gas-lift allocation optimization. J Jpn Pet Inst 52(3):102–107