Available online at www.sciencedirect.com
SCIENCE DIRECT®

Minerals Engineering 17 (2004) 1063–1074

MINERALS ENGINEERING

This article is also available online at:
www.elsevier.com/locate/mineng

A multi-objective evolutionary algorithm approach for crusher optimisation and flowsheet design

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

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

Received 1 April 2004; accepted 3 June 2004

Abstract

The performance of crushing equipment in mineral processing circuits is often critical to the generation of final product. A multi-objective evolutionary algorithm has been developed that allows the crusher internal geometry to be created and evaluated against multiple performance objectives. The multiple-objective approach is particularly important in mineral processing, as the optimum performance of single machines is often a trade-off between competing process drivers. A case study is presented that demonstrates the application of the technique to the design of cone crusher liners. New crusher liner profiles resulting from the application of the evolutionary algorithm suggest that significant improvements in the generation of lump product can be obtained. The extension of the approach to wider process plant design is discussed in terms of the objectives and issues to be addressed.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Comminution; Process design; Optimisation; Artifical intelligence; Evolutionary algorithms

1. Introduction

Design as applied to mineral processing is similar in concept to that applied in many other industries. The basic requirement is that a safe, efficient, sustainable design should deliver the desired results. The desired results vary, but again the essentials are that the basics are covered, with the other main drivers being the optimisation in terms of minimum capital, minimum operating cost and maximised return on investment.

The dilemma is always that to reach a design that satisfies the requirements, and covers all bases, would demand a level of study that would consume too much time and money. The design process therefore needs to be automated. Automation traditionally meant the methodical application of certain processes in a "blind" manner, i.e. a massive iterative search that ploughed through a huge number of alternatives.

The application of optimisation algorithms to "blind" searches has led to more efficient techniques, but essentially these techniques tend to provide a means of narrowing the search, which in turn can preclude the pursuit of options that may be desirable.

Essentially the ideal method should not limit the design scope at any stage in the synthesis, but rather it should develop and modify the search to reach the best solution.

For a computer based system to reach this goal there is a requirement for it to have some form of "intelligence". There has been much made of the term "intelligence", particularly related to the field of Artificial Intelligence, or A.I., and dependent on the field of investigation, A.I. has either delivered or not. Mineral

* Corresponding author. Tel.: +61 8 9327 2920; fax: +61 8 9327 2999.
E-mail address: ted.bearman@riotinto.com (R. Bearman).

doi:10.1016/j.mineng.2004.06.018

Processing is a field that has seen limited application of A.I., although there are certainly some success stories.

2. Previous A.I. approaches for mineral processing

The A.I. approach most commonly applied to mineral processing is that of Expert Systems (ESs). Expert Systems use logic, decision making and knowledge processing. In their simplest form, they are no more than a decision-tree, however in their more complex mode they have been integrated with Fuzzy Logic, Neural Networks (NNs) and numerical models, thus providing better ways of dealing with vague data, representation of complex data sets and the representation of equipment performance.

Commercialisation of ESs has always been a difficulty, and even at the height of popularity it was quoted that only 4% of all systems developed reached commercialisation where they were in everyday use. Where ES approaches have impacted the minerals industry is in the high-level control area. The commercially available G2 ES has found a niche in the control of process plants. In a typical application G2 sits over the top of a Distributed Control System (DCS) and uses its knowledge base of information to make decisions, usually in an advisory capacity.

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 in an expert system approach. Using this toolkit, MinnovEX has built expert systems that can handle unit operations, including SAG mills, AG mills, ball mills, flotation columns and cells, crushers, feeders, and pelletizers.

In the case of the SAG mill controller, the system uses variables including circuit feed rate, water addition, mill speed, mill load and steel addition to stabilize and optimize the operation of the circuit. The output is communicated directly to a SCADA system developed for the application. MinnovEX report that 15 such control systems have been installed in the mineral sector.

Another well-reported ES based system is known as OCS (Broussaud and Guyot, 1999). OCS employs real-time process control software with embedded modules that provide flexibility to the base ES namely:

- Fuzzy Expert module.
- Soft Sensor module with one (or several) adaptive predictive model(s).
- Optimiser module.
- Neural Network module.
- Vision Module.

Attempts to use ESs in the design of processing flowsheets have been less successful. Bearman et al. (1990) used a simple forward searching, decision tree to design quarry plant flowsheets for the generation of various size fraction products. The designs invariably led to process designs that would achieve the required throughput and product sizes, but iterative considerations such as recirculating load were simply factored in to the overall capacity of the machines. As such the designs were simply a solution to a particular problem, not an optimised solution.

A more advanced approach to the issue of flowsheet design using ESs was proposed and examined by Prince (1997). The aim of Prince's work was to use an ES coupled to existing numerical models of equipment to build a flowsheet in a unit-by-unit manner. The circuit was solved using the net profit generated from the circuit as the goal. That is, 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 (e.g., numerous re-cycle streams and maximum size reduction).

A development beyond the constraints of the ES approach was investigated by Venter et al. (1997). Venter proposed an approach from the field of Genetic Algorithms (GA), namely Learning Classifier Systems (LCS). Using this technique objects programmed with intelligence in various forms 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 heuristic, or empirical data, leading to subjectivity. 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.

In the next section, the field of evolutionary computation and its application, including GA and associated techniques, is examined.

3. Evolutionary computation

Evolutionary computation is the generic name given to the family of population-based optimisation algorithms that mimic the process of evolution observed in nature to solve problems in computers.

Following Darwin's principle of natural selection, evolutionary algorithms model the processes of reproduction, mutation, survival of the fittest, and genetic inheritance in the computer, thus ensuring a population subjected to a competitive environment (competition in the form of space in computer memory) "evolves" to the environment at hand. If the simulated environment represents a problem to be solved, the population learns to solve that problem.

Note though that evolution is not a directed process—no prior knowledge of the problem is required ahead of time in order for the population to evolve to solve the problem of interest. Instead, evolution is an indirect process—solutions that (by chance) are "better" at solving the problem are rewarded more than the "weaker" members of the population. Reward in an evolutionary sense involves passing on genetic material through reproduction—"better" members are rewarded by reproducing more frequently than the "weaker" members of the population. Over time, those traits that produced the advantage will become dominant in the population and new advantages will be realised. Evolution is indeed blind; this feature allows evolutionary algorithms to be applied to problems where traditional A.I. approaches fail.

Evolutionary algorithms can take many shapes and forms. This work will focus on one form of evolutionary algorithm, a specialised form called an evolution strategy.

The basic evolution strategy algorithm has the following steps:

1. Create an initial population of candidate solutions.
2. Evaluate the success of these solutions.
3. Create a population of child solutions by mutating the members of the current population.
4. Evaluate the success of these child solutions.
5. Select the "best" solutions from the parents and children together.
6. Repeat steps 3 to 5 until some termination criterion is met.

The algorithm proceeds by maintaining a population of candidate solutions that encode information about the problem to be solved (akin to DNA in humans encoding information about how to build a person). As the algorithm proceeds, this population of candidate solutions forms the population of parent solutions for the next "generation" of the algorithm. These parent solutions produce (via reproduction) offspring solutions, which are typically closely related to their parents (genetic inheritance). However reproduction is not an exact process and random errors (mutations) are introduced during the reproduction process. Since only a fixed number of solutions are maintained from one generation to the next (competition for limited space in the population), the population is subjected to an "evolutionary selection pressure", forcing the population to evolve to solve the problem at hand.

Step 5 of the evolutionary strategy algorithm above requires selecting the "best" solutions from the combined parent and offspring population. But how is "best" determined?

The answer lies in steps 2 and 4 of the evolutionary algorithm. These steps call for evaluating the success of candidate solutions. Through these steps, each candidate solution is assigned a numerical value of goodness (called the solution's "fitness") that represents how good that solution is at solving the problem. Step 5 of the algorithm then uses these fitness scores to determine which candidate solutions are kept in the next generation.

Evolutionary computation approaches have been used for a variety of design problems, ranging from the design of geometric shapes (e.g., jet nozzle design by Klockgether and Schwefel (1970) and flywheel design by Eby et al. (1999)), through to the design of networks (e.g., heat exchanger design by Gross et al. (1996) and the design of chemical engineering process structures by Emmerich et al. (2001)).

4. Case study—an evolutionary algorithm for crusher liner design

Crushers are critical components of the comminution process—they provide a relatively simple means of crushing raw material into smaller sizes. The performance of a crusher is governed by many factors, Bearman (1994), but a key determinant is the design and profile of the liners that constitute the actual crushing surface. This case study investigates how well an evolutionary algorithm is capable of optimising both the design of the internal liners of the crusher and the control settings that drive the crusher. This is achieved by evolving the design of a single crusher in a simple comminution circuit.

4.1. The problem

Fig. 1 shows the comminution circuit investigated in this study. The circuit consists of a single input stream, one cone crusher, and one screen for separating product from oversize material. The feed enters the system via the conveyor from the top left and falls into the crusher. The crushed material is then passed through a screen that allows particles less than 32 mm to pass through and report as product. Oversize particles larger than 32 mm are recycled back to the crusher for further crushing.

Fig. 2 shows a schematic diagram of the cone crusher used in this study. Material is introduced into the crusher from above, and is crushed as it flows downwards through the machine. The inner crushing surface, or "mantle", is mounted on the conical crushing head and is driven in an eccentric motion swivelling around the axis of the machine. The outer crushing surface, or "bowl", is held stationary. Material flows into the crushing chamber from above, and is crushed between the

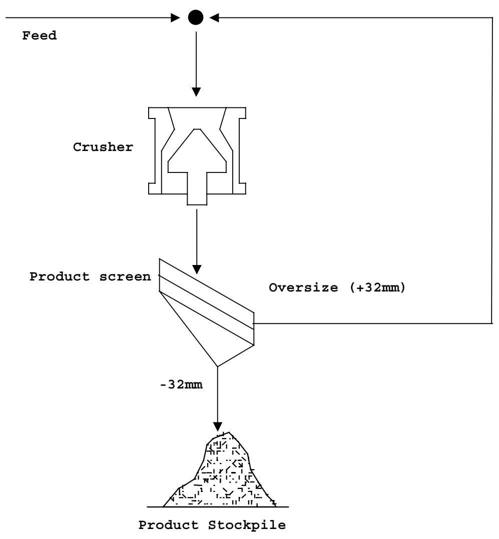


Fig. 1. The simple comminution circuit used in this study.

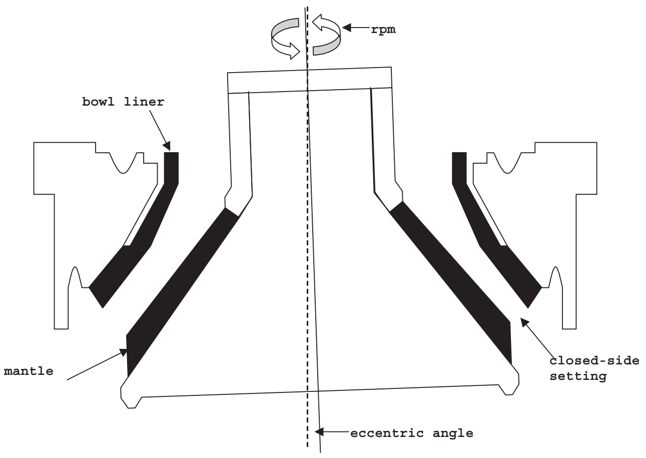


Fig. 2. Schematic diagram of the cone crusher used in the comminution circuit used in this study (original concept taken from Napier-Munn et al. (1996)).

two surfaces by compressive forces due to the eccentric motion. After compression, the chamber widens and allows material to flow to lower parts of the crushing chamber, and eventually to fall through and exit the machine.

The principle goal of this circuit is to generate material of the desired size, i.e. −32 mm. This single goal can obviously be achieved relatively simply by reducing the closed-side setting of the crusher and increasing the speed of rotation. However, this improvement in size reduction comes at a cost—the throughput of the circuit is dramatically reduced due to the decrease in volume the crusher can handle. A balance between improving size reduction and reducing the throughput of the circuit must be struck. That is, both factors (size reduction and throughput) must be considered in assessing the relative success of a potential design.

4.2. Crusher simulation

To determine the success of a candidate design, the performance of the design in the comminution circuit must be evaluated. This is done via simulation. The inputs to the simulation are the:

- Physical properties of the feed (composition, hardness, etc);
- Size distribution of the feed (the proportion of particles in different size fractions);
- Geometry of the mantle and bowl liners;
- Closed-side setting;
- Rotational speed of the head and
- Eccentric angle of the head.

The final four of these were selected as the design variables for the chosen problem. The outputs of the simulation are the:

- Size distribution of the product;
- Power needed to crush the feed and
- Maximum amount of material that can flow through the crusher without overloading the crusher (i.e., its capacity).

From these outputs it is possible to calculate the steady-state size distribution of the product and capacity of the circuit that includes the crusher. This output data is used to evaluate the performance of proposed designs and is used by the evolutionary algorithm to determine which candidate solutions are selected as parents in the next generation of the algorithm. Each evaluation takes approx 300 ms on a 700 MHz Pentium III computer.

The problem described above is well suited to an evolutionary computation approach. The problem cannot easily be described analytically, but a simulation is available that can be used to evaluate candidate solutions. The search space is large—too large for an exhaustive search—and there is little to guide an engineer in determining good designs for a given scenario.

5. Design of the evolutionary algorithm

In this section, the design and specification of the different components making up the evolutionary algorithm are discussed.

To implement an evolutionary algorithm, the following must be specified:

- The representation scheme to be used;
- The nature of the mutation operators;
- Constraints imposed on variations produced by the mutation operator;

- The initialisation method for seeding the initial population;
- The termination condition of the algorithm;
- The selection mechanism for choosing which candidate solutions proceed into future generations and
- The method of fitness evaluation to be used by the algorithm.

We must also address the problem of what to do with infeasible solutions—solutions generated by the algorithm that are deemed to be illegal (e.g. a liner with a non-sensical shape).

These specifications are detailed in the remainder of this section.

5.1. Representation

The representation of the internal machine settings, (the closed-side setting, eccentric angle, and rotational speed) is straightforward—these being real values within given ranges. The best way to represent the geometric shapes of the two liners is less clear. The shape of each liner is defined by its vertical cross-section. The shape of the machine structure dictates the shape of the "back" of each liner, so it is only the "front" of each liner (the actual crushing surface) that is represented.

Each shape is represented as a series of line segments, using a variable-length list of points, each represented by a pair of coordinates. The first coordinate pair for the first segment and the last coordinate pair for the last segment are fixed, but each other coordinate is another real-valued object variable. Thus, if there are n line segments on the mantle and m line segments on the bowl liner, then the genotype consists of a vector of

$$3 + 2(n-1) + 2(m-1)$$

real-valued object variables.

5.2. Mutation

When a parent is mutated to produce a child, each object variable is mutated independently using self-adaptive mutation rates (allowing each candidate solution to control its own mutation amount) as described in Back et al. (1997). Specifically, each object variable is mutated using the formula:

$$X_i' = X_i + \sigma_i' \cdot N_i(0,1)$$

where $N_i(0,1)$ is a normally distributed random value with mean 0 and standard deviation 1. Each strategy parameter σ_i is mutated using the formula:

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$$

where τ and τ' are constants set to 0.25 and 0.1 respectively and $N(0,1)$ is sampled once for each individual.

In addition, mutation operators were provided to increase or reduce the number of segments in a liner. Whether to apply these operators is determined randomly with a fixed probability. The mutation to reduce the number of segments randomly selects two adjacent segments to merge and discards the common end point. The operator to increase the number of segments randomly selects a segment to split into two, using the segment midpoint as the common end point. This was done to allow the algorithm to generate more complex or simpler liner shapes as required.

5.3. Constraints

There are a variety of feasibility constraints upon potential designs. These can be categorised as follows:

1. *Physical constraints*: The sequences of coordinate pairs must describe shapes that make sense operationally. In particular, the liners must have at least a certain thickness to be practical. This constraint was violated so rarely that it is not worth the computational expense to perform the validity checking. If the final solution returned violates this constraint, the algorithm can simply be re-run.
2. *Setting constraints*: Each machine setting must be confined to a given range. This is done by parameter repair—any mutated value that is too low is set to the minimum value for that setting and any that is too high is set to the maximum value.
3. *Modelling constraints*: The crusher simulation is very complex and assumes (sometimes implicitly) that liners have "sensible" shapes. To keep our designs in the "sensible" region, we impose a heuristic constraint that the sequence of x-coordinates and the sequence of y-coordinates for each liner surface always change monotonically. This constraint is enforced by repairing any coordinate that violates this constraint at the time of creation. Even so, the simulation occasionally fails. In these cases, the design is assumed to be nonsensical and assigned a fitness score of 0, quickly eliminating it from further consideration by the algorithm.

5.4. Initialisation

The population of candidate solutions is initialised with copies of the existing standard design, but mutated control settings. These copies are quickly eliminated in the first few generations of a typical execution.

5.5. Termination criterion

The algorithm is run for a fixed number of generations before terminating. At the end of the run, the best solutions found by the algorithm are reported back for further investigation.

5.6. Selection

Selection is the process of deciding which candidate solutions from the parent and offspring populations proceed (or survive) into the next generation. Selection determines which solutions will act as parents in the next generation, thus determining which solutions contribute to the evolution of the population.

For this problem, we follow the methodology of an $(\lambda + \lambda)$ evolutionary strategy—each member of the current generation produces precisely one child, thus generating an offspring population exactly the same size as the parent population (the combined parent and offspring population is hence twice the size of the original parent population). Selection then proceeds by choosing the best half (ranked on fitness) of this combined population to proceed into the next generation.

6. Defining a fitness metric for the evolutionary algorithm

The last thing to consider when designing an evolutionary algorithm is that of how to measure the success of a candidate solution (i.e., how to calculate a candidate solution's fitness). In this section we explore two different methods for determining the fitness of candidate solutions in the evolutionary algorithm.

As already discussed, the principal objective of interest in this case study is the minimisation of the size of the product produced by the crusher in the comminution circuit. Specifically, we define *P80* to be a measure of the size of the 80th percentile in the product (i.e. the size k mm such that 80% of the product is smaller than k mm). Using the native representation of ore particles in the crusher simulator, a higher value of *P80* corresponds to a smaller product, so the algorithm must maximise the value of *P80*.

Alongside minimising the size of the product produced by the circuit, we also want to maximise the capacity of a circuit. The placement of the crusher in a circuit is important because a crusher that itself has a high capacity may not be suitable if it generates a lot of oversize material: the presence of this recirculating material reduces the rate at which feed can be introduced into the circuit. The term "capacity ratio" is defined as the ratio of the amount of material entering the crusher to the amount of feed entering the circuit (at steady-state operation). A higher capacity ratio corresponds to more recirculating material.

The capacity of a circuit may be limited by one of three factors:

1. The capacity of the crusher. If a crusher has capacity CAP tons/h and capacity ratio CR, the capacity of the circuit will be limited by:

   CAP/CR

2. The power requirements of the crusher. A high rotational speed in particular delivers a lot of crushing but requires a lot of power. If a crusher with maximum power output MP kWh requires P kWh to process a circuit feed of F tons/h, the capacity of the circuit will be limited by:

   $F \times (MP/P)$

3. The capacity of the recirculation conveyor in the circuit. If a crusher has capacity ratio CR and the conveyor has a capacity of MR tons/hour, the capacity of the circuit will be limited by:

   $MR/(CR - 1)$

Each of these factors potentially limits the capacity of the circuit. Therefore, we define the actual capacity of the circuit as the minimum of these values.

Notice the potential trade-offs for the various design variables. For example, a large closed-side setting will increase the capacity of the crusher, but will also increase the amount of recirculating material, raising the capacity ratio. Similarly, a high rotational speed will lead to more crushing in each pass through the chamber, but will also increase the power requirements of the crusher, possibly reducing the overall capacity.

Having now defined two potentially competing objectives (*P80* and capacity), it must be determined how to resolve these metrics together for use in the evolutionary algorithm. That is, a method is required to determine some way of combining these values together to produce a measure of success (fitness) of a candidate solution.

6.1. A single objective evolutionary algorithm

Hingston et al. (2002), uses a single objective fitness metric that combines the two competing objectives into a single fitness score via a linear combination of the two was introduced.

Firstly, both the capacity and *P80* size figures are normalised by dividing by the corresponding values for the standard design and settings. The actual fitness of a candidate solution is then determined by the formula:

$$0.05 \times CAP + 0.95 \times P80$$

where CAP is the capacity of the circuit, *P80* is the size measure, and the constants are chosen to equalise the variability of the two components. Thus the fitness of the standard design is 1.0. Higher fitness is better.

6.2. A multi-objective evolutionary algorithm

Barone et al. (2002), uses an alternative way of the combining the two competing measures together to determine the fitness of a candidate solution. Instead of using a linear combination of the two measures, both objectives are used to define the Pareto ranking of a design relative to a set of potential designs.

The ranking scheme proposed in Fonseca and Fleming (1998), and further described in Veldhuizen and Lamont (2000) is used. Pareto dominance for crusher designs is defined as follows:

A design u is said to *dominate* a design v iff $CAP(u) > CAP(v)$ and $P80(u) > P80(v)$.
A design x is *Pareto optimal* with respect to a set of designs Ω iff there is no design in Ω that dominates x. Thus a design that is Pareto optimal cannot be improved in any objective without degrading other objectives.
Finally, the *Pareto rank* of a design x, with respect to a set of designs Ω, is the number of designs in Ω which dominate x.

Thus x is Pareto optimal iff x has a Pareto rank of 0.

In this multi-objective approach, Pareto rank, rather than a composite fitness value, is used as the basis for selection in the evolutionary algorithm.

Using Pareto rank however introduces a problem. In the multi-objective algorithm, different candidate solutions will often have the same Pareto rank (and hence fitness). In fact, it is desirable for the Pareto "front" of equally ranked solutions to expand over time to increase coverage of the search space. Indeed, towards the end of the run, we want most (if not all) of the population to be Pareto optimal. This introduces the problem of what to do when forced to select between equally ranked (and hence equally good) candidate solutions, i.e. given m equally ranked candidate solutions and $n < m$ free spaces in the population, select n solutions from the m to fill the population. How does the algorithm decide which of these equally good solutions should be kept and which should be discarded (note that is this is unlikely to occur in the single objective algorithm, as a candidate solution's fitness is scored not by integer-valued rank, but by a real-valued number)?

Typically the choice between equally ranked solutions is either made randomly or based on the proximity to "near-by" solutions. In this work however, a variable population is used, thus allowing the parent population of candidate solutions to grow and shrink over time. Thus, the issue of selecting between equally ranked solutions is avoided—the algorithm can proceed without ever needing to discard good solutions. Experiments show that regular "revolutionary" breakthroughs cap the growth of the population to a manageable size.

7. Results and discussion

In this section, results of experiments with the two different algorithms introduced in the last section—the single objective evolutionary algorithm and the multi-objective evolutionary algorithm—are presented.

7.1. Single objective results

In this section a series of experiments with the single objective algorithm are described. An example set of runs produced by the algorithm is presented that is indicative of the performance attained on the test problem.

The algorithm was run ten times with a population size of 50 for 200 generations on each run. Table 1 shows the performances of the best designs from these runs.

The results show an average increase in capacity of around 140%, and around 10% in *P80*. Indeed large financial rewards can be realised by utilising the designs produced by the system.

Fig. 3 shows how the fitness values and the two components, *P80* and capacity, evolve during a typical run, Run 10. Improvements in capacity have been scaled down by a factor of 19 to reflect the fitness function

Table 1
Performance of the best designs from ten runs of the single objective evolutionary algorithm

Run	Capacity	*P80*	Fitness
1	2.068	1.106	1.154
2	2.176	1.106	1.160
3	1.981	1.108	1.152
4	2.288	1.101	1.160
5	3.087	1.094	1.194
6	1.781	1.117	1.150
7	1.958	1.106	1.149
8	3.065	1.093	1.191
9	2.591	1.105	1.180
10	2.947	1.107	1.199

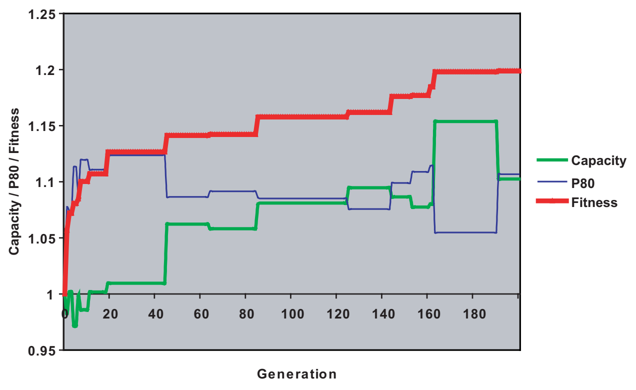


Fig. 3. Plot of the fitness progression of Run 10 from Table 1 of the single objective evolutionary algorithm.

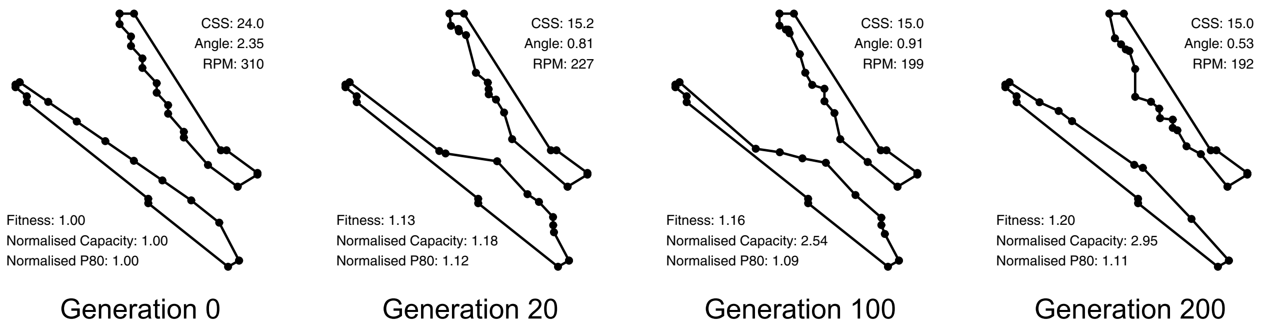


Fig. 4. The best liner pairs at different stages through one run of the single objective evolutionary algorithm.

scaling. It can be seen that improvements tend to be made by favourable trade-offs between the two components.

Fig. 4 shows the best liner pairs from selected generations evolved during another run. It can be seen that the evolved shapes are distinctly different from the standard design. Whilst engineers can provide a post-hoc rationale for the revised design, and this provides confidence in the validity of the designs, it is virtually impossible to predict in advance the effect of a change in shape, much less to intuit a high quality design for a specific scenario.

It is worth noting that each run takes around 30 min to complete. In a real design exercise, a running time of several hours (or even days) would still be very acceptable, so there is plenty of scope for increased task complexity in the future.

7.2. Multi-objective results

In this section, an example set of runs of the multi-objective algorithm on the test problem is described. The system was run five times with a population size of 50 for 200 generations on each run.

Fig. 5 shows the final Pareto fronts for the five sample runs. In all cases a good range of designs is found, showing different trade-offs of the two objectives.

Fig. 6 shows the evolution of the Pareto front during Run 5 from Fig. 5. Note that while the fronts for Generations 100 and 200 appear to cross, no design in Generation 200 is actually dominated by one in Generation 100. Some designs in Generation 100 are still present in Generation 200 (indeed one design in Generation 20 is still present), and the use of lines to interpolate between the population members creates the illusion of a crossover. The situation is exacerbated by the difficulty in improving *P80* values beyond a certain level: this has been confirmed by experiments where maximising *P80* was the sole objective.

Results depicted in Fig. 6 hint at the diversity of solutions found by multi-objective approach. Indeed, the best-evolved crusher for the *P80* measure has shown over a 12% increase in *P80* (and a 9% increase in capacity), while the best-evolved crusher for capacity has shown over a 224% increase in capacity (while sacrificing less than 1% in *P80*). The best "all-round" solution has managed to increase both *P80* and capacity significantly.

Fig. 7 shows a sample of evolved liner pairs and settings from another run. The first row shows liners from

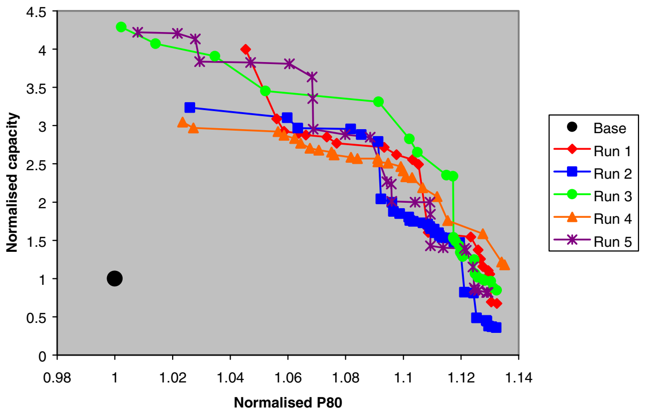


Fig. 5. The final Pareto fronts from five runs of the multi-objective evolutionary algorithm.

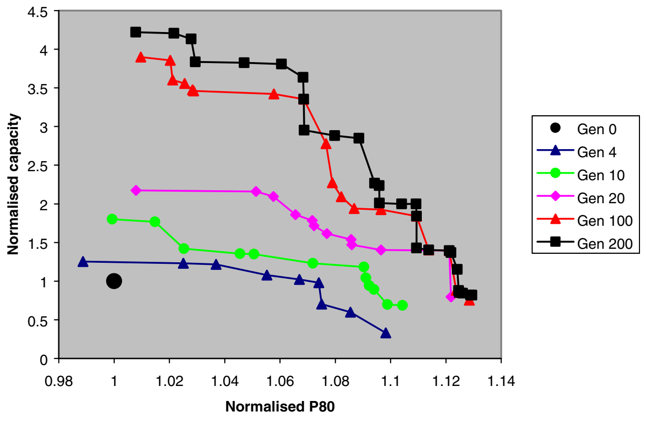


Fig. 6. The progression of the Pareto front of Run 5 of Fig. 5 for the multi-objective evolutionary algorithm.

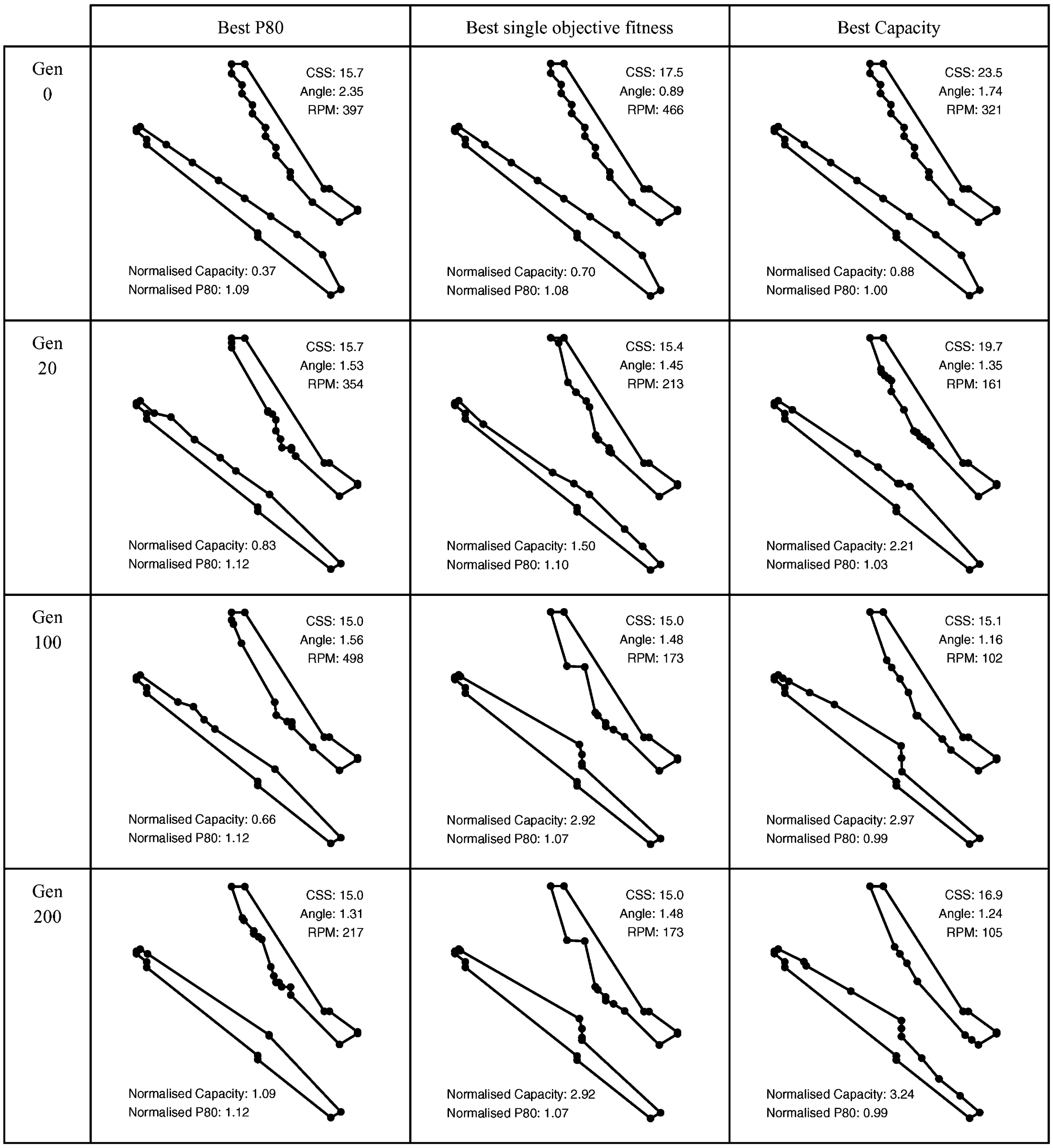


Fig. 7. The best liner pairs at different stages through one run of the multi-objective evolutionary algorithm.

Generation 0—a selection of random mutations on the standard design. The middle rows show liners from part way through the run, and the final row shows liners from the final Pareto front. The first column shows the design with the best *P80*, while the last column shows the design with the best capacity. The second column shows the design with the highest fitness according to the single objective fitness measure.

The true multi-objective approach used in the second algorithm offers clear benefits in this application over the simpler approach of using a combined fitness function. The multi-objective algorithm removes the need for arbitrary weightings, which engineers have trouble specifying in advance. There is no need to separately apply capacity constraints, as non-dominated solutions inevitably satisfy the constraints anyway.

One final advantage of the multi-objective algorithm over the single objective approach is the presentation of a range of potential solutions. Whereas the single objective algorithm returns an overall best solution, the multi-objective approach presents a range of possible solutions, each optimal in different combinations of the different objectives. This gives the design engineer the opportunity to pick and choose amongst a selection of different designs—the human engineer is then able to select the design that best fits her unique problem and is able to consider all the non-human factors in her choice that otherwise could not be modelled by the computer.

7.3. PPA—the evolutionary algorithm crusher design tool

The culmination of this work is a system capable of optimising the design of crushers. The resultant system is christened PPA.

Fig. 8 shows the user interface of the PPA system during the execution of a typical run. The top right corner shows a scatter plot of the current generation in objective-space. The user can select a particular design in the plot to view its details elsewhere on the screen. The top left corner depicts the selected crusher in a circuit: it shows the liner shape and the material flows through each part of the circuit. The user can click on one of these flows to view a graph of the size distribution of the corresponding ore stream. The bottom left corner shows the settings and fitness for the selected design. The bottom right corner has various controls for the parameters of the system.

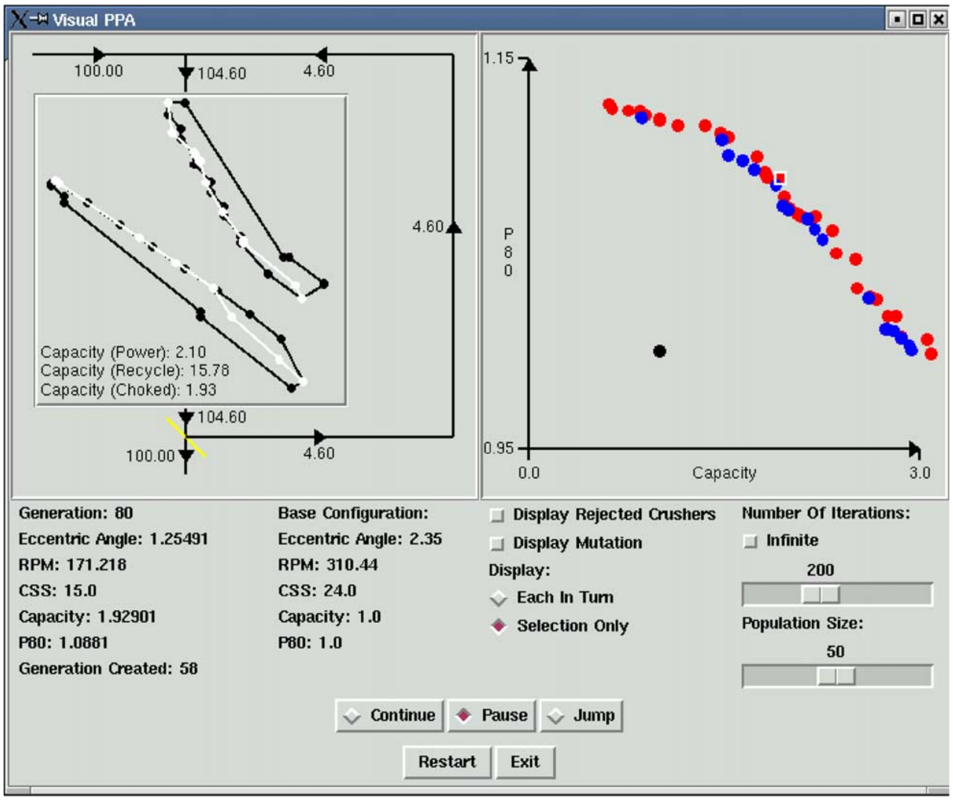
The user interface also provides an intuitive visualisation for engineers, enabling them to see the effects of trading off the different objectives on the evolved designs. This gives the engineer confidence in the process—instead of relying on the "magic" of the single objective algorithm in producing one overall best solution, the engineer is able to explore the range of potential solutions found by the multi-objective approach in order to determine the different trade-offs made by the system.

8. Future work—an evolutionary algorithm for flowsheet design

The design of comminution circuits relies on the assembly of nominally suitable equipment into a flowsheet form. The design is supported and confirmed by material characterisation, pilot plant testing and process simulation. Given that there are always time-cost constraints on the design process it is inevitable that this can lead to non-optimal designs.

Any technique that can produce a fast, efficient analysis of the whole space around the design issue is valuable. For the technique to additionally examine modifications and implement evolution of the designs as part of the design process is an exciting prospect.

Hence the goal of this work is not only the automated design of individual comminution circuit units such as cone crushers, but also the automated creation and optimisation of entire comminution circuits. As with the opt-



Capacity (Power): 2.10
Capacity (Recycle): 15.78
Capacity (Choked): 1.93

Generation: 80	Base Configuration:
Eccentric Angle: 1.25491	Eccentric Angle: 2.35
RPM: 171.218	RPM: 310.44
CSS: 15.0	CSS: 24.0
Capacity: 1.92901	Capacity: 1.0
P80: 1.0881	P80: 1.0
Generation Created: 58	

Fig. 8. The user interface of the PPA system.

imisation of the cone crusher, evolutionary computation techniques can also be applied to comminution circuit optimisation. However, for this approach to be successful, an appropriate abstract representation for comminution circuits, that is well suited to evolutionary operators such as mutation and reproduction, must be chosen.

Some form of graph-based representation of the comminution circuit would seem a natural abstraction to use. Indeed, similar approaches have been used in the past. For example, Emmerich et al. (2001) employed an evolutionary graph-based optimisation approach for the optimisation of the Hydrodealkylation (HDE) process, a chemical engineering problem.

Analogously, in the domain of comminution circuit optimisation, graph nodes would represent comminution units such as cone crushers and cyclones, and directed edges would identify material flow between units. Specific unit characteristics, such as a cone crusher's closed-side setting and eccentric angle, would be encoded as parameters associated with nodes, and similarly edges would be associated with parameters describing feed characteristics, such as mineral composition and size distributions.

Initial thoughts suggest that this representation seems satisfactory; however, other graph-based possibilities exist (e.g. ore streams could be encoded as nodes, while graph edges could represent the processes that act on the different streams). Further research is required to determine which approach is best for the application to comminution circuits.

Moreover, assuming an underlying graph-based representation, appropriate graph structure oriented mutation and reproduction mechanisms must also be defined. Simple mutation via randomly inserting or deleting units in the graph structure would result in a massive search space full of unfeasible solutions, as would randomly swapping parts of different graph fragments between population members during reproduction. However, requiring correctness preserving mutation and reproduction operators might impede the search process, or require expert domain knowledge to be directly encoded into the search process. Identifying the best trade-off between these two is not likely to be trivial. This will be addressed in future work in this area.

9. Conclusions

This study has investigated the application of evolutionary algorithms to industrial design. In particular, the effectiveness of evolutionary algorithms on a difficult practical engineering design problem—that of optimising the design of the internal liners and control settings for a crusher in a comminution circuit—has been discussed. Initial results promise significant financial benefits.

In many ways, this problem is an ideal application for evolutionary algorithms—the pay-off is high; the problem is too complex to solve analytically; the search space is too large to explore unaided; a well defined evaluation function exists for the problem; and a straightforward representation scheme, suitable for manipulation by genetic operators is easy to define. Many challenges remain in incorporating more realism in the problem definition (e.g., including variety in feed) and validating the predicted performance with field trials.

Acknowledgments

This work was supported by Rio Tinto Research Grant BTC005. The authors would like to acknowledge the support and assistance of Rio Tinto Technical Services.

Part of this work was supported by Australian Research Council Linkage Grant LP0347915.

References

Back, T., Ulrich, H., Schwefel, H.-P., 1997. Evolutionary computation: comments on the history and current state. IEEE Transactions on Evolutionary Computation 1 (1), 3–16.

Barone, L., While, L., Hingston, P., 2002. Designing crushers with a multi-objective evolutionary algorithm. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers, pp. 995–1002.

Bearman, R.A., Barley, R.W., Hitchcock, A., 1990. The Development of a Comminution Index for Rock and the Use of an Expert System to Assist the Engineer in Predicting Crushing Requirements. Minerals Engineering 3 (1/2), 117–127.

Bearman, R.A., 1994. Cone crusher design and operation, Proceedings of 1st CMTE Annual Conference, Pinjarra Hills, Brisbane, July 20–22nd 1994, pp. 56–59.

Broussaud, A., Guyot, O., 1999. Optimising control systems: improving profitability in minerals processing., Ind. Miner. Lond., no. 384, Sept. 1999, pp. 101–107.

Eby, D., Averill, R.C., Punch, W.F., Goodman, E.D., 1999. Optimal design of flywheels using an injection island GA. Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 13, 327–340.

Emmerich, M., Grötzner, M., Schuetz, M., 2001. Design of graph-based evolutionary algorithms: a case study for chemical process networks. Evolutionary Computation 9 (3), 329–354.

Fonseca, C.M., Fleming, P.J., 1998. Multiobjective optimisation and multiple constraint handling with evolutionary algorithms—Part I: unified formulation. IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans 28 (1), 26–37.

Gross, B., Hammel, U., Maldaner, P., Meyer, A., Roosen, P., Schutz, M., 1996. Optimization of heat exchanger networks by means of evolution strategies. In: The Sixth Conference on Parallel Problem Solving from Nature. Springer, pp. 1002–1011.

Hingston, P., Barone, L., While, L., 2002. Evolving crushers. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02). IEEE Publishers, pp. 1109–1114.

Klockgether, J.S., and Schwefel, H.-P., 1970. Two-phase nozzle and hollow core jet experiments. 11th Symposium on Engineering

Aspects of Magnetohydrodynamics, California Institute of Technology, pp. 141–148.

McCaffery, K.M., Katom, M., Craven, J.W., 2001. Ongoing evolution of advanced SAG mill control at Ok Tedi. Prepr. Soc. Min. Metall. Explor., no.01-69.

Napier-Munn, T.J., Morrell, S., Morrison, R.D., and Kojovic, T., 1996. Mineral Comminution Circuits. Julius Kruttschnitt Mineral Research Centre.

Prince, R.G.H., 1997. Computer Aided Design of Mineral Processing Flowsheets, AMIRA Project P452, Final Report.

Veldhuizen, D., Lamont, G., 2000. Multiobjective evolutionary algorithms: analysing the state-of-the-art. Evolutionary Computation 8 (2), 125–147.

Venter, J.J., Bearman, R.A., Everson, R.C., 1997. A Novel Approach to Circuit Synthesis in Mineral Processing. Minerals Engineering 10 (3), 287–299.