Presented at the "European 3-D Reservoir Modelling Conference", Society of Petroleum Engineers, Stavanger (Norway), April 1996.

SPE 35518

Optimisation of Production Strategies using Stochastic Search Methods

T. J. Harding and N. J. Radcliffe, Quadstone Ltd.* and P. R. King, BP Exploration

This paper was prepared for presentation at the European 3-D Reservoir Modelling Conference held in Stavanger, Norway, 16–17 April 1996.

This paper was selected for presentation by the SPE Program Committee following review of information contained in an abstract submitted by the author(s). Contents of the paper, as presented, have not been reviewed by the Society of Petroleum Engineers and are subject to correction by the author(s). The material, as presented, does not necessarily reflect any position of the Society of Petroleum Engineers or its members. Papers presented at SPE meetings are subject to publication review by Editorial Committee of the Society of Petroleum Engineers. Permission to copy is restricted to an abstract of not more than 300 words. Illustrations may not be copied. The abstract should contain conspicuous acknowledgment of where and by whom the paper was presented. Write Librarian, SPE, P.O. 8333836, Richardson, TX 75083-3836 U.S.A. Fax 01-214-952-9435

Abstract

This paper describes the application of stochastic search techniques to the production scheduling of a group of linked oil and gas fields. The goal was the maximisation of total net present value and a genetic algorithm using problem-specific crossover operators was particularly successful in this respect.

Introduction

Reservoir engineers must decide upon a "best" management strategy for the exploitation of each petroleum resource, typically the strategy which will maximise economic return. In particular, a production schedule, specifying rates of extraction over the lifetime of the reservoir(s), must be chosen. From this, decisions about the construction of processing facilities and pipelines will follow. A quantitative approach to the search for the "best" production schedule requires the construction of a mathematical model, which must capture enough characteristics of the reservoir(s) to predict the costs and benefits of any given schedule.

Even when a single independent field is under consideration, finding the "best" schedule can be a non-trivial problem, as all but the simplest models are likely to be non-linear and contain discontinuities. BP has a number of multi-field potential developments world-wide. In such cases, the problem is more difficult still. The total number of production rates is of course proportionately larger, but extra complexity arises from the inter-dependence of the fields. There are extra degrees of freedom, namely the respective timing of each field, and in addition the costs associated with each field depend on the sum of the production rates from some or all of the other fields.

If possible, the reservoir engineer will also wish to ensure robustness of the choice of extraction strategy in the presence of uncertainty in the economic model—the most obvious source of uncertainty in this case being the reservoir characterisations.

This paper describes how several techniques—genetic algorithms in particular—were used to search the space of possible production schedules based on an indicative data set from one multi-field development. The techniques show potential ways to achieve significant increases in net present value, which were consistent with traditional discrete analysis.

The NPV model

The NPV model in question is a non-linear function of a set of real variables $\{x_{i,t}\}$. Each $x_{i,t}$ represents the target annual average production rate for the primary stream of field i in year t. (Depending on the field, the primary stream is either oil or gas). If uncertainty is neglected, the model assumes these rates to be the actual achieved rates. (If, however, uncertainty is taken into account, the actual rates will in general differ from the target rates.) The model has n_F fields, and the accounting period begins in year 1. Years 1 to 3 contain only capital outlay and no production, after which each field has a maximum productive life of n_Y years. The total number of independent variables $x_{i,t}$ is therefore $n_F \times n_Y$, which in the particular version of the model used for this study, was over 200.

The objective function When performing an optimisation the *objective function*, when presented with a solution, assigns to it a numerical value which reflects its quality. In the present case, the objective function comprises the model which predicts the NPV of the group of fields, on the basis of the production schedule described by the input variables $\{x_{i,t}\}$.

First consider the NPV for one field in the group. If r is the discount rate,

*TJH and NJR were at Edinburgh Parallel Computing Centre, University of Edinburgh, while this work was carried out.

$$\begin{aligned}\text{NPV} \quad = \quad & \sum_{t=1}^{n_Y+3} \frac{1}{(1+r)^{t-1}} \{ \text{ revenue}_t \\ & -\text{drilling-costs}_t - \text{capital-costs}_t \\ & -\text{operating-costs}_t - \text{fixed-costs}_t \ \}. \end{aligned} \tag{1}$$

The revenue is the sum of the three stream revenues—for oil, gas and natural gas liquids. The computation of the stream profile is the most compute-intensive part of the model—the annual production rate for each *non*-primary stream is computed from the primary stream rates $\{x_{i,t}\}$, and because this relationship is non-linear, the revenue is non-linear in the $\{x_{i,t}\}$. The drilling costs depend on the number of wells required to supply the target rate, and are therefore a function of the maximum primary stream rate over the course of extraction. The model includes a swing factor on these maximal rates.

Thus far, each field is independent of other fields. However, the fields in the group are linked, and facilities at those fields which act as nodes must handle the production from client fields (as well as from the field itself). The capital and operating costs for a field i are proportional to the maximum rate which the field will have to handle, and accordingly they contain terms of the form:

$$\max_t \{ \sum_{j \in \{\text{fields serviced by } i\}} x_{j,t} \}.$$

For the purpose of calculating these costs, an extra conceptual field is introduced to represent the onshore facilities serving all the fields. The spend on the drilling and capital costs occurs in the 3 years prior to the start of production for each field. The operating costs for each field apply from the start of production (the start year) until abandonment (the end year). The start and end years are defined in terms of non-zero $x_{i,t}$, which makes the costs discontinuous in the $\{x_{i,t}\}$.

The constraints For simplicity, each field is modelled as a simple "tank"—its behaviour is assumed to be governed by just two parameters. These are the *deliverability*—the relative "ease" by which the primary stream can be extracted—and the *reserve*, the total amount of the primary stream hydrocarbon in the field. These parameters were estimated from reservoir simulations. The $\{x_{i,t}\}$ obey two types of constraint:

Productivity constraints. The "tank" model constrains production for a given field in any year to be less than (or equal to) a certain constant fraction of the remaining reserve. (It follows that if extraction were maximised, the schedule for each field would follow an exponential decay curve.)

Sequencing constraints. Each field must start operation before (or at the same time as) any fields which it serves. (The corresponding end year constraints are not treated explicitly, rather the objective function "repairs" violations as part of evaluation.)

"Good" schedules Whether for a single field or a group of fields, "good" production schedules—those with a high NPV—tend to follow an overall "plateau-with-tail" shape. Extraction is limited initially by the cost of building facilities to handle the flow (the plateau phase), and then by the productivity constraints (the tail phase), until abandonment. A typical schedule of this form can be seen in figure 5. When choosing a production schedule, the most basic feature to consider is therefore the total primary stream plateau rate—a high plateau provides early revenue, but a lower plateau reduces costs. However, within this general pattern, the NPV is still quite strongly dependent on the "phasing" of the start years and the respective field abandonment dates (particularly where delayed effects, such as gas re-injection, apply).

Optimisation of NPV with search techniques

Stochastic search and optimisation The ideal goal of optimisation is to find the solution that has the highest value of the objective function—a *global optimum*—in the present case, a production schedule that maximises the NPV. However, unless the form of the objective function is particularly simple, or the search space (the set of all valid production schedules, in this case) is very small, it is generally impossible to construct optimisation techniques that are guaranteed to find a global optimum. For this reason, most optimisation problems are tackled using *search* techniques, which explore the search space but do not guarantee to find a global optimum. A *local* optimum is a solution which is better than (or at least as good as) every solution in its "neighbourhood".

Search techniques can be considered as automated and rationalised versions of "what-if" analysis, in which a numerical model is run repeatedly while the values of the underlying variables are modified. Most search techniques begin by choosing one point (solution) from the search space as a starting point —the choice may be random or informed. Thereafter, the process typically involves maintaining a notion of a single *current point.* The search normally proceeds by repeatedly generating a trial *move* within the search space, by modifying the current point through the application of a *move operator*. The new point is then evaluated and a decision is made about whether to replace the current point with the new point. The "move" operator used may be either *deterministic* or *stochastic.* In the simplest case, the "move" to the new point is made only if it is better than (or as good as) the current point—such techniques are known generically as *hill-climbers*.

In real-world optimisation, one often settles for the best local optimum that one can find. Good search techniques normally produce near-optimal solutions relatively quickly. The process of search can be enlightening in itself—on real-world problems, stochastic search techniques tend to produce an "ensemble" of good solutions rather than a single "best", and this often gives some understanding of the variety of good solutions available.

Optimisation of NPV by simulated annealing Simulated annealing (Kirkpatrick *et al.*) is a technique commonly used for high-

dimensional, possibly discontinuous, optimisation problems. Simulated annealing takes its inspiration from a cooling crystal, in which the molecules align themselves provided that the cooling is sufficiently slow. In this case E, the total energy of the crystal, is the objective function, minimised by the formation of a perfect crystal.

A classical simulated annealing algorithm follows the pattern of a stochastic search described in the previous section. If a move improves the objective function, it is accepted and replaces the current solution. If the move worsens the objective function by $\Delta E > 0$, it is accepted with probability $e^{-\frac{\Delta E}{T}}$ where T is a parameter called "temperature".

The temperature T is set to a high value at the start of a run and slowly reduced during its course according to some schedule (hence "annealing"). The idea is that the system has the freedom to explore coarse features (such as the basins of different *local* optima) early on, but later becomes something closer to a classic "hill-climbing" algorithm to refine the search. As with any optimisation problem, the best solution encountered so far is always separately recorded.

Location of a global optimum is *statistically* guaranteed, provided that the temperature reductions are "sufficiently small" and that at each temperature, the number of moves is "sufficiently large" for the system to reach "equilibrium", though this is rather hard to measure in practice and may take a (very) long time. Selecting the proper choice of annealing schedule is a large, active field of research. In practice, better solutions can often be obtained in finite time by not permitting equilibration (see the discussion of "quenching" etc. by Ingber).

For the production scheduling problem in hand, $E = -\text{NPV}$ and two move operators were used:

`MoveStartYear`*:* This produced a shift in the production schedule, for one randomly-selected field, by plus or minus one year (subject to sequencing constraints).

`MoveRate`*:* This resulted in a new choice of the primary stream rate, for one field and one year, uniformly chosen within a band centred around the current rate. The width of the band was proportional to a parameter $\alpha \in [0, 1]$, up to a maximum of the whole valid range.

Optimisation of NPV by sequential quadratic programming Sequential quadratic programming (used here in the form of NAG routine `E04UCF`) is a deterministic optimisation technique based on calculus. It is essentially a "hill-climber", moving at each step in the direction that it believes to be the most direct route "uphill", subject to the constraints (which may be non-linear). It therefore generally finds a local optimum, usually "near" to the starting-point. It can, however, have difficulty if there are discontinuities in the objective function.

Optimisation of NPV by hybrid simulated annealing techniques Users of stochastic search techniques find that hybrid algorithms—combining a stochastic search with other, more problem-specific, optimisation techniques—often out-perform pure stochastic search techniques. In a variation of the simulated annealing algorithm described above, several "major iterations" were performed, each of which comprised five independent simulated annealing runs. The best schedule found in each major iteration was optimised with sequential quadratic programming and the result was used as the starting solution for the next major iteration.

Optimisation of NPV by genetic algorithms

Genetic algorithms *Genetic algorithms* (and, more generally, *evolutionary* algorithms) are a class of stochastic search techniques inspired by natural evolution (Holland). Crudely, this inspiration is that, over time, organisms adapt to their environment, becoming *fitter.* The aim of genetic algorithms is to simulate key aspects of natural evolving systems, and so harness their problem-solving ability for search and optimisation problems.

Genetic algorithms differ from the search techniques described previously in a number of important ways. First, instead of using a single current point, genetic algorithms maintain a *population* of points from the search space. The second key difference concerns the choice of the next point to test. Instead of basing this choice on the current solution, genetic algorithms typically use two *parent* solutions. Thus, in addition to a *unary* move operator (see below), genetic algorithms employ *binary* move operators, known as *recombination* or *crossover* operators, which take a pair of parent solutions and combine them in some way to produce a new *child* solution. The idea is to produce a child which "inherits" some of its properties from each parent. Several ways in which two production schedules can be crossed to produce a child schedule will be described below—for various reasons, chiefly optimisation performance and the constraints, the crossover is not quite as simple as taking some production rates from one parent schedule and other production rates from the other.

"Survival of the fittest" is modelled by the process of *selection*. There are two opportunities to apply selection pressure in genetic algorithms. The first is in the choice of parents, which may be biased towards better ("fitter") members of the population. Secondly, when children are produced, the choice of which members of the population they should replace may be biased toward the poorer solutions. While it is not particularly important where in the algorithm selection pressure is applied, the level of selection pressure is important. If it is too high, the search will tend to get quickly stuck in local optima. If it is too low, the search may be unduly slow, or may even fail to generate solutions of improving quality.

Typically, a conventional unary move operator (called a *mutation* operator in this context) is applied to the child solution before it is evaluated. Mutation is one of the techniques used to

maintain diversity in the population and thereby prevent *premature convergence* to an unacceptably poor solution. Gaussian creep mutation, which potentially adds Gaussian noise to each variable (Davis; Baeck *et al.*), was used in this case.

A simple genetic algorithm is illustrated in figure 1 and a comprehensive description of techniques used in evolutionary computing can be found in Surry & Radcliffe.

Representations and operators in genetic algorithms Ultimately, the success of any search method depends on the order in which it generates points in the search space. The basic requirement in the case of a genetic algorithm can roughly be stated as "the effect of crossing two good solutions should often be to produce another good solution". This requirement is of course *problem-specific*—it depends on the nature of the objective function. In most cases, salient features of the problem are known and move operators that incorporate this knowledge are very likely to outperform those that do not (Davis).

It is most important, when designing a genetic operator, to have regard to the moves that it generates in the *search space*. However, in practice, it is usually convenient to define an operator in terms of the effect it has on the internal data structures used to *represent* the solutions. The simplest representation tried for this problem was a two-dimensional, *real* array of size $n_F \times n_Y$, which directly represented the target production rates $\{x_{i,t}\}$. The crossover operator employed with this representation was "rectangular crossover with blend-$\frac{1}{2}$ at boundaries" (RECT-BLX-$\frac{1}{2}$), shown in figure 2. Before evaluation of the child schedule, a repair operator was applied to ensure the child satisfied the constraints (this is discussed in the next section).

Blend crossover (BLX) is often used in evolutionary algorithms for real-parameter optimisation. Given a single pair of real variables $x \leq y$ to recombine, BLX-$\frac{1}{2}$ makes a uniform choice for its outcome in the range $[x - \delta/2, y + \delta/2]$ where $\delta = y - x$. Values are then truncated to fall into the valid range. The use of BLX at the boundaries in RECT-BLX-$\frac{1}{2}$ supplements mutation as a means of reaching different real variable values from those present in the initial population.

More problem-specific representations were also tried, incorporating explicit start and end years in one case, and annual production totals in another. These required more sophisticated move operators (generalisations of RECT-BLX-$\frac{1}{2}$), but led to significantly faster or better optimisation, as shown in figure 4.

Handling constraints In many cases (including the present case) simple syntactic move operators are inadequate because of constraints on valid solutions, which will tend to be violated by merely cutting and splicing existing solutions. There are three principal methods for handling constraints in genetic algorithms—building more sophisticated operators that "understand" (ensure compliance with) the constraints (Radcliffe & Surry), using repair operators to "correct" infeasible child solutions, and employing penalty functions. Of these, the last is the crudest, and is normally only adopted when neither of the first two methods can be used. For this problem, the repair technique was used.

When repair operators are employed, an interesting decision arises as to whether to place the infeasible child in the population (using the repaired version of the child only for the purposes of computing the objective function), or to place the repaired child in the population instead. While, on the face of it, it appears more sensible always to place the repaired version of the child in the population, the genetic algorithm may have difficulty exploring the whole solution space as infeasible regions can provide a barrier to the search. There is empirical evidence (Davis & Orvosh) that adopting a probabilistic approach to saving repaired solutions is often a good strategy. This is borne out by the results shown in figure 4.

Other genetic algorithm parameters Many variations of a genetic algorithm are possible. As well as the choice of operators described in previous sections, the authors investigated the effect of the following parameters on this problem.

Generation of the initial population. A "maximally random" strategy performed best, rather than, for example, seeding the population with schedules of the "plateau-with-tail" form.

Population size. In general, larger populations find better solutions but, of course, computational cost scales linearly with population size. In many real-world problems, such as the one under discussion here, the available compute time is a strong constraint and the population size has to be chosen to produce the best result in the time available. A size of 1024 was found to be a good compromise in this case.

Population structure. *Structured* populations introduce a "spatial" structure to "localise" the interaction between the solutions. In an *island* model, a separate genetic algorithm runs on each of a number of conceptual islands, and solutions occasionally migrate from one island to another. In a *fine-grained model*, each solution has a unique location on a grid and mating only occurs within local neighbourhoods (*demes*). The aim in any case is to encourage the development of different kinds of solution in different parts of the population ("speciation") and thereby avoid premature convergence and improve final performance. Structured population models also facilitate implementation on parallel or distributed computers for faster optimisation. On this problem, structured and unstructured models with appropriate parameters were found to perform equally well.

Selection. Binary tournament selection (with $p = 1$, Goldberg & Deb) was used. To select each parent, this strategy picks two members at random from the population and then chooses the better of the two.

Replacement. The best strategy found for this problem involved a "degree of elitism", in which a sufficiently good solution might survive indefinitely in the population.

Optimisation of NPV with a "memetic" algorithm A *memetic* algorithm is a hybrid genetic algorithm, which incorporates some degree of local optimisation of each new solution before evaluation. If the local optimisation is guaranteed to find a local optimum, then the memetic algorithm effectively searches the space of local optima rather than the whole solution space. For this project, sequential quadratic programming was used for the local optimisation. Although the high cost of local optimisation meant that the algorithm was only possible with a small population and a small number of generations, it produced the best schedule of all (figure 5).

Results of NPV optimisation

Because the evaluation of NPV was the most significant computational cost, the performance of each algorithm was measured by the best NPV it obtained after a fixed number of evaluations of the objective function (in this case one million, representing an hour on a fast workstation). A secondary measure was the speed at which the algorithm attained "good" solutions, if this occurred notably sooner than a million evaluations. Several runs were performed for each choice of parameters, to assess the variability of the result.

NPV values, oil and gas rates are normalised relative to a reference schedule. The results are presented in order of increasing NPV.

Results from simulated annealing Out of many choices of parameters, the best involved an exponential cooling schedule of ten steps, from $T = T_0 = 10^{-3}$ to $T = 3 \times 10^{-4}$ (relative to normalised NPV) with characteristic move size $\alpha = T/T_0$, and `MoveStartYear` used instead of `MoveRate` precisely every 500th move. The best schedule obtained with simulated annealing is shown in figure 3, with NPV=1.326. The starting point in this case was a *big-bang* schedule (with NPV=0.051), in which all fields operate at maximum achievable rates over the whole period of n_Y years. At least with the particular move operators used in this project, simulated annealing appears to have difficulty in reducing rates to zero to remove the long "tail" on the schedule.

Results from sequential quadratic programming Unsurprisingly, the choice of starting point was very important here. A hundred runs from *random* starting points (a total of about 1.1 million evaluations) produced a best NPV of only 1.451. The best schedule obtained through sequential quadratic programming started from the "big-bang" schedule (NPV=0.051), and about 10,000 evaluations were required to achieve an NPV of 1.525. In comparison with the best schedule produced by simulated annealing (figure 3, NPV=1.326), this schedule had a slightly higher and longer (four-year) oil "plateau", less phasing (almost all the fields started immediately), a smoother "tail", and production was abandoned much earlier (year 23), providing an increase in revenue of 0.061 *and* a decrease in cost of 0.138 (both discounted).

Results from hybrid simulated annealing with sequential quadratic programming The best schedule obtained by this method had an NPV of 1.544. The algorithm required around one million evaluations per run (with little variability in the final NPV between runs). The original starting point was the best schedule produced by sequential quadratic programming (NPV=1.525), from which the new schedule differed by extending the operating life of about half of the fields, reducing the operating life of one field slightly, smoothing the profile of one field, and reducing the plateau rate for another, resulting in a further 0.017 extra revenue *and* a further cost saving of 0.002 (both discounted).

Results from genetic algorithms Apart from population size, the two features which had the strongest effect on the algorithm's performance were the choice of representation and the strategy for repair This is illustrated in figure 4.

The best schedule obtained by a pure genetic algorithm had an NPV of 1.553. This schedule combined good features of the best schedule produced by simulated annealing (figure 3, NPV=1.326)—namely, a similar degree of phasing and similarly low plateau rate—with those of the best schedule produced by sequential quadratic programming (NPV=1.525)—a four-year oil plateau and early abandonment (in this case, year 20). In comparison to the latter schedule, this schedule produced 0.033 less revenue, but provided a cost saving of 0.060 (both discounted).

Results from a memetic algorithm The best result produced by the memetic algorithm is shown in figure 5, with NPV=1.577. It was also the best schedule found overall. It required about eight million evaluations (with little variability in the final NPV between runs)—after the standard one million runs, the NPV was typically only 80–85% of this figure. The schedule is broadly similar to the best schedule produced by a pure genetic algorithm (NPV=1.553), but with an even lower and longer (five-year) oil plateau and with phasing slightly more like that of the best schedule produced by sequential quadratic programming (NPV=1.525). In comparison to the latter, this schedule produces 0.074 less revenue, but provides a cost saving of 0.126 (both discounted).

Optimisation of NPV with uncertainty

The most significant source of uncertainty in this problem is the characterisation of the reservoirs. In the context of the "tank model", this manifests itself as uncertainty in the deliverabilities and reserves, which is translated into uncertainty in the actual rates achieved for a given target schedule. To model this, during each evaluation, the deliverability and reserve were sampled from statistical distributions previously fitted to data from "off-line" reservoir simulations. The aim was to optimise the *expected value* of the NPV, which was estimated from the sample mean of repeated evaluations of NPV.

Genetic algorithms, by nature, cope more robustly than many other methods with the optimisation of a "noisy" objective function. Choice of sample size is a critical trade-off—it must be high

enough to permit the algorithm to distinguish true variation in fitness from noise, but not so large that the low number of achievable evaluations limits the evolution process.

Despite the NPV being quite noisy (with a standard deviation typically around 0.3), we have found that genetic algorithms are able to optimise successfully. The schedules with the best "deterministic" NPV have proved remarkably robust when uncertainty was introduced—although there is a small, statistically significant, trade-off between maximising the expected NPV and maximising the "deterministic" NPV. The results of this work will be the subject of a future paper.

It is worth remarking here that appropriate genetic algorithms are also able to perform multi-criterion optimisation. They could be used, for example, to explore the trade-off between maximisation of NPV and minimisation of one or more measures of risk or exposure.

Conclusions

A genetic algorithm, from a random start, out-performed both simulated annealing and sequential quadratic programming (and even a hybrid of the two) using a few tens of CPU hours. The overall effort required to do this was no greater than that of the other techniques. An intelligent choice of genetic operators and representation, guided by knowledge of the problem, provided apparently significant performance improvements. Further gains were achieved through the use of a hybrid genetic algorithm with local optimisation (a "memetic" algorithm). The gain in NPV from the use of genetic algorithms represented a few per-cent over other techniques. As always, it must be borne in mind that the model is an abstraction from reality and confidence in the result of the optimisation can only be as high as confidence in the model incorporated into the objective function. Nevertheless, for problems such as this, where the value is in billions of dollars, small improvements may represent significant benefits.

The reservoir engineer is typically interested not just in a "best" solution, but in understanding which features characterise good solutions, and in mapping out the extent to which good alternatives to the "best" solution exist. A stochastic search technique naturally produces an ensemble of good solutions rather than a single "best". Whilst all the good schedules found were reassuringly similar in many features—not least the NPV value—there were interesting qualitative differences between the schedules produced by different optimisation techniques. In particular, genetic algorithms (with the operators employed here) appeared better than calculus-based techniques at searching over phasing configurations, and also better than simulated annealing at optimising the plateau and tail rates.

Appendix: RPL2

The software environment used for all the techniques in this project was the Reproductive Plan Language RPL2 (Surry & Radcliffe). RPL2 comprises an interpreter, run-time system and a set of operator libraries. To this the user typically adds his or her own library of operators (written in C or Fortran) which bind to RPL2 instructions. The subsequent development cycle for search algorithms is therefore comparatively short. Representations can be user-specified if required. RPL2 runs on a variety of platforms (including a number of parallel and distributed systems—the language supports parallel execution transparently) and is available from Quadstone Ltd.

References

Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo), 1991.

Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold (New York), 1991.

Lawrence Davis and David Orvosh. Shall we repair? Genetic algorithms, combinatorial optimization, and feasibility constraints. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann (San Mateo), 1993.

D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, 1990.

John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press (Ann Arbor), 1975.

Lester Ingber. Simulated Annealing: Practice versus Theory. *Math. Comput. Modelling*, 18(11):29–57, 1993.

S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by Simulated Annealing. Technical report, IBM Thomas J. Watson Research Center, 1982.

Nicholas J. Radcliffe and Patrick D. Surry. Fitness variance of formae and performance prediction. Technical report, To appear in Foundations of Genetic Algorithms 3, 1994.

Patrick D. Surry and Nicholas J. Radcliffe. *The Reproductive Plan Language RPL2*. Quadstone Ltd., 1995.

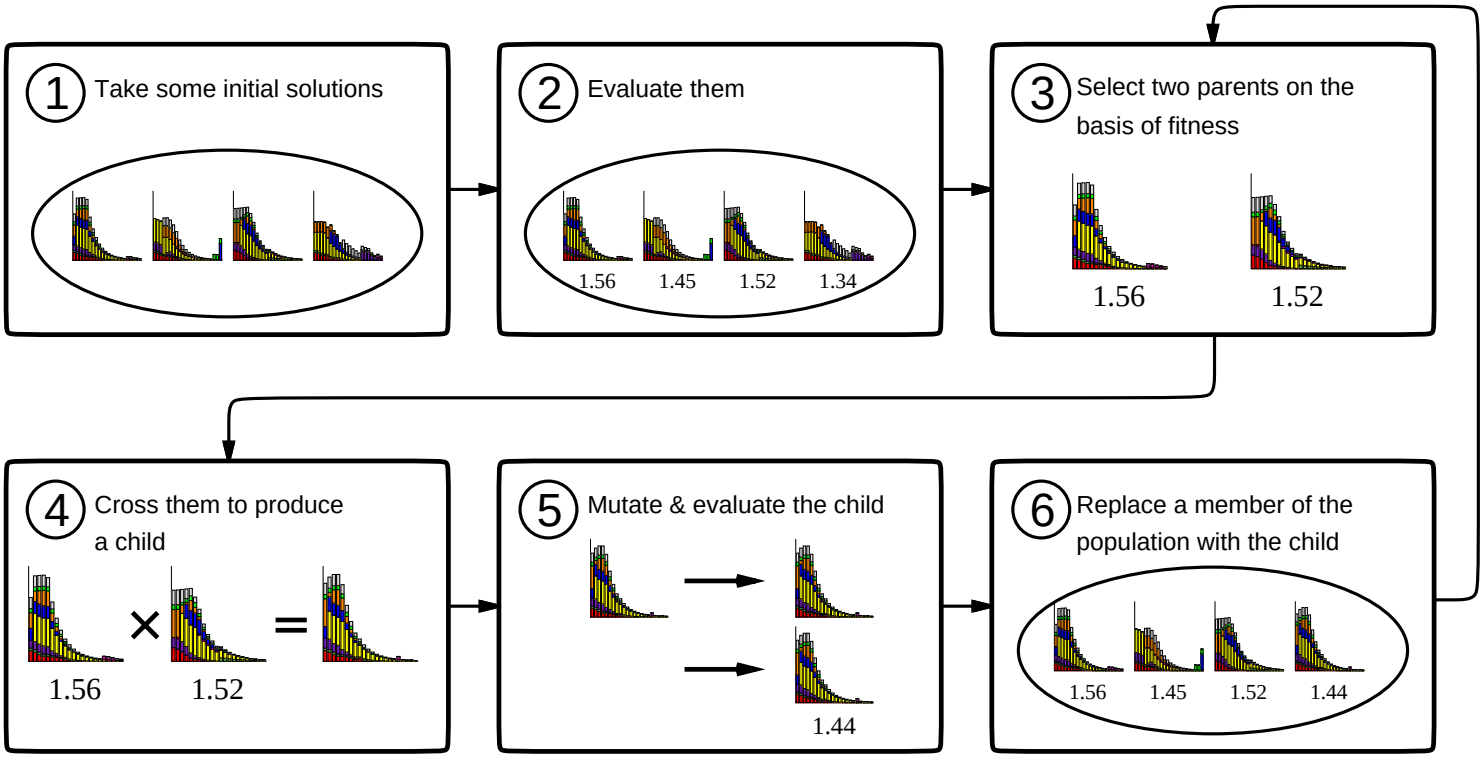

Figure 1: A simple genetic algorithm for evolving good production schedules. The bar-charts represent production schedules and the numbers represent the associated NPV predicted by the model (in normalised units).

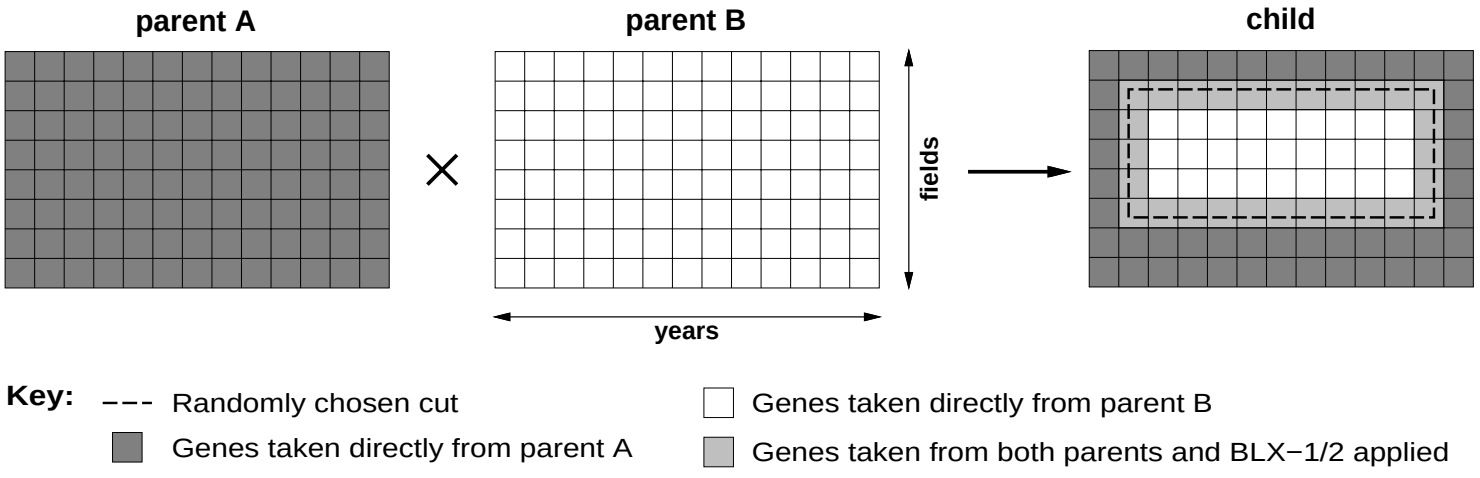

Figure 2: The RECT-BLX-$\frac{1}{2}$ crossover operator.

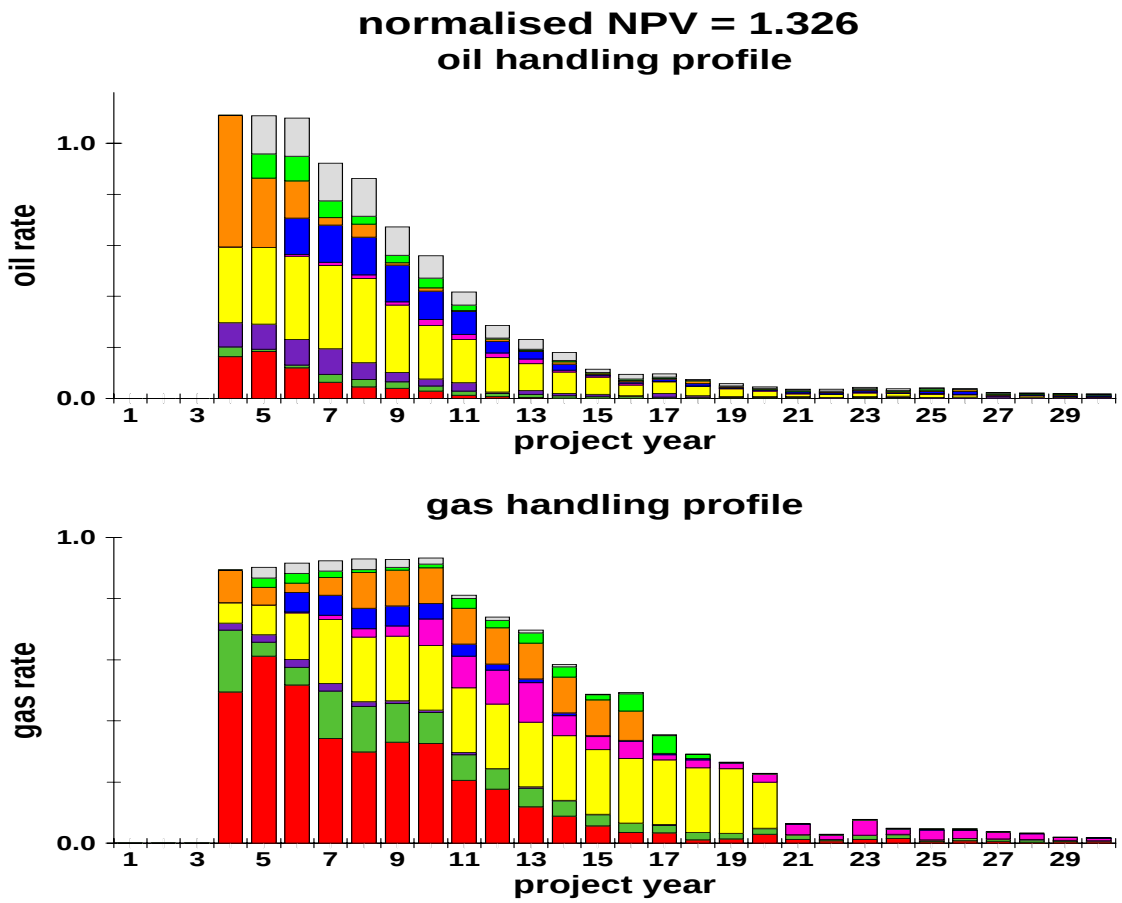

Figure 3: The best schedule obtained with simulated annealing. The bar-charts show the annual oil and gas production rates for each field in each year. The long "tail" makes operating costs high.

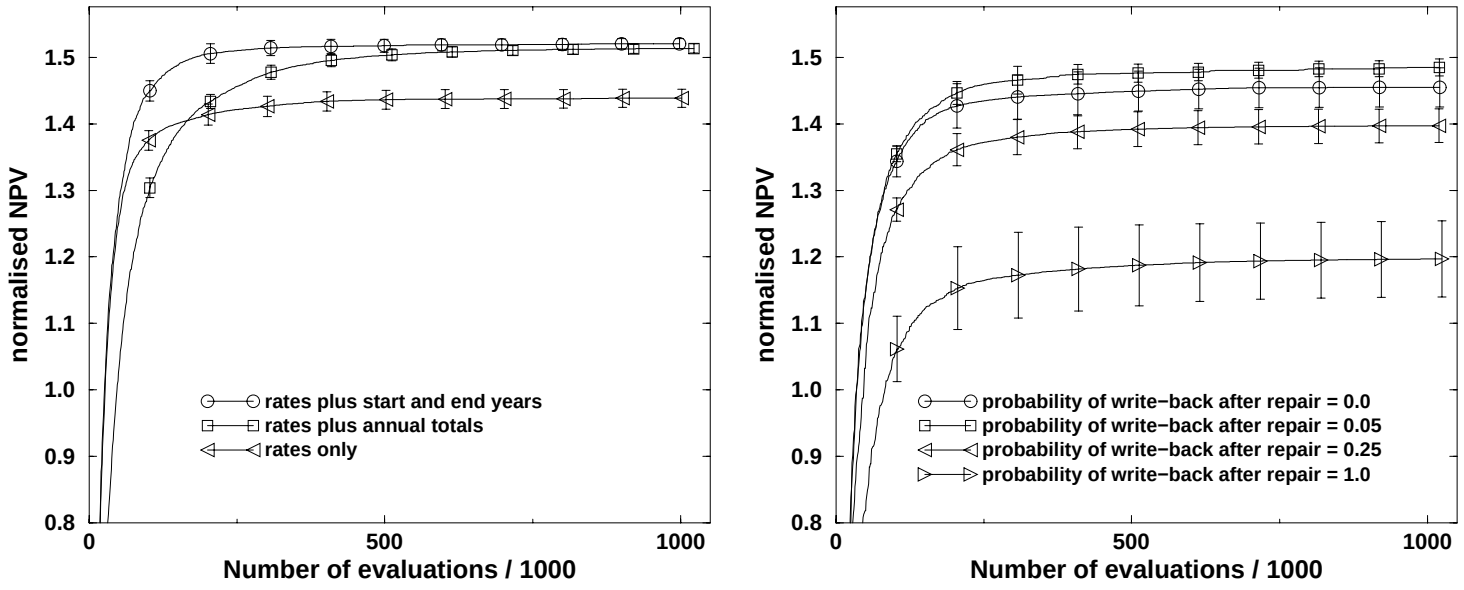

Figure 4: Two factors which had a significant effect on optimisation performance were the choice of representation, and the probability of placing the repaired version of each child into the population.

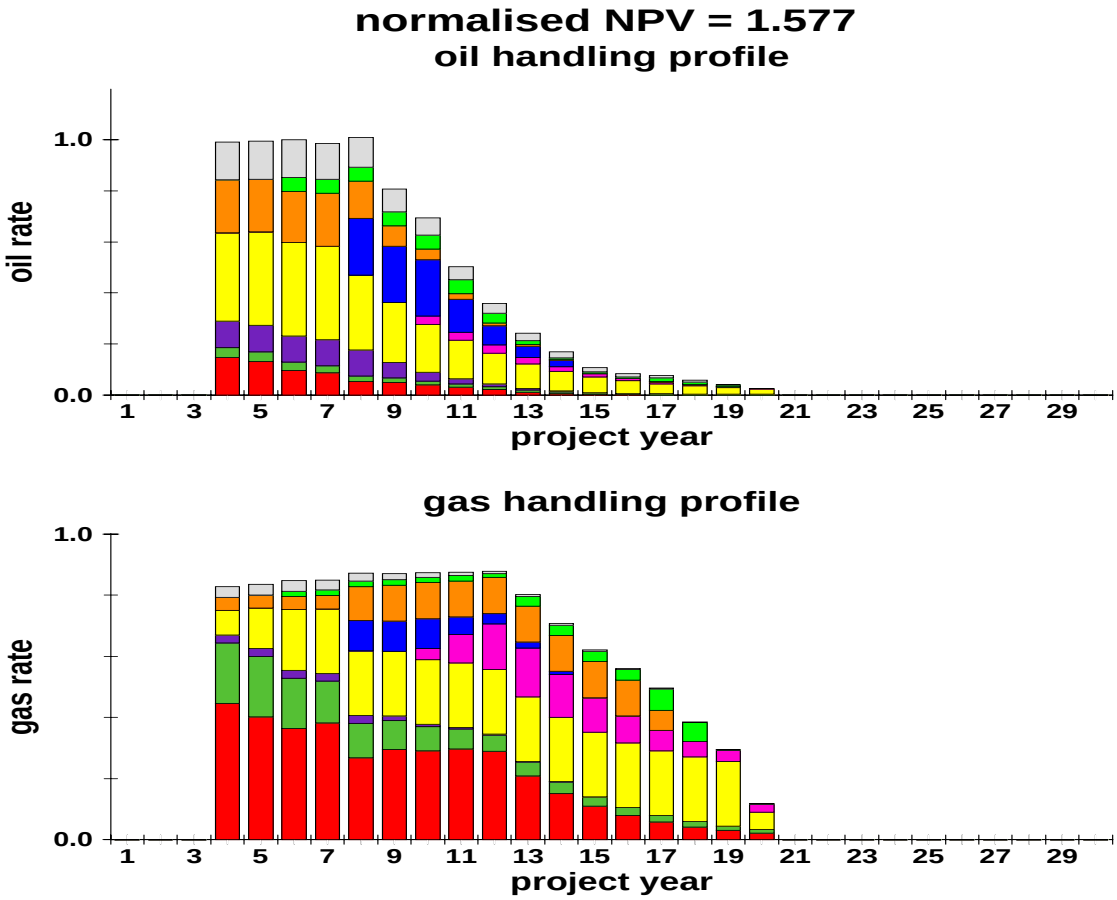

Figure 5: The best result produced by the memetic algorithm.