Genetic algorithms and scenario reduction

by M. Armstrong*, A. Vincent*†, A. Galli*, and C. Méheut†

Synopsis

Scenario reduction is designed for selecting a representative subset of geostatistical simulations out of a much larger set. Three steps are involved: measuring the dissimilarity between two simulations; finding a metric to measure the distance between any subset of *k* simulations and the full set of *N* simulations; and finding an efficient algorithm for selecting the subset that minimizes the metric. This paper focuses on the third question. We show that genetic algorithms are an efficient way of approaching the minimum when the population of subsets to be sampled is large. Two case studies based on the Walker Lake data-set are presented: firstly choosing *k*=4 simulations out of a total of 100, and secondly choosing 20 out of the same 100 simulations.

In the first case it was possible to compute all possible combinations exhaustively and hence to demonstrate that the algorithm converges to the true minimum. This was not possible in the second case. Instead we demonstrate that it outperforms the random draw algorithm used in earlier work. A procedure for tracking individual selections during the iterative procedure was developed. This allows us to measure the evolution in the percentage of progeny resulting from crossing-over and from mutation that survived in the next generation. It gives valuable insight into how to choose the parameter values in the algorithm. Another key finding is that there is a trade-off between the number of individuals per generation and the number of generations required for the algorithm to converge.

Keywords
genetic algorithms, crossing-over, mutants, Walker Lake.

Introduction

Nowadays many more conditional simulations of orebodies, reservoirs, and aquifers can be generated than in the past. In some applications it is possible to post-process all of them but in others this is impossible. For example, in mining, pit optimization and scheduling are computer-intensive and time-consuming; similarly for fluid flow simulations and production optimization in the oil industry. When only a certain number of the conditional simulations (*k*, say) can be post-processed, the question is how to choose a representative set of that size out of the full set of *N* simulations. This process can be split into three steps:

1. Measuring the dissimilarity between two simulations
2. Finding a metric to measure the distance between the full set of *N* simulations and any given subset of size *k*
3. Finding an efficient algorithm for selecting the best subset (i.e. the one that minimizes the metric.

Armstrong *et al.*, (2010, 2013) proposed using a metric denoted by $D(J,q)$[1], based on the scenario reduction metric[2] developed by Heitsch and Romisch (2009), together with a random search algorithm. The procedure gave very encouraging results when it was used to select subsets containing k = 10, 12, or 15 simulations out of a total of N = 100 simulations. More recently we have started working on cases in the oil industry with larger values of *k* and *N*, where the total population of possible subsets is much larger. Table I gives the total numbers of subsets for different values of *k* and *N*. For very small values of *k*, the best strategy is to test all the subsets exhaustively. Otherwise an efficient search procedure is required. The difficulty in finding the subset that minimizes the metric in such a large discrete population is that standard gradient-based methods cannot be used. Genetic algorithms seem better suited to this.

[1]The procedure we developed for measuring the dissimilarity between two simulations and the metric *D*(*J*,*q*) is summarized in Appendix 1.

[2]The German research group led by Heitsch and Romisch specializes in the stochastic optimization of large systems, using multi-stage programming with recourse (e.g. for electricity prices or hydroelectric systems). A branching tree structure is used to model the evolution of prices over time or of the water input into dams. The problem is that the number of branches in the tree increases exponentially with time, and sooner or later the tree has to be pruned. As each of the price paths is called a scenario, the procedure for pruning the tree is called 'scenario reduction'. In their work Heitsch and Romisch recognized the fact that the tree is not perfectly known and took this into account when developing their scenario reduction metric *D*(J,q).

* *Cerna, Mines-Paristech, Paris, France.*
† *Aviomex Pty Ltd, LascauxLafarge, France.*

Table I

Number of combinations of *k* simulations selected from *N* simulations without replacement

	k = 1	*k* = 3	*k* = 4	*k* = 5	*k* = 10	*k* = 12	*k* = 15	*k* = 20
N = 100	100	1.62e+5	3.92e+6	7.53e+7	1.73e+13	1.05e+15	2.53e+17	5.36e+20
N = 700	700	5.69e+7	9.92e+9	1.38e+12	7.30e+21	2.63e+25	3.12e+30	2.49e+38

This paper proposes an improved algorithm that uses genetic algorithms for selecting the best subset. The procedure is tested on two examples: firstly for subsets with $k = 4$ simulations out of a set of 100 simulations and secondly for subsets of $k = 20$ from the same set of 100 simulations. In the first case, as the metric can be computed exhaustively for all possible subsets, we were able to check whether the genetic algorithm found the true minimum or was trapped in a local minimum. In the second case, we compared the genetic algorithm with the random search procedure developed earlier.

The paper is structured as follows. In the next section we describe the genetic algorithm that we have used. In the following section we give background information on the 100 simulations that are used in both case studies. Then the first case study is presented. We show that the genetic algorithm effectively reaches the global minimum. We also show that there is a trade-off between the number of individuals in each generation and the number of generations required to converge. The more individuals per generation; the fewer generations are needed. The second case study is presented in the following section. Even with a total population of 5.36 x 10^{20} combinations, genetic algorithms give much better results than those obtained with the random search procedure. Other cases with even larger sets of simulations are currently being studied. Our experience to date has shown that the speed of convergence in large cases depends on the choice of the parameter values used, namely, the number of individuals in each generation and the number of progeny obtained by crossing-over and by mutation. In order to understand the impact of these parameters on the convergence we developed a method for tracking individual selections over time. The survival rates of parents, progeny from crossing-over, from mutations of a single gene and of all the genes, from one generation to another, were evaluated. We show how this evolves over time.

The conclusions are given in the last section. For the sake of brevity, the procedure developed for measuring the dissimilarity between two simulations and the metric $D(J,q)$ is given in Appendix 1. One of the reviewers of the paper asked why we focus only on minimizing the metric $D(J,q)$ instead of comparing the values of the objective function computed using the full set of simulations rather than on a subset of them. This is a very good question. Heitsch and Romisch (2009) showed that minimizing the metric $D(J,q)$ is equivalent to minimizing metric (2) (Appendix 2). for all objective functions f in a fairly wide class. So we only need to find the subset Q that minimizes $D(J,q)$. A more detailed explanation is given in Appendix 2.

Genetic algorithm used

Sastry, Goldberg, and Kendall (2005) provide a good description of genetic algorithms, which were first invented in the mid-1970s (Holland, 1975), together with a review of developments up to 2005. Their usefulness in solving many difficult real-world applications has been demonstrated over the past 40 years; for example, for electric power systems (Valenzuala and Smith, 2002; Burke and Smith, 2000); for scheduling university courses (Paechter *et al.*, 1995, 1996) and university examinations (Burke and Newall, 1999); for rostering nurses (Burke *et al.*, 2001); for warehousing (Watson *et al.*, 1999), and for scheduling sports (Costa, 1995) and machines (Cheng and Gen, 1997).

The algorithm we use is similar to that of Cao and Wu (1999). Each individual in the population is represented by a vector of length k. These vectors are considered as 'chromosomes' with k genes. The algorithm selects two individuals in the population to be parents, and gets them to mate by crossing the chromosomes to produce 'children'. Occasionally a spontaneous mutation occurs. As in natural selection, the fittest of the children are more likely to be the parents of the succeeding generation. The probability of an individual being chosen to be a parent depends on its fitness.

In our application the k genes in each chromosome represent the numbers of the k simulations to be kept. At the outset 10 000 individuals (vectors) were generated by drawing k numbers at random between 1 and N. The fittest 1000 of these were then selected to reproduce. Here the fitness is defined as $1/D(J,q)$. The algorithm selects two individuals to be parents. The gene cross-over is carried out by picking a cross-over point P at random between 1 and k (included). The first P genes from one parent are then joined to the $(k\text{-}P)$ genes from the other parent. Figure 1 illustrates a crossing-over.

Mutation is another important feature of natural selection and of genetic algorithms. A position within each chromosome is selected at random. A gene (a simulation number between 1 and N) is selected at random to replace the gene at that location. Figure 2 illustrates a mutation in which simulation no. 66 is mutated into simulation no. 29. The others remain unchanged. Figure 3 summarizes the procedure that we have programmed.

This procedure of mating and mutation is repeated

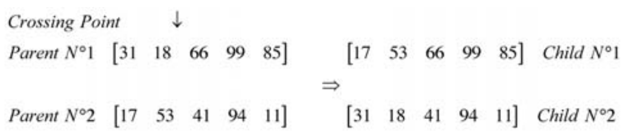
Crossing Point ↓
Parent N°1 [31 18 66 99 85]
Parent N°2 [17 53 41 94 11]
⇒
[17 53 66 99 85] Child N°1
[31 18 41 94 11] Child N°2

Figure 1—Crossing-over. The two chromosomes from the parents cross over at a random point. The two genes on the left from Parent no. 1 are joined to the three on the right from Parent no. 2, giving child no. 1 with mixed genes, and likewise for the second child

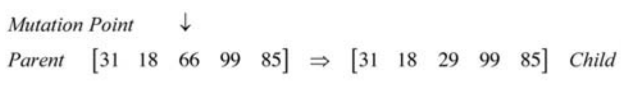

Figure 2—Mutation. A gene chosen at random is changed to another randomly selected gene. Here simulation no. 66 is replaced by simulation no. 29

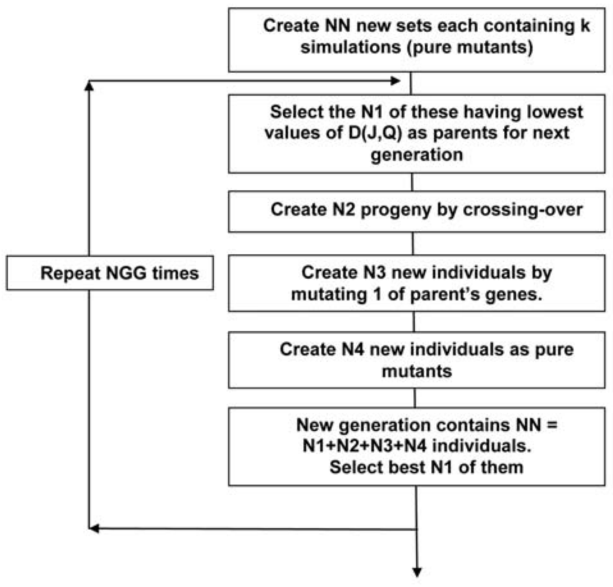
Create NN new sets each containing k simulations (pure mutants)
Select the N1 of these having lowest values of D(J,Q) as parents for next generation
Create N2 progeny by crossing-over
Repeat NGG times
Create N3 new individuals by mutating 1 of parent's genes.
Create N4 new individuals as pure mutants
New generation contains NN = N1+N2+N3+N4 individuals. Select best N1 of them

Figure 3—The genetic algorithm that was used. Initially *NN* new sets of *k* simulations are created by drawing the simulation numbers at random (pure mutants). The best *N*1 of these are selected to be the parents for the next generation (i.e. those with the lowest values of *D*(*J*,*Q*)). Next, *N*2 new individuals are created by crossing-over from the parents, then *N*3 individuals are created by mutating one gene in one of the parents (1-mutants), and lastly *N*4 pure mutants are created. The best *N*1 individuals are selected from the *NN* = *N*1+*N*2+*N*3+*N*4 individuals in that generation. This is repeated NGG times

through *NGG* generations. At each step the fitness of all the individuals in the population is computed by calculating the inverse of $D(J,q)$ for each individual and the fittest are kept. It would be possible for individuals with repeated values of simulation numbers (e.g. 1, 1, 2, 3 ...) to be created by mutation or crossing-over. As they turn out to have higher values of $D(J,q)$ they are eliminated rapidly.

Background information on two case studies

The simulations used in both case studies were generated for a case study on the impact of hedging on a hypothetical gold mine in Australia. They were based on the Walker Lake data-set (Isaaks and Srivastava, 1989), except that the grades were modified to reproduce the statistical characteristics of the Kisladag gold deposit in Turkey and the size of the blocks and selective mining units was rescaled to match its annual production. See Armstrong *et al.* (2010 and 2013) for details. As the profitability of mines depends primarily on the recoverable reserves, especially the quantity of metal recovered, we chose to use the metal above cut-off in each panel to measure the dissimilarity between simulations. To be more precise, each simulation is represented by a vector giving the metal above cut-off for a series of 16 possible cut-offs for each panel. In studies using the Walker Lake data, the area is usually divided into 30 panels each containing 100 selective mining units. So we computed the metal above cut-off for each panel for the 16 cut-offs corresponding to different possible gold prices. Each simulation was represented by a vector of length 480 = 30 x 16. The underlying idea is that two simulations are 'very similar' if the metal above cut-off in one simulation is very close to that of the other simulation for every cut-off and for every panel[3].

We say that these vectors are a proxy for the simulations because they encapsulate the essential characteristics of the simulation in a shortened form. In contrast to mining, the proxies for oil and gas reservoir simulations and aquifer simulations should reflect their fluid flow characteristics.

Having computed the proxy for each simulation, we compute an $N \times N$ matrix D of the dissimilarities between simulations where N is the total number of simulations. Let S denote the selected subset of k simulations; let J be the subset of the remaining $(N - k)$ simulations. The metric $D(J,q)$ between the subset S and the full set of simulations is computed using the procedure described in Appendix 1. In previous work (Armstrong *et al.*, 2010, 2013), subsets containing 12 simulations were considered so there was a total of about 1.05×10^{15} possible combinations. In the second example here we select subsets of 20 simulations so the population to be sampled contains 5.36×10^{20} possible combinations, that is, there are 100 000 times as many candidate subsets. This is why we are looking for a more efficient sampling method.

Results for first case study: *k* = 4, *N* = 100

The first step was to find the true global minimum for subsets containing 4 simulations out of a total of 100 simulations. This involved computing the metric $D(J,q)$ exhaustively for about 3.9 million possible subsets. Figure 4 shows the histogram of all the values of the metric. The global minimum turned out to be 0.2564.

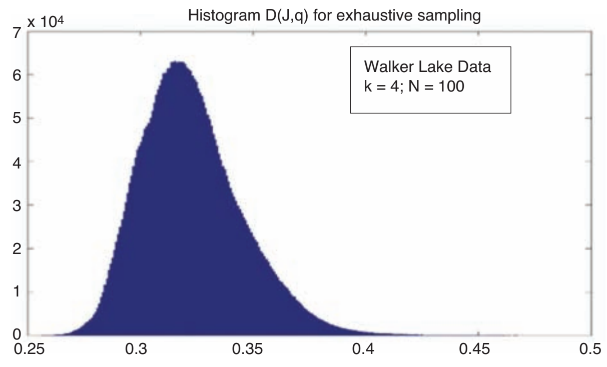

Figure 4—Exhaustive histogram of the values of the metric *D*(*J*,*q*) obtained from 3 921 225 possible subsets of 4 simulations from 100 simulations. The minimum is 0.2564

3Strictly speaking, this dissimilarity measure is not a distance because a zero value could be obtained for two simulations that were not identical simply by permuting the selective mining units within one or more panels.

The next step consisted of running the genetic algorithm 100 times with different initial sets of 1000 subsets. All of the runs of the genetic algorithm reached the global minimum after at most 8 generations. Figure 5 shows the evolution of the metric for all hundred runs of the genetic algorithm together with the global minimum (shown by the dotted red line). We had wondered if there was enough 'genetic material' in a population of this size to reach the global minimum. If not, the genetic algorithm might have been trapped in a local minimum and been unable to get out of it. Our worries were in vain.

We then repeated the test using a much larger population containing 10 000 subsets instead of 1 000. The 10 runs of the genetic algorithm all reached the global minimum but after 4 generations instead of 8 (Figure 6). This shows that there is a trade-off between the number of subsets considered in each generation and the number of generations required to reach the minimum.

Results for second case study: $k = 20$ $N = 100$

Before running the genetic algorithm, we carried out 2 million random draws of 20 numbers between 1 and 100. The lowest value of $D(J,q)$ found by the random search procedure was 0.1946 compared a mean of 0.2216 and a standard deviation of 0.0077. Figures 7 presents the histograms of all 2 million values of $D(J,q)$ (left) and of the lowest 5000 values (right).

As the genetic algorithm was run for 50 generations, each with 10 000 subsets, about 500 000 subsets were evaluated. The minimum value of $D(J,q)$ found was 0.1868, compared to 0.1946 after evaluating 2 million randomly selected subsets. This confirms that the genetic algorithm is more efficient at finding subsets with low values of the metric.

Figure 8 presents the evolution of the minimum value of the metric $D(J,q)$ shown by the solid line and also the maximum (fine line) and the mean (dotted line). As expected the minimum decreases monotonically as a function of the number of generations. The mean also decreases steadily but the maximum is more or less constant. This is because new randomly drawn subsets are being included. The genetic algorithm has effectively succeeded in generating lots of promising subsets with values of the metric that are much lower than the random search procedure used earlier.

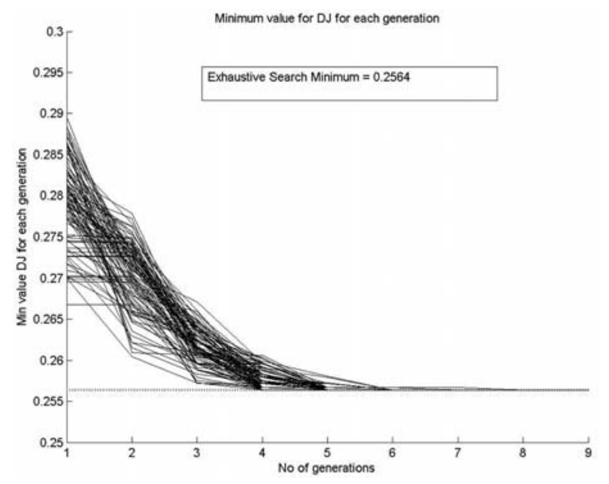

Figure 5—Evolution of the minimum value of the metric *D*(*J*,*q*) for 100 runs of the genetic algorithm each containing 1000 possible subsets of 4 simulations from 100 simulations. All of the runs reached the absolute minimum 0.2564 after at most 8 generations

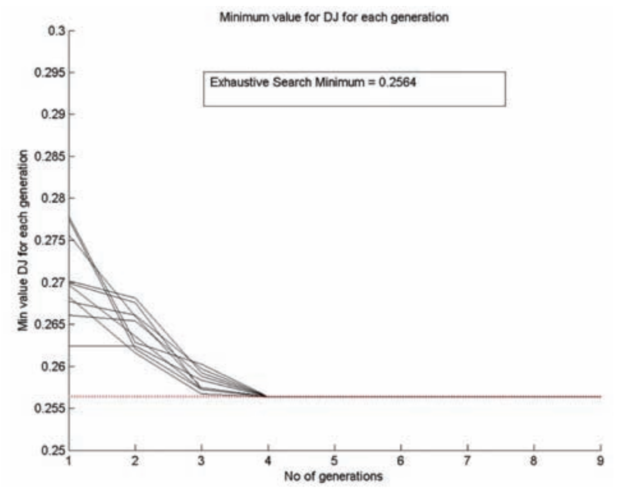

Figure 6—Evolution of the minimum value of the metric *D*(*J*,*q*) for 10 runs of the genetic algorithm each with 10 000 possible subsets of 4 simulations from 100 simulations. All of the runs reached the absolute minimum 0.2564 after at most 4 generations. There is a trade-off in computational effort between the number of generations and the population of subsets per generation

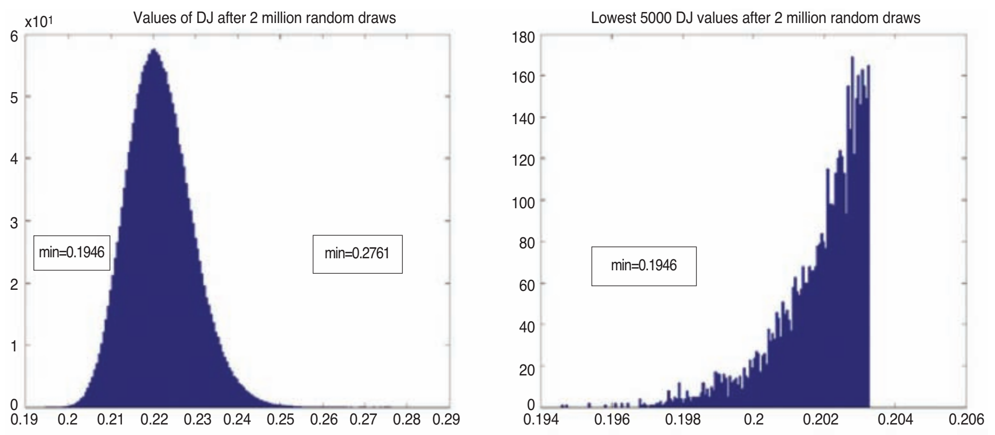

Figure 7—Histograms of all 2 million values of the metric *D*(*J*,*q*) obtained by the random search procedure (left) and of the lowest 5000 of the values

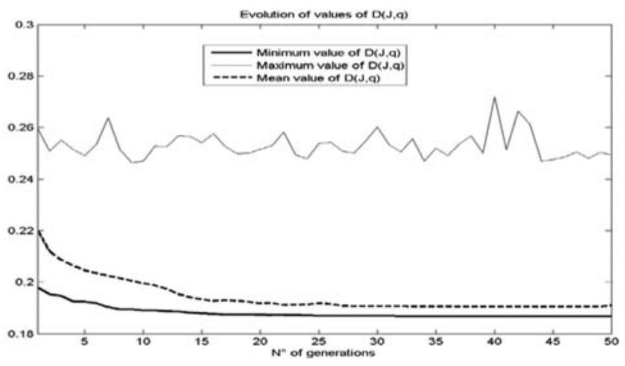

Figure 8—The evolution of the minimum value of *D*(*J,q*) obtained by the genetic algorithm (solid line), and also the mean (dotted line) and the maximum (thin line). Each generation consisted of 10 000 subsets. As expected, the minimum decreases monotonically as a function of the number of generations. The mean also decreases steadily but the maximum is more or less constant. This is because new randomly drawn subsets are being included

Tracking individual selections

As both the genetic algorithm and the random search procedure are iterative procedures, there is no guarantee of their reaching the true minimum. In this example we found that 8009 of the 10 000 subsets in the 50th generation had a value of $D(J,q)$ equal to the minimum 0.1868. We found that they were all identical but they had appeared at different times. Table II presents the 7 of these 8009 subsets with the value of the metric in the first column followed by the identity number of that subset, those of its parents, and then the generation number when it first appeared. The first of these subsets was first created in generation no. 32 as a result of a mutation. The '0' for the second parent indicates a mutation of one gene from a parent as opposed to a cross-over. In addition, a certain number of pure mutants are introduced in each generation in order to ensure that the genetic material can be renewed. Pure mutants can be distinguished because they have '0' for both parents. We refer to individuals created by mutating one gene as 1-mutants to distinguish them from pure mutants.

The fact that the simulation numbers in all 8009 subsets are identical suggests that the algorithm has either reached the true minimum or is trapped in a local one.

Percentage of parents, progeny, and mutants surviving to the next generation

Having a method for tracking individual selections makes it possible to follow their evolution over time from one generation to another. This is important when choosing how many progeny, 1-mutants, and pure mutants to create each generation. To be more specific, we wanted to know how many of these four classes survived from one generation to another. Figure 9 shows the percentage of each class that survives over time from the first generation until the 50th. At the outset about 10% of each type of individual survived to the next generation. For the first few generations the pure mutants (mauve) are important, but this drops off rapidly. For the first 8–10 generations the 1-mutants (red) have a high survival rate. The survival rate for progeny (created by crossing-over) drops off more slowly. After 35 generations, most of the parents survive to the next generation. So the population is quite fit (with low values of the metric $D(J,q)$.

Looking at these results one might be tempted to reduce the number of pure mutants and 1-mutants after about 25 generations, but the last few drops in the value of the metric (i.e. after 35 generations) turned out to be due to mutants. We interpret this as meaning that new genetic material has to be introduced into the gene pool in order to find the minimum, or else it would become too narrow – 'too inbred', one might say.

Local minima

Two problems when using iterative algorithms are the criterion for stopping and avoiding being trapped in local minima. In our case, these problems increase in importance as the size of the subset to be sampled (and the computer

Table II

The ID number, those of their two parents and the generation in which they first appeared, for 7 of the 8009 subsets with the minimum value of *D*(*J,q*) found by the genetic algorithm. The subsets are all identical but were found by different paths, starting out from a subset created by a mutation in generation no. 32 (the '0' for the second parent indicates a mutant)

	D(J,q)	ID no.	Parent no.1	Parent no. 2	No. of generations
1	0.1868	287232	265613	0	32
2	0.1868	295801	287232	262044	33
3	0.1868	301516	287232	271762	34
4	0.1868	313743	271179	300883	35
5	0.1868	316606	313706	309055	36
6	0.1868	442001	308331	316186	50
7	0.1868	448245	316223	316488	50

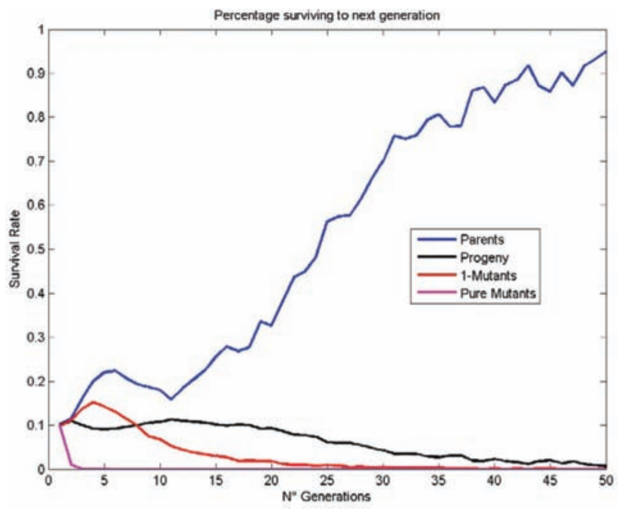

Figure 9—Proportion of parent sets of parents (blue), progeny (black), 1-mutants (red), and pure mutants (mauve) that survive to the next generation, as a function of the number of generations

time required) increases. In this section we show that while it would be tempting to assume that the algorithm has converged when the value of $D(J,q)$ stops decreasing for some time, the algorithm may well be trapped in a local minimum. To illustrate this point, we ran the genetic algorithm ten times for the second case using the following parameters: 10 000 individuals per generation; 1000 were retained as parents ($N1$) for the next generation; 2000 progeny ($N2$) were created by crossing-over; 7500 individuals ($N3$) were obtained by mutating 1 gene; and a further 500 were pure mutants ($N4$). This was repeated over 500 generations. Figure 10 shows the minimum value of $D(J,q)$ per generation for the first 50 of the 500 generations for the 10 repetitions. The value of $D(J,q)$ did not drop after that point. Three distinct long-term minima were found for $D(J,q)$: 0.1868349 (5 cases out of 10 repetitions), 0.1871285 (2 cases), and 0.1871484 (3 cases). At first we expected the algorithm to have identified one particular set of k simulations corresponding to each minimum. We were surprised to find that this was not the case. Figure 11 shows all the values of $D(J,q)$ for the top 1000 individuals in each generation. Note how the range of values narrows as the number of generations increases, until all the individuals have the same $D(J,q)$ value. In fact, the individuals are identical. This means that new genetic material brought by the 1-mutants and the pure mutants was not 'fit' enough to be selected for the following generation; nor were the new individuals created by crossing-over.

Looking back at Figures 10 and 11, it is now clear that we had wasted time and effort continuing to run the algorithm for 500 generations; it had reached a local minimum after 50–60 generations. The algorithm was unable to get out of the local minimum because all the individuals were the same. It could not create 'fit enough' individuals to survive the Darwinian selection process. We are currently modifying certain aspects of the algorithm to overcome this problem.

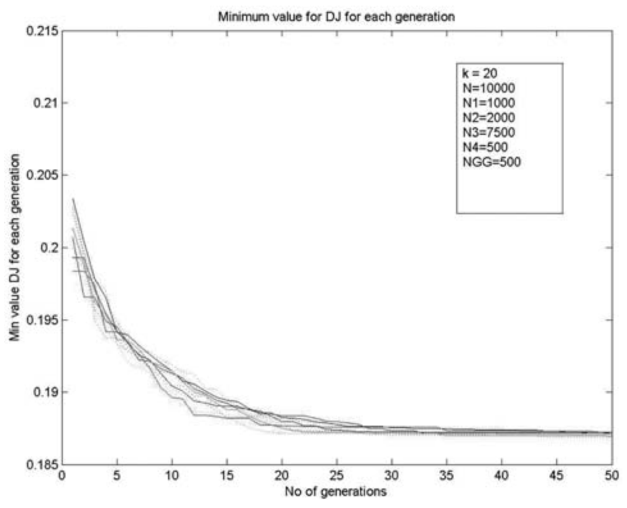

Figure 10—Lowest value of $D(J,Q)$ in each generation for 50 of the 500 generations. This was repeated 10 times

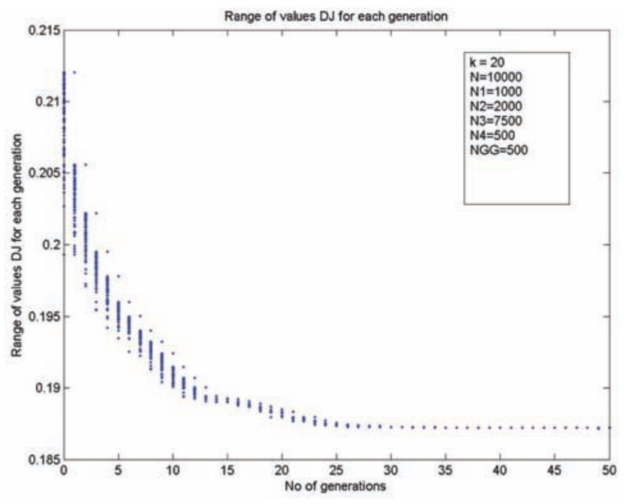

Figure 11—Range values of $D(J,Q)$ amongst the best $N1$ individuals in each generation for 50 of the 500 generations, for a typical run (out of the 10 runs). Note how the range shrinks over time

Conclusions and perspectives for future work

In the two tests presented and in others that we have carried out, the genetic algorithm outperformed the random search method used earlier. We believe that this is true in general. However, we still need to know more about its performance characteristics. Firstly, what criterion should be used to stop the algorithm? An arbitrary number of generations, and in that case how many? Or some criterion based on the algorithm's performance?

Secondly, we need a better understanding of how the genetic algorithm functions, particularly when the number of combinations in the space to be sampled is large. Does the crossing-over mechanism contribute more than mutations? Is one more efficient early on and the other more useful later on? How many individuals should there be in each generation? In the first example, we showed that there is a trade-off between the number of generations required and the number of individuals per generation. The more individuals per generation, the faster the value of $D(J,q)$ drops – but at the cost of more computations. Further work is required to clarify these points and also to find ways to create 'fit enough' individuals in order to avoid getting trapped in local minima. Having said that, while it is important from a theoretical point of view to understand the convergence properties, this is not primordial in practice because the sets of simulations that make up the local minima are all very similar.

References

Armstrong, M., Ndiaye, A.A., and Galli, A. 2010. Scenario reduction in mining. *IAMG Annual Meeting*, Budapest, 29 August – 2 September 2010.

Armstrong, M., Ndiaye, A.A., Razantsimba, R., and Galli, A. 2013. Scenario reduction applied to geostatistical simulations. *Mathematical Geosciences*, vo. 45, February 2013. pp.165–182. DOI 10.1007/s11004-012-9420-7

Burke, E.K. and Newall, J.P. 1999. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, vol. 3. pp. 63–74.

Burke, E.K. and Smith, A.J. 2000. Hybrid evolutionary techniques for the maintenance scheduling problem. *IEEE Transactions on Power Systems*, vol. 15. pp. 122–128.

BURKE, E.K., COWLING, P.I., DE CAUSMAECKER, P., and VANDEN BERGHE, G. 2001. A memetic approach to the nurse rostering problem. *Applied Intelligence*, vol. 15. pp. 199–214.

CAO. Y.C. and WU Q.H. 1999. Teaching genetic algorithms using Matlab. *International Journal of Electrical Engineering Education*, vol. 36. pp 139–153.

CHENG, R.W. and GEN, M. 1997. Parallel machine scheduling problems using memetic algorithms. *Computers and Industrial Engineering*, vol. 33. pp. 761–764.

COSTA, D. 1995. An evolutionary tabu search algorithm and the nhl schedule problem. *INFOR*, vol. 33. pp. 161–178.

DUPACOVÁ, J., GRÖWE-KUSKA, N., and RÖMISCH, W. 2003. Scenario reduction in stochastic programming: an approach using probability metrics. *Mathematical Programming*, vol. 95. pp. 493–511.

HEITSCH, H. and ROMISCH, W. 2009. Scenario tree modelling for multistage stochastic programs. *Mathematical Programming A*, vol. 118, no. 2. pp. 371–406. DOI 10.1007/s10107-007-0197-2

HOLLAND, J. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

ISAAKS, E.H. and SRIVASTAVA, R.M. 1989. An Introduction to Applied Geostatistics. Oxford University Press, New York. p. 561

PAECHTER, B., CUMMING, A., and LUCHIAN, H. 1995. The use of local search suggestion lists for improving the solution of timetable problems with evolutionary algorithms. *Evolutionary Computing: AISB Workshop 1995*. Fogarty, Y. (ed.). *Lecture Notes in Computer Science*, vol. 993. Springer, Berlin. pp. 86-93.

PAECHTER, B., CUMMING, A., NORMAN, M.J., and LUCHIAN, H. 1996. Extensions to a memetic timetabling system. *The Practice and Theory of Automated Timetabling I*. Burk, E.K. and Ross, P. (eds). *Lecture Notes in Computer Science*, vol. 1153. Springer, Berlin. pp. 251–265.

RACHEV, S.T. and RÖMISCH, W. 2002. Quantitative stability in stochastic programming: the method of probability metrics. *Mathematics of Operations Research*, vol. 27. pp. 792–818.

SASTRY, K.D., GOLDBERG, D.E., and KENDALL, G. 2005. Genetic algorithms: a tutorial. *Search Methodologies: Introductory Tutorials in Optimization, Search and Decision SupportMethodologies*. Burke, E.K. and Kendall, G. (eds.). Springer, Berlin. pp. 97–-125.

VALENZUALA, J. and SMITH, A.E. 2002. A seeded memetic algorithm for large unit commitment problems. *Journal of Heuristics*, vol. 8. pp. 173–196.

WATSON, J.P., RANA, S., WHITELY, L.D., and HOWE, A.E. 1999. The impact of approximate evaluation on performance search algorithms for warehouse scheduling. *Journal of Scheduling*, vol. 2. pp. 79–98.

Appendix 1: The dissimilarity matrix *D* and the metric *D*(*J*,*q*)

Dissimilarity matrix D

The first step is to choose a proxy to represent the simulations. This is a vector that encapsulates the main features of simulations. For example, in open-pit mining applications, it could be the quantity of metal in each panel above a set of cut-offs; in petroleum engineering it could be streamline fluid flow simulations. The next step is to construct the square $N \times N$ matrix of distances between pairs of proxies. This matrix is of course symmetric with zeros down the diagonal.

Metric D(J,q)

Let $\prod$ be the probability measure associated with the initial set of N geostatistical simulations which are denoted by ξ^i. for $i = 1, \dots N$. Let p_i be the probability of the ith simulation. In general geostatistical simulations are considered to be equally probable so $p_i = 1/N$. We want to find a probability measure Q with only k simulations, that is as close as possible to $\prod$. Let S be the set of the numbers of the k simulations to be retained; let J be the set of the numbers of the (N-k) others. Let q_j be the probability of the jth simulation to be retained. In contrast to p_i, the probabilities q_j are no longer equally likely. We use the metric developed by Heitsch and Romisch:

$$D(J;q) = \mu\left(\sum_{i=1}^{N} p_i \delta_{\xi^i}, \sum_{j \notin J} q_j \delta_{\xi^j}\right)$$

where μ is the Kantorovitch functional.

When this metric $D(J,q)$ is transposed to geostatistical simulations, it is computed from the dissimilarity matrix in the following way. The rows and columns of the dissimilarity matrix are rearranged so that the k simulations to be retained (e.g. in S) are placed before the other (N-k) simulations (e.g. in J). The dissimilarity matrix can now be partitioned as:

$$\begin{bmatrix} D_{Ss} & D_{SJ} \\ D_{JS} & D_{JJ} \end{bmatrix}$$

Computing the new probabilities q

In order to evaluate $D(J,q)$ we first determine the probabilities q_j in the new measure. This is done by taking the simulations in J one by one and finding the member of S that is closest to each one. The probability of the simulation being eliminated is then assigned to the closest member of S. Speaking figuratively, each member of S ends up as the 'head' of a group consisting of itself plus those members of J that are closer to it than to any other member of S. So its new probability q_j is the sum of its own initial probability p_i plus those of the others in its group. Some members of S find themselves at the centre of a large group; other groups have only a few members, while some are loners (singletons). Having determined the new probabilities, it is easy to compute the metric $D(J,q)$.

A short example

The easiest way to illustrate these two steps is via a short example. Suppose that we want to select 5 simulations out of a total of 20 equally probable geostatistical simulations. Suppose that S ={2, 7, 12, 13, 15} and J ={1, 3, 4, 5, 6, 8, 9, 10, 11, 14, 16, 17, 18, 19, 20}. The rows and columns in the dissimilarity matrix are rearranged. Table III (a) gives the sub-matrix D_{JS}.

Taking the members of J one by one, find the lowest value in each row in D_{JS}. For example, the closest member of S to simulation no. 1 (in J) is simulation no. 12. In fact, simulation no.12 is the closest one to 9 of the simulations in J (no. 1, 3, 6, 8, 10, 16, 17, 19, and 20). These are highlighted in yellow in Table IIIa. So its new probability is (9+1)/20 = 0.5. Five of the simulations in J highlighted in blue are closest to no. 7 (no. 5, 9, 11, 14, and 18). So the new probability for no. 7 is (5+1)/20 = 0.3. One simulation (no. 4) is closest to no. 13, so its new probability is 0.1 and the remaining two simulations (no. 2 and no.15) are singletons that are far from any other simulations. Table IV gives the new probabilities.

Table III

(a) Five simulations no. 2, 7, 12, 13, and 15 are to be retained. The matrix D_{JS} below is used for assigned the other 15 simulations to the closest one out of the 5 retained, as shown by the coloured highlighting, (b) Minimum value (left column) and closest simulation (right column). *D*(*J,q*) is computed by summing the row minima each multiplied by their probability. As the 20 geostatistical simulations are equally probable, it suffices to sum the minima and multiple by 0.05

(a)	2	7	12	13	15
1	.626	.400	.376	.627	.696
3	.743	.490	.327	0.953	.828
4	.826	.439	.497	.395	.513
5	.692	.290	.393	.708	.530
6	.702	.432	.276	.762	.794
8	.451	.330	.284	.784	.557
9	.800	.291	.353	.397	.464
10	.554	.331	.267	.843	.633
11	.601	.238	.259	.654	.741
14	.795	.284	.309	.616	.511
16	.416	.340	.264	.618	.505
17	.485	.281	.209	.778	.575
18	.800	.184	.299	.641	.497
19	.779	.536	.447	.862	.699
20	.443	.299	.290	.563	.477

(b)	Minimum	Closest
1	.376	$1 \rightarrow 12$
3	.327	$3 \rightarrow 12$
4	.395	$4 \rightarrow 13$
5	.290	$5 \rightarrow 7$
6	.276	$6 \rightarrow 12$
8	.284	$8 \rightarrow 12$
9	.291	$9 \rightarrow 7$
10	.267	$10 \rightarrow 12$
11	.238	$11 \rightarrow 7$
14	.284	$14 \rightarrow 7$
16	.264	$16 \rightarrow 12$
17	.209	$17 \rightarrow 12$
18	.184	$18 \rightarrow 7$
19	.447	$19 \rightarrow 12$
20	.290	$20 \rightarrow 12$
D(*J,q*) = 4.422 x 0.05 = 0.2211		

Table IV

The new probabilities *q* for the 5 simulations selected

Simulation no.	2	7	12	14	15
Probability *q*	0.05	0.30	0.50	0.10	0.05

In order to compute the value of *D*(*J,q*) we go back to Table III. Each of the row minima is multiplied by the original probability of that simulation. Since there are 20 equally likely simulations, the probabilities are all 0.05. In this example the value of *D*(*J,q*) is 0.2211, that is the sum of the row minimums (0.422) multiplied by 0.05, as shown in Table IIIb.

Appendix 2: The stability property of the objective function

One of the reviewers of the paper asked why we focus only on minimizing the metric *D*(*J,q*) when the overall objective is the accuracy of the objective function computed over the selected subset of the simulations compared to the full set. The answer is because of the stability properties of the stochastic programming problem with respect to small perturbations. Here we summarize the explanation given in Heitsch and Romisch (2003) based on work by Dupacova *et al.* (2003) and Rachev and Romisch (2002).

Our overall objective when using a set of simulations is to optimize an objective function *f* over all the simulations weighted by their probabilities. This is reformulated in stochastic decision problems as

$$n\left\{\int_{\Omega} f_0(\omega,x)P(d\omega) : x \in X\right\} \qquad [1]$$

where

$X \subset R^m$ is a nonempty closed convex set

Ω is a closed subset of R^s

the function *f* from $\Omega \times R^m$ to *R* is continuous with respect to and convex with respect to *x*

P is a fixed Borel probability measure on Ω.

Typically the integrand *f* is not differentiable but is locally Lipschitz continuous on Ω. In Dupacova *et al.* (2003) and Rachev and Romisch (2002) it is shown that model (1) is stable with respect to small perturbations in terms of the probability metric:

$$\underset{fCF}{sup}\left|\left\{\int_{\Omega} f_0(\omega,x)P(d\omega) - \int_{\Omega} f_0(\omega,x)Q(d\omega)\right\}\right.$$

Heitsch and Romisch (2003) show that minimizing the metric *D*(*J,q*) is equivalent to minimizing (2) for all objective functions *f* in a fairly wide class. We only need to find the subset *Q* that minimizes *D*(*J,q*) in order to minimize the absolute difference between the approximate objective function obtained using the subset *Q*, and that obtained using the full set of simulations. Readers can consult Heitsch and Romisch's paper for the mathematical proofs, and the exact conditions on the objective function *f*.

Interpreting these results in terms of geostatistical simulations

The probability measure *P* gives the probabilities of all the initial scenarios. In general the geostatistical scenarios are considered to be equally likely. The probability measure *Q* gives the probabilities of simulations in the reduced subset; these are not equally likely. Some simulations are 'typical' whereas as others are less likely. They could correspond to 'bonanzas' or 'disasters', which are both important for decision-makers.

Expression [2] gives an upper bound on the absolute difference between the objective function obtained using the full set *P* and the reduced set *Q* for all possible objective functions. ◆