Maximising overall value in plant design

S. Huband a, D. Tuppurainen c, L. While b,*, L. Barone b, P. Hingston a, R. Bearman c

a School of Computer and Information Science, Edith Cowan University, Mt. Lawley, WA, Australia
b School of Computer Science & Software Engineering, The University of Western Australia, Nedlands, WA 6009, Australia
c Rio Tinto OTX, GPO Box A42, Perth, WA, Australia

Received 27 April 2006; accepted 23 July 2006
Available online 18 September 2006

Abstract

Existing plant designs are often conservative and as a consequence the opportunity to achieve full value is lost. Even for well-designed plants, the usage and profitability of mineral processing circuits can change over time, due to a variety of factors from geological variation through processing characteristics to changing market forces. Consequently, plant designs often require optimisation in relation to numerous variables, or objectives. To facilitate this task, a multi-objective evolutionary algorithm has been developed to optimise existing plants against multiple competing process drivers, as evaluated by simulation. A case study involving primary through to quaternary crushing is presented, in which the evolutionary algorithm explores a selection of flowsheet configurations, in addition to local machine setting optimisations. Results suggest that significant improvements can be achieved over the existing design, promising substantial financial benefits. An extension of the evolutionary algorithm to employ wider flowsheet modifications is also discussed.

© 2006 Elsevier Ltd. All rights reserved.

Keywords: Comminution; Process optimisation; Artificial intelligence

1. Introduction

Optimising the value of a mining operation is not straightforward. There is significant uncertainty surrounding many elements in the process, from reserve definition through to the metallurgical and extraction characteristics of the ore. Given that the economics of the operation are the focus of the design exercise, the numerous variables are difficult to manage in a manner that allows potentially thousands of variations to be examined to provide the optimum value generation. Variables are inherent in all the major areas including mine life, mine design, cut-off-grade, mining schedule, ore variability, process plant flowsheet, throughput rates, size reduction per processing stage, optimal beneficiation size, grind size-recovery relationship, operating costs, capital depreciation, closure, and environmental issues.

To deal with the large design space and to seek optimum value creation, many techniques have been applied to assist in the total design process. In this study, we use an evolutionary algorithm to optimise the design of the comminution part of the process so as to provide an optimised value for the full operation. The use of an evolutionary algorithm highlights many issues associated with how and why designs are developed, the amount of information required to provide the required level of accuracy, and the strength of the approach in terms of validation of input data and the delivery of a holistic view of the design process.

The paper is structured as follows. Section 2 discusses previous applications of artificial intelligence techniques to mineral processing, and Section 3 discusses the evolutionary computation technique used here. Section 4 describes the problem used in this study, and Section 5 describes the details of the evolutionary algorithm used for the problem. Section 6 lists and discusses our experiments and results, and Section 7 concludes the paper.
2. Previous applications of artificial intelligence to minerals processing

Previous attempts to apply artificial intelligence techniques to problems in minerals processing have focused on two techniques: expert systems and evolutionary computation.

2.1. Expert systems

Expert systems (ESs) use logic, decision making, and knowledge processing. In their simplest form, they are no more than a decision tree. In their more complex mode they have been integrated with fuzzy logic, neural networks, and numerical models, thus providing better ways of dealing with vague data, the representation of complex data sets, and the representation of equipment performance.

- Gensym’s G2, G2 Diagnostic Assistants (GDA), and NeurOn-Line neural network software (McCaffery et al., 2001) have been used to provide reasoning, inference, and fuzzy logic functions. MinnovEX has used this toolkit to build expert systems that can handle unit operations, including SAG mills, AG mills, ball mills, flotation columns and cells, crushers, feeders, and pelletisers. They report that 15 such control systems have been installed in the mineral sector, as of 2004.
- OCS (Broussaud and Guyot, 1999) employs real-time process control software with embedded modules that add flexibility to the base ES, namely a fuzzy expert module, a soft sensor module with adaptive predictive models, an optimiser module, a neural network module, and a vision module.
- Bearman et al. (1990) used a simple forward-searching decision tree to design quarry plant flowsheets for the generation of various size-fraction products. This invariably led to process designs that would achieve the required throughput and product sizes, but iterative considerations such as recirculating load were merely factored into the overall capacity of the machines. As such the designs were simply a solution to a particular problem, not an optimised solution.
- Prince (1997) used an ES coupled to existing numerical models of equipment to build up a flowsheet unit-by-unit. The circuit was solved using the net profit generated as the goal. The design was judged to be complete when adding another unit gave no further increase in net profit. All decisions were controlled by a knowledge base built from experience and industry practice. The approach was capable of generating process designs to meet the input requirements, but the financial driver in conjunction with the limits imposed by the knowledge base led to designs that favoured features not necessarily desirable in real processes (for example numerous re-cycle streams and maximum size reduction).

2.2. Evolutionary computation

Evolutionary computation is the generic name given to a family of population-based optimisation techniques that have been inspired by Darwin’s theory of evolution by natural selection (Darwin, 1859).

- Venter et al. (1997) proposed an approach from the field of genetic algorithms, namely learning classifier systems. Intelligent objects bid against each other for a position in the circuit. The simple flowsheets examined in the study demonstrated the concept, but most of the intelligence was in the form of subjective heuristics or empirical data. The strength of the work was that it showed that circuits could be assembled with the GA approach, however full process optimisation of the assembled circuit remained elusive.
- While et al. (2004) used an evolution strategy to optimise the performance of a simple comminution circuit with a single cone crusher with a recirculating load. They allowed the algorithm to vary the shape of the crusher’s liners and various operating parameters, trying to simultaneously maximise the quality of the product and the capacity of the circuit. Relative to an existing design, they reported an improvement of up to 12% in the P80 of the product, or up to 224% in capacity, or simultaneous improvements of 7% in P80 and 192% in capacity.
- Huband et al. (2005) extended the above work to allow the system to make minor changes to the structure of the circuit, as well as choosing the types and settings of the individual units. Designs were assessed against two “real” objectives (the P80 of the product and the cost of the circuit), and also against a third “error” objective used to allow feasible and infeasible designs to co-exist in the population. Again relative to an existing optimised design, they reported simultaneous improvements in both real objectives, including designs that reduced P80 by around 50% at no increase in cost.

3. Background technology

The two principal concepts employed in this study are evolutionary algorithms and multi-objective optimisation.

3.1. Evolutionary algorithms

The fundamental idea in an evolutionary algorithm is to maintain a population of individuals, each of which represents a potential solution to the problem at hand. The key requirement is that it must be possible to evaluate each individual, to determine which individuals represent “good” solutions to the problem, and which ones represent “bad” solutions. The evaluation of an individual is known as its “fitness”. Individuals representing bad solutions are eliminated from the population. Individuals representing
good solutions survive in the population and are allowed to reproduce under mutation to generate new individuals which are “similar” to themselves: as in nature, the “children” inherit features from their “parents”. The tendency is for the population to accumulate and combine features of good solutions to produce a series of populations where the fitness of the best individuals continually improves.

This basically represents an undirected search through “design space”, with continual evaluation of individuals. A typical evolutionary algorithm has the following broad steps.

1. Create an initial population of individuals, and evaluate each one.
2. Create a population of child solutions by combining and mutating the individuals in the current population, and evaluate each one.
3. Select the best individuals from the combined parent and child populations.
4. Repeat Steps 2 and 3 until some termination criterion is satisfied.

The preferential survival and reproduction of good individuals means that features of good individuals will proliferate in the population, and that these features will subsequently be combined to produce superior individuals.

The evolutionary approach is best-suited to domains where solving the problem analytically is either impossible or impractically slow, and where the size of the domain precludes a comprehensive or exhaustive search. Evolutionary algorithms have been applied successfully to a wide range of problem classes, such as optimising geometric shapes (for example Klockgether and Schwefel, 1970), optimising network layouts (for example Emmerich et al., 2001), optimising operational settings for complex systems (for example While et al., 2004), optimising the scheduling of people and other resources (for example Bagchi, 1999), and many others. These problem classes include a large number of difficult engineering problems that have previously been soluble only by trial-and-error and/or engineering judgement.

3.2. Multi-objective optimisation

One particular area where evolutionary algorithms perform well is where individuals must be evaluated according to more than one objective: this is known as “multi-objective optimisation”. For example, a car might be judged independently by its safety and its fuel economy. Ideally, we would like a car that excels in both objectives, but this may not be possible: for example, extra safety features may incur more weight, which degrades fuel economy. Which objective is more important will depend on the eventual use to which the car is put. What is needed is a population of designs, each offering a different trade-off between the two objectives. End-users can then select from this population according to their needs.

The principal difficulty that this creates for an optimisation technique is in the ranking of solutions. Given individuals $x$ and $y$, if $x$ is better than $y$ in both objectives, then of course $x$ is superior to $y$. But often $x$ will be better than $y$ in one objective and worse in the other, making it hard for the algorithm to rank them. This problem is exacerbated as the number of objectives increases. So how do we choose the “best” individuals to survive in the population and to reproduce?

This problem is commonly solved using a procedure known as Pareto ranking (Goldberg, 1989). If $x$ is better than $y$ in all objectives, clearly $x$ is superior to $y$: $x$ is said to dominate $y$. Given a set of individuals $S$, if $x$ is not dominated by any member of $S$, $x$ is said to be non-dominated wrt $S$. The ranking scheme that we use in this case study works as follows.

1. Initialise $k$ to 0.
2. Determine the individuals in $S$ that are non-dominated wrt $S$ (call this set $S^*$).
3. Give the members of $S^*$ the rank $k$, remove them from $S$, and increment $k$.
4. Repeat Steps 2 and 3 until $S$ is empty.

Thus the best individuals have rank 0, and if $x$ dominates $y$, $x$ will have a lower rank than $y$.

Now if we have to select $n$ individuals from the population to survive, we select whole ranks starting from 0 until we reach a rank that would not fit into the $n$, then we use some splitting mechanism to choose part of the next rank to complete the new population. Typically such a mechanism tries to optimise the composition of the new population, perhaps by promoting a diverse range of trade-offs or a diverse set of underlying forms.

4. Case study problem

Fig. 1 shows the default comminution circuit used in this study.

Run of mine material is fed into a primary crusher, and the output is screened. Screen undersize goes directly into the fine ore bins, and the oversize is stacked in a coarse ore stockpile.

Reclaimed coarse ore is fed into a crushing plant consisting of an open circuit secondary crusher and a tertiary crusher in closed circuit with two single-deck screens. The screens are fed with the combined output of the secondary and tertiary crushers. Screen oversize is (re-)processed by the tertiary crusher and the undersize forms the crushing plant product and is taken to the fine ore bins.

The fine ore bins feed two parallel milling circuits and their combined product is treated in a leaching process. Each milling line has a selected target P80 value (the size of the 80th percentile in its output) which determines the capacity of that line. The milling lines have a pre-defined order in which they are utilised: material is fed into the
preferred line until that line reaches capacity, then excess material is fed into the subsequent line(s).

The evolutionary algorithm is allowed to consider several changes to this circuit.

- It can alter the types and settings of the primary and secondary screens.
- It can discard the tertiary circuit, i.e. the secondary screens and the tertiary crusher, leaving the secondary crusher to feed directly into the fine ore bins; or it can move the tertiary crusher into open circuit.
- It can replace each of the secondary and tertiary crushers with a high-pressure grinding roll (HPGR), and alter the types and settings of these units.
- It can change the feed order of the milling lines.
- It can alter the P80 value of any milling line, and it can add an HPGR to the start of any line.
- It can change the layout of the milling lines, either by discarding the rod mill or by converting it into a third milling line with a ball mill.
- It can alter the charge level of any mill, i.e. the proportion of the interior volume that is occupied by balls or rods as appropriate.

By exploring these options and combinations thereof, the algorithm should be able to derive a new design with improved performance. The fundamental design goal of the circuit is to maximise the profit of the operating
company. However, circuit performance depends on the prevailing operating conditions, which are typically unknown in advance: at the very least, knowledge of anticipated operating conditions is imperfect. As such, a circuit that can perform well in response to changing circumstances is desirable; i.e. some degree of risk management is required. This makes the problem well-suited to a multi-objective approach, which can return a population of designs offering a range of trade-offs between multiple objectives. In this study we use one objective that represents the profit of a circuit given the expected feed, and another objective that represents the profit given a harder feed that is more difficult to process. The extra objective enables us to evolve circuits that are more robust.

The economic performance of the overall operation is evaluated with a wide range of models. Resources are modelled using a relationship between a chosen cut-off grade for rejected ore, and the resulting grade and quantity of ore available to be processed. The performance and capacity of a flowsheet is determined by a steady state mass-balance using the unit models listed in Table 1. The mass-balance searches for the highest sustainable feed-rate that does not exceed either capacity constraints for individual units or utilisation constraints placed on the various plant sections. An estimate of daily operating costs is calculated based on electricity consumption, reagent use, maintenance requirements, and mining and handling costs. Net revenue from future metal sales is incorporated into a discounted cash flow calculation and offset against the investment costs due to any capital upgrades made to the flowsheet. This forms the net present value (NPV) of the operation, and is used as the figure of merit that represents the economic performance of a solution.

### 5. Design of the evolutionary algorithm

To apply an evolutionary algorithm to a problem such as the above, several aspects need to be specified.

- How are designs mutated? i.e. given one or more designs, how do we generate their “children”?
- How is the initial population of designs generated?
- When does the algorithm terminate its search?

The remainder of this section examines these issues, with particular reference to the problem under discussion.

#### 5.1. Representation of designs

The form of a solution can be divided into two parts: those parts that are the same for each design, and those parts that will vary between different designs. The representation of each individual in the population needs to capture the parts that vary between different designs. In our case study, the evolutionary algorithm can vary the following.

- For the primary screen:
  - the aperture of the top deck;
  - the aperture of the bottom deck;
  - its type, i.e. conventional inclined or multi-slope;
  - and its custom load factor.
- Whether the tertiary circuit is retained or discarded.
- Independently for the secondary and tertiary crushers:
  - their types, i.e. cone crusher or HPGR;
  - for each cone crusher, its closed-side setting, i.e. the gap between the liners at their closest approach; and for each HPGR, its type, i.e. big or small.
- Whether the tertiary crusher is in closed circuit or in open circuit.
- For the secondary screen-pair:
  - the aperture of each unit;
  - and the type of each unit, i.e. conventional inclined or multi-slope.
- The feed order of the milling lines.
- The placement and types of the mills:
  - either two milling lines with one ball mill each;
  - or three lines with one ball mill each;
  - or one line with one ball mill and another line with one rod mill followed by a ball mill (as shown in Fig. 1).
- Independently for each milling line:
  - its target P80;
  - and the presence or absence of a quaternary HPGR, and its type.
- Independently for each mill:
  - its charge level.

| Crushers | Whiten-Anderson model fitted to site-survey data. Models the relationship between closed-side setting, size reduction, and crusher capacity efficiency when the screen is overloaded. |
| Screens  | Standard calculation to predict nominal capacity based on screen deck area, aperture size, and machine type. Models the reduction in screen efficiency when the screen is overloaded. |
| Milling  | Conventional Bond work-index model relating product size, mill capacity, and power consumption. |
| HPGR     | Regression of reduction ratio and feed top-size based on a range of pilot scale tests with a similar ore. |
| Leaching | Predicts metal recovery using a first order kinetic equation. Predicts both recovery loss at higher feed-rates and enhanced recovery due to a finer mill product. |
No other design aspects can be varied by the evolutionary algorithm, and are thus common to all designs. For example, all designs must employ a primary crusher, all designs must employ a double-decked primary screen, the secondary screens are always single-decked, and so on. Conversely, the evolutionary algorithm can choose between conventional inclined or multi-sloped screens, just as it can choose to employ a third milling line, etc.

The motivation for permitting some design changes whilst restricting others is determined by the interplay of several factors. Foremost amongst these is the goal of allowing the evolutionary algorithm to consider the same upgrade options that an engineer confronted with the same problem would consider. However, where an engineer can consider only a few options at a time, the evolutionary algorithm is able to trial a large number of options, in many combinations.

The ability to trial many different combinations of options simultaneously motivates the inclusion of further variables for the evolutionary algorithm to manipulate. Such modifications would not normally be worth investigating due to the time required to do so for unclear returns on NPV. Neither of these is a good reason for excluding such options from the evolutionary algorithm, which may find unexpectedly “synergistic” upgrade combinations.

5.2. Evaluation of designs

Designs are evaluated to assess and compare their performance in solving the problem at hand. In our case study, designs are evaluated independently according to four objectives.

5.2.1. Maximise NPV when applied to “soft” material

How well the design performs when presented with the expected feed. That is, the expected economic return of the design assuming the expected feed prevails over the life of the mine.

5.2.2. Maximise NPV when applied to “hard” material

How well the design performs when presented with a “hard” feed. The hard feed is characterised by both a higher P80 and raised ore hardness values than the expected material.

This objective captures the expected economic return of the design assuming that a harder ore prevails over the life of the mine. It is not uncommon for mines to be designed with respect to an expected feed, only to discover that initial expectations were wrong. Such mistakes can be costly. By considering a trade-off between the expected ore NPV and a harder ore NPV, a more conservative design could be selected, one capable of adjusting well to a different ore.

Although not considered in this case study, additional objectives could be introduced to consider a wider range of ore hardness. Doing so would make sense for a green-field project, where greater uncertainties exist.

5.2.3. Minimise overflow in crusher feeds

A poor design might require a crushing machine to process material at a greater rate than its capacity. Excess material will then overflow the mouth of the crusher and be lost. Obviously we want to minimise the amount of overflowing material: in fact we will consider a design to be feasible only if there is no overflowing material at all.

This objective captures the total volume of overflowing material (generally due to crushing units exceeding their capacity) in the system. That is, the volume of ore that would require extraction from the system in order for mass balancing to successfully conclude.

5.2.4. Minimise oversize material in crusher feeds

A poor design might present material to a crusher that is larger in size than the maximum-sized particle it can accept. Again clearly the amount of this material is to be minimised, and again we will consider a design to be feasible only if there is no oversize material at all.

Similar to the previous objective, this objective sums the total amount of oversize material at each unit in the circuit after mass balance, for both the hard and soft ores. Thus, feasible solutions are capable of mass balancing without error for both soft and hard feeds.

5.3. Selection of designs

In order to select which designs survive into subsequent generations and get a chance to breed, we have to rank them in some way according to their fitness values. As noted in Section 5.2, some designs might perform so poorly in some respect that they would not be counted as successful solutions to the problem. Such designs are labelled as “infeasible” designs. However, such designs might have some features that could contribute to successful designs in subsequent generations, so we do not normally just discard them immediately. Many approaches are used to deal with infeasible designs: in this case study we address the issue of infeasibility through the selection mechanism.

The approach we use is to quantify the “degree of infeasibility” in a poor design and then to select against it for future generations. The quantification is through the third and fourth objectives listed in Section 5.2, and the ranking scheme used is as follows.

(1) Perform a Pareto ranking on the population, as described in Section 3.1.
(2) Break ties according to the fourth objective.
(3) Break any remaining ties according to the third objective.

The effect of this ranking scheme is to allow infeasible designs to survive in the population, if they have suffi-
ciently good NPV objectives, but to eliminate them in the long run (through Steps 2 and 3) as the number of feasible designs increases. Thus “reasonable” infeasible designs can survive as long as the population is not dominated too much by feasible designs.

5.4. Mutation of designs

When a design is selected to breed new designs, the intention is that the new design should be similar to the previous one, but obviously not identical. Thus each of the parameters in the design is varied slightly in the generation of the child: this is known as “mutation”. Also often multiple designs are selected to contribute to the specification of the child, each passing on some of its parameters in a process known as “crossover”.

The implementations of mutation and crossover used in this study are fairly standard, so we do not discuss them here. Details are available from the authors on request.

5.5. Initialisation

The individuals in the initial population for an evolutionary algorithm are usually derived in one of two ways: either randomly, if the goal is an open search of the entire design space; or from some known existing “good” individual, if the goal is to fine-tune this design.

In this study we derive the initial population randomly. This gives the evolutionary algorithm neither initial bias nor advantage, and in fact the quality of random solutions is typically very poor, if not infeasible.

5.6. Termination

A variety of termination objectives are possible:

• using as much time as is made available;
• or achieving a minimum performance standard;
• or detecting a tailing-off of improvement in designs;
• or others.

In this initial phase of the case study we take the simple approach of running the algorithm for a fixed number of generations. This is the most common choice in the literature.

6. Experiments, results, and discussion

We ran the algorithm with a population of 100 for 750 generations from a randomly seeded population. We performed 20 runs, each one taking approximately 12–18 h on a modern PC.

Fig. 2 shows attainment surfaces that plot the increase through time of the performance of the population from each run. A 50% attainment surface shows the level of performance attained by half of the runs after a specified number of generations. The 0% attainment surface shows the level of performance attained by the best designs derived at the end of the runs. The NPV objectives are normalised relative to the existing design, therefore the performance of the existing design is (1,1) by definition. We plot only feasible designs, so the two error objectives are 0 for all designs and need not be plotted.

Fig. 2. Increase in normalised performance through time from 20 runs of the evolutionary algorithm. The 50% surfaces represent the average performance from the runs; the 0% surface represents the best performance from the runs.
Fig. 2 shows a clear increase in performance through time as both NPV objectives are maximised, represented by the fact that later fronts on the graph are higher and more to the right than earlier fronts. The average of the runs beats the existing design at (1, 1) within around 50 generations, then continues to improve through the remaining generations.

Fig. 3. Average increase in hypervolume through time from 20 runs of the evolutionary algorithm.

| Table 2 |
|-----------------|----------------|-----------------|-----------------|-----------------|
| **Details of the best designs from 20 runs of the evolutionary algorithm, and their normalised NPVs** |
| Engineer A (existing) | Engineer B (upgrade) | Best soft NPV | Intermediate trade-off | Best hard NPV |
| Normalised soft NPV | 1.0000 | 1.0078 | 1.0287 | 1.0251 | 1.0176 |
| Normalised hard NPV | 1.0000 | 1.0128 | 1.0328 | 1.0467 | 1.0531 |

**Primary crushing circuit**
- Screen-ApertureTop (mm): 65.0
- Screen-ApertureBottom (mm): 19.0
- Screen-CustomLoadFactor (%): 100
- Screen-Configuration: CI

**Fine crushing circuit**
- SecondaryCrusherType: CC
- SecondaryConeCrusher-CSS (mm): 20.0
- UsingTertiaryCircuit: True
- ClosedCircuit: True
- TertiaryCrusherType: CC
- TertiaryConeCrusher-CSS (mm): 12.0
- Screen-Aperture (mm): 16.0
- Screen-Configuration: CI

**Milling circuit**
- RodMillStatus: Retained
- FeedOrder: 1, 2
- Line-1-MillingLine-RodMillCharge (%): 40.0
- Line-1-MillingLine-BallMillCharge (%): 45.0
- Line-1-MillingLine-TargetP80 (μm): 298
- Line-1-UsingHPGR: False
- Line-2-MillingLine-BallMillCharge (%): 36.0
- Line-2-MillingLine-TargetP80 (μm): 306
- Line-2-UsingHPGR: False
- Line-2-HPGR-Model: Small

CI denotes a conventional inclined screen, MS denotes a multi-slope screen, and CC denotes a cone crusher.
Fig. 3 plots the average hypervolume of the population from each run through time. Hypervolume is a technical measure of the quality of the population in an evolutionary algorithm (While et al., 2006): basically if one population has a larger hypervolume than another population, the former population is taken to be better.

Fig. 3 shows that the average hypervolume increases through time in the expected manner, showing that the populations improve over time. Fig. 3 also gives the hypervolume of the existing and upgraded designs for information, but note that as each is just a single individual, they are not strictly comparable to the average of a number of populations.

Table 2 gives details of selected designs produced by the 20 runs, both details of the designs evolved by the evolutionary algorithm, and their normalised performance figures. Remember that the algorithm returns a set of designs, so we have chosen three interesting designs to compare with the existing designs. We list the design with the highest soft NPV, and the design with the highest hard NPV. But both of these designs sacrifice one objective to maximise the other, so we also list a design that offers a significant intermediate improvement in both NPV objectives.

The algorithm has discovered a significant improvement over the current operating methodology. The parallel milling lines have a different trade-off between product size and milling capacity. There is a potential gain from reducing the product-size of the mill that would have the smallest loss of capacity due to the finer product-size, and increasing the product-size of the mill that would gain the most capacity due to the increase. This enabled the algorithm to find designs offering more than triple the NPV gain compared to the suggested Engineer B upgrade, without requiring additional investment.

There were also significant benefits derived from the process of constructing the overall model that were distinct from the actual optimised results. The evolutionary algorithm provides a robust methodology to test the overall quality and suitability of the model for the task at hand. A qualified person examining the optimised solution set can quickly find errors in the assumptions used for the modelling. If errors are present, then either the solutions will exploit the errors to achieve unrealistic results, or the solutions will not exploit a change that would be expected to be beneficial. This is a significant bonus for the designer, providing a degree of confidence that all the important aspects of the model have been tested with a wide range of data and conditions.

7. Conclusions and future work

We have described the application of an evolutionary algorithm to the problem of optimising the performance of a comminution plant, by fine-tuning the settings of individual machines, by tuning machines in the circuit to work well in tandem, and by making (minor) structural changes to the circuit. The performance of the plant is optimised independently for two different types of feed, and the algorithm returns a range of designs offering different trade-offs between the two objectives.

The results indicate that the algorithm is able to find designs that offer useful improvements in performance, particularly when applied to hard material, and that it is able to find designs that are better than the existing design in both objectives simultaneously. A fuller investigation promises significant financial benefits.

We plan to extend our evolutionary algorithm to explore wider flowsheet modifications in several ways, including:

- allowing it to upgrade cone crushers;
- allowing it to vary aspects of the leaching process;
- allowing it to introduce ore-sorting machines; and
- allowing it a greater range of structural variations, possibly via a library of “structural templates”.

Acknowledgement

This work was supported by Rio Tinto OTX and Australian Research Council Linkage Grant LP0347915.

References