Genetic Algorithms and Scenario Reduction

Margaret Armstrong[1], Arnaud Vincent[2], Alain Galli[3] and Christophe Méheut[4]

Abstract Scenario reduction is designed for selecting a representative subset of geostatistical simulations out of a much larger set of simulations. 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 very large. Two case-studies are presented.

Introduction

Nowadays many more conditional simulations of ore-bodies, reservoirs and aquifers can be generated than in the past. In some applications it is possible post-process all of them but in others this is impossible. For example, in mining pit optimisation and scheduling are computer intensive and time consuming; similarly for fluid flow simulations and production optimisation 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 one (i.e. the one that minimizes the metric.

[1] Cerna, Mines-Paristech, 60 Boulevard Saint-Michel, 75272 Paris, France, margaret.armstrong@mines-paristech.fr

[2] Cerna, Mines-Paristech, 60 Boulevard Saint-Michel, 75272 Paris, France, arnaud.vincent@mines-paristech.fr

Aviomex Pty Ltd., LascauxLafarge, 87500 Saint Yrieix la Perche, France
arnaud.vincent@aviomex.com

[3] Cerna, Mines-Paristech, 60 Boulevard Saint-Michel, 75272 Paris, France,
alain.galli@mines-paristech.fr

[4] Aviomex Pty Ltd., LascauxLafarge, 87500 Saint Yrieix la Perche, France
christophe.meheut@aviomex.com

Armstrong et al ([1] & [2]) proposed using a metric denoted by D(J,q)[5], based on the scenario reduction metric[6] developed by Heitsch & Romisch [3], 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 with larger values of k and N, where the total population of possible subsets is much larger. Table 1 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 minimises the metric in such a large discrete population is that standard gradient-based methods cannot be used. Genetic algorithms seem better suited to this.

Table 1 Number of combinations of k simulations selected from N simulations without replacement.

	k = 1	k = 3	k = 5	k = 10	k = 15	k = 20
N = 100	100	1.62e+5	7.53e+7	1.73e+13	2.53e+17	5.36e+20
N = 700	700	5.69e+7	1.38e+12	7.30e+21	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 got stuck 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. The first case-study is presented in Section 4. We show that the genetic algorithm effectively reaches the global minimum. Section 5 presents the second case-study. Even with a total population of 5.36×10^{20} combinations, genetic algorithms give much better results than those obtained with the random search procedure. Other

[5] To save space, the procedure we developed for measuring the dissimilarity between two simulations and the metric D(J,q) are not described in the text. They summarized in Appendix 1.

[6] The German research group led by Heitsch and Romisch are specialists in the stochastic optimisation 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 & 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).

cases with even larger sets of simulations are currently being studied. Further work is underway on the convergence properties of these types of algorithms.

Genetic algorithm used

Genetic algorithms were first invented in the mid-70s [4]. The algorithm we use is similar to [5]. 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 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-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 N°66 is mutated into simulation N°29. The others remain unchanged.

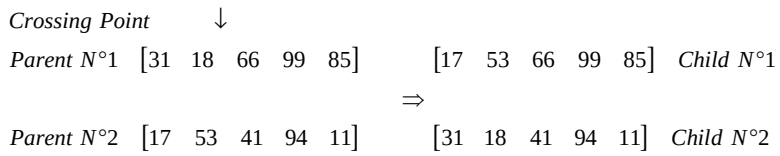

Figure 1 Crossing-over: The two chromosomes from the parents cross-over at a random point. The two genes on the left from Parent N°1 are joined to the three on the right from Parent N°2, giving child N°1 with mixed genes, and likewise for the second child.

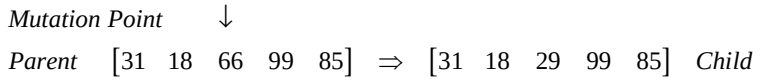

Figure 2 Mutation: A gene chosen at random is changed to another randomly selected gene. Here simulation N° 66 is replaced by simulation N° 29.

This procedure of mating and mutation is repeated through NG 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.

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 [6] 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 [1] and [2] 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 cutoff in each panel to measure the dissimilarity between simulations. To be more precise, each simulation is represented by a vector giving the metal above cutoff 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 cutoff 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 cutoff in one simulation is very close to that of the other simulation for every cutoff and for every panel[7].

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 proxy 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 x 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 the Appendix. In previous work [1] and [2], 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.

[7] Strictly 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.

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 3 shows the histogram of all the values of the metric. The global minimum turned out to be 0.2564.

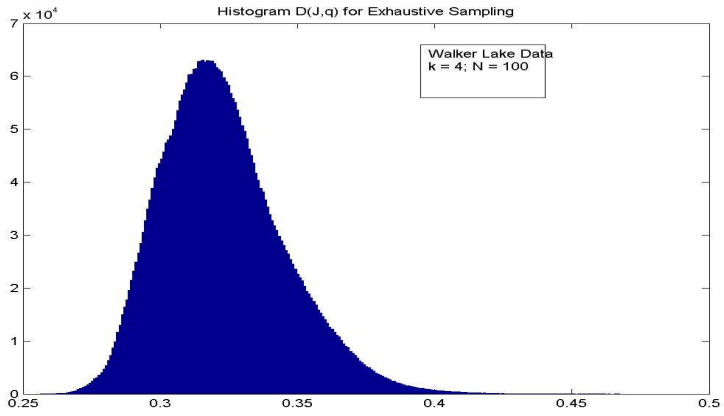

Figure 3 Exhaustive histogram of the values of the metric D(J,q) obtained from 3,921,225 possible subsets of 4 simulations from 100 simulations. The minimum is 0.2564.

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 4 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 got stuck 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 of subsets 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 5). 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.

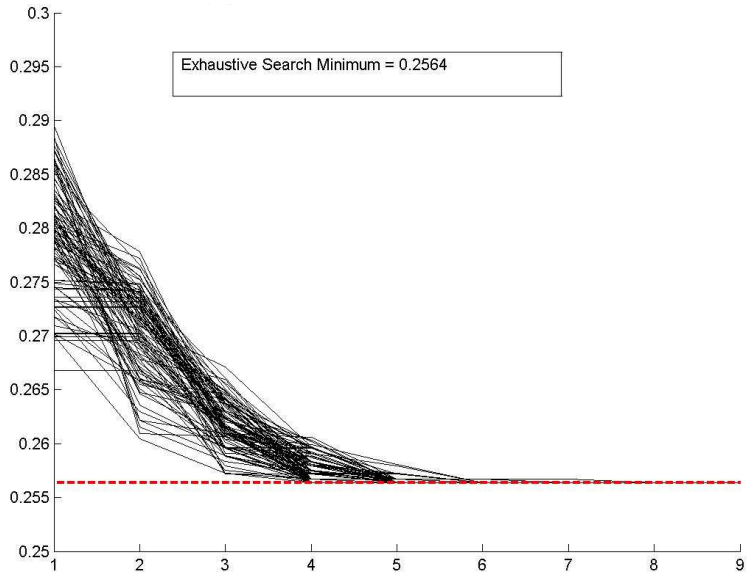

Figure 4: Evolution of the metric D(J,q) for 100 runs of the genetic algorithm each containing 1000 possible subsets of 4 simulations from 100 simulations. All of the runs reached the absolute minimum 0.2564 after at most 8 generations.

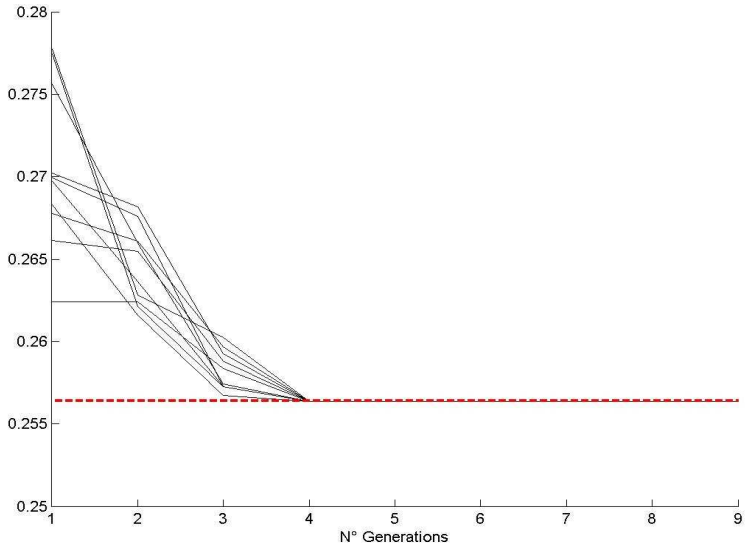

Figure 5: Evolution of the metric D(J,q) for 10 runs of the genetic algorithm each with 10,000 possible subsets of 4 simulations from 100 simulations. All of the runs reached the absolute minimum 0.2564 after at most 4 generations. There is a trade-off in computational effort between the number of generations and the population of subsets per generation.

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 6 presents the histograms of all 2 million values of D(J,q) (left) and of the lowest 5000 values (right).

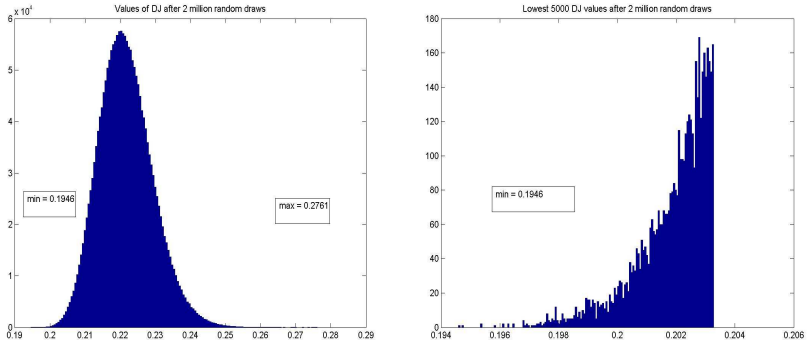

Figure 6: Histograms of all 2 million values of the metric D(J,q) obtained by the random search procedure (left) and of the lowest 5000 of the values (right).

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 7 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. Figures 8 (a), (b) and (c) show the histograms of the values of D(J,q) for the 2^{nd}, 25^{th} and 49^{th} generations. The long tail to the right in Figs 8 (b) and (c) is caused by newly subsets drawn at random each generation but except for these, the values of the metric decrease steadily as a function of the number of generations.

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.

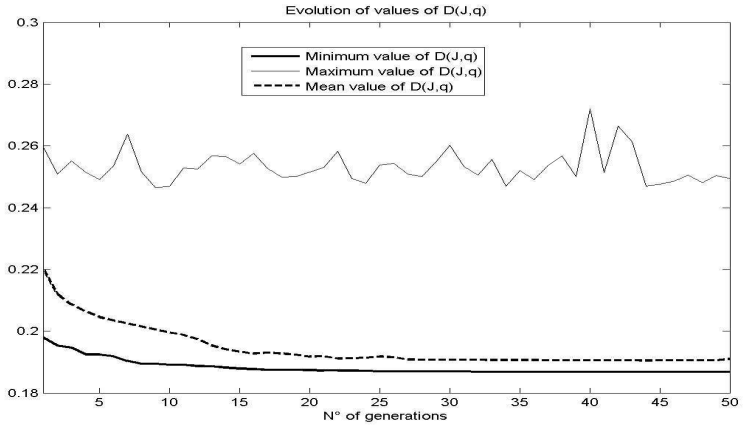

Figure 6: The evolution of the minimum value of D(J,q) obtained by the genetic algorithm (solid line), and also the mean (dotted line) and the maximum (thin line). Each generation consisted of 10,000 subsets. As expected, the minimum decreases monotonically as a function of the number of generations. The mean also decreases steadily but the maximum is more or less constant. This is because new randomly drawn subsets are being included.

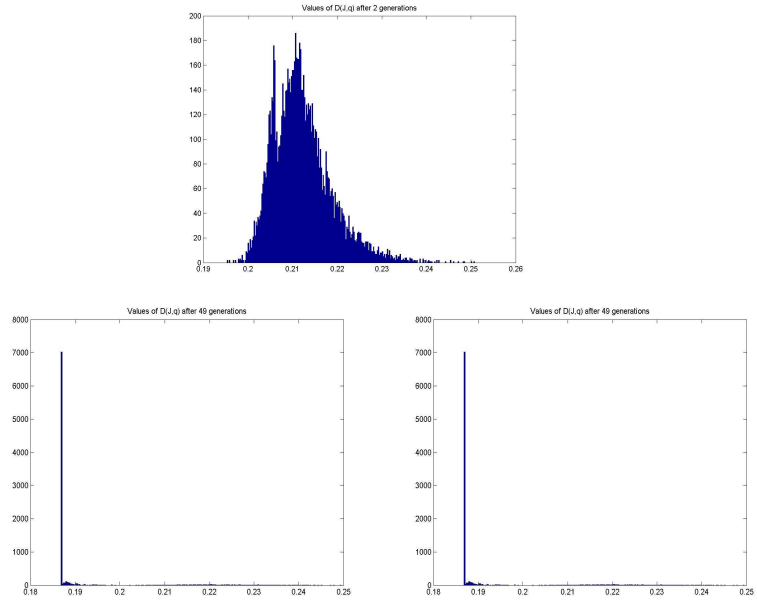

Figure 7: Histograms of the values of the metric D(J,q) obtained by the genetic algorithm after the second generation, the 25th and the 49th. Note that most of the values decrease but there is a long tail of values due to new randomly drawn subsets.

Table 2: 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 N°32 (the "0" for the second parent indicates a mutant)

	D(J,q)	ID N°	Parent N°1	Parent N°2	N° 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

As both the genetic algorithm is an iterative procedure, there is no guarantee of their reaching the true minimum. In this example we found that 8009 of the 10,000 subsets in the 50^{th} 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 2 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 N° 32 as a result of a mutation. The "0" for the second parent indicates a mutation as opposed to a cross-over.

The fact that the simulation numbers in all 8009 subsets are identical suggests that the algorithm may have reached the true minimum, especially as it first found the combination in generation N° 32 and no improvement was found in the subsequent 18 generations.

Conclusions and perspectives for future work

The tests carried out confirm that the genetic algorithm outperforms the random search method used earlier. Having established this point, we now need to know more about its performance characteristics. Firstly what criterion should be used to stop the algorithm? A purely 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. 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. We also suspect that if too few individuals are used per generation, the algorithm may get stuck in local minima rather than finding the global one. Further work is required to clarify these points. Finally work is currently in progress on the convergence of the algorithm from a theoretical point of view.

Bibliography

[1] Armstrong, M., A.A. Ndiaye & A. Galli (2010) “Scenario Reduction in Mining” presented at the IAMG Annual Meeting in Budapest, 29 August – 2 Sept 2010.

[2] Armstrong, M., A.A. Ndiaye, R. Razantsimba & A. Galli (2012) "Scenario Reduction Applied to Geostatistical Simulations”, accepted for publication in Maths Geosciences.

[3] Heitsch & Romisch (2007) Scenario tree modelling for multistage stochastic programs, Mathematical Programming, Ser A DOI 10.1007/s10107-007-0197-2

[4] Holland J., (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor

[5] Cao Y.C. & Q.H. Wu (1999), "Teaching Genetic Algorithms Using Matlab” Int J. Elect Educ, Vol 36, pp 139-153.

[6] Isaaks, E.H. and R.M. Srivastava, (1989) An Introduction to Applied Geostatistics, Oxford University Press, New York, 561pp

Appendix: 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 cutoffs; in petroleum engineering it could be streamline fluid flow simulations. The next step is to construct the square NxN matrix of distances between pairs of proxies. This matrix is of course symmetric with zeros down the diagonal.

Metric D(J,q)

Let $\mathcal{P}$ be the probability measure associated with the initial set of N geostatistical simulations which are denoted by ζ^i. for i=1, … N. Let p_i be the probability of the i^{th} simulation. In general geostatistical simulations are considered to be equally probable so $p_i=1/N$. We want to find a probability measure $\mathcal{Q}$ with only k simulations, that is as close as possible to $\mathcal{P}$. 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 j^{th} 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 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 center 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).

Small example

The easiest way to illustrate these two steps is via a small 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 3 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 N°1 (in J) is simulation N° 12. In fact simulation N°12 is the closest one to 9 of the simulations in J (N° 1, 3, 6, 8, 10, 16, 17, 19 & 20). These are highlighted in yellow. So its new probability is (9+1)/20 = 0.5. Five of the simulations in J highlighted in blue are closest to N° 7 (N° 5, 9, 11, 14 & 18). So the new probability for N° 7 is (5+1)/20 = 0.3. One simulation (N° 4) is closest to N° 13, so its new probability is 0.1 and the remaining two simulations (N° 2 and 15) are singletons that are far from any other simulations. Table 4 gives the new probabilities.

Table 3: Five simulations N° 2, 7, 12, 13 & 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

	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

Table 4: The new probabilities q for the 5 simulations selected

Simulation N°	2	7	12	14	15
Probability q	0.05	0.30	0.50	0.10	0.05