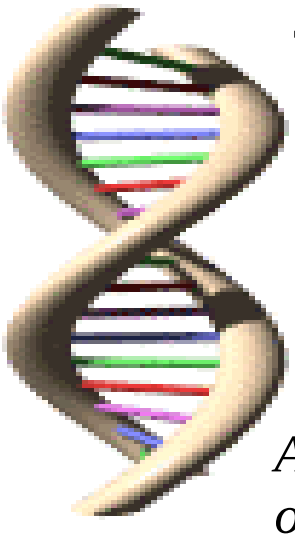
Dr P. Hingston
Dr L. While
Dr L. Barone

August 2003

Evolutionary Computation for Industrial Design



A background briefing on the use of evolutionary computation to optimise industrial designs

DRAFT

Executive Summary

Nature has a process that optimises the design of living organisms. This process is called evolution. Evolution is based on a set of key principles including the use of a genetic code and Darwinian natural selection. Today's scientists and engineers can use the same principles to solve difficult design optimisation problems using computers to simulate evolution.

So-called *evolutionary algorithms* are now routinely used to solve real-world problems that are impractical to solve by traditional analytical methods. A particular application area where evolutionary algorithms can shine is the creation of industrial designs.

As an example relevant to WA industry, take the design of a processing plant for a mine. The quality of this design can have a major economic impact on a mining operation. A small percentage improvement represents a great deal of added value, and any mistakes mean lost value. Yet the process of creating and improving flowsheets for plant is largely intuitive, relying critically on the experience of the individual engineer. An automated tool that could improve the quality and consistency of flowsheet design would bring continuing benefits and competitive advantage.

We believe that such tools can be developed using evolutionary algorithms. In this position paper, we outline the principles of these algorithms and describe case studies showing the benefits achieved in industrial design settings.

We believe there is an opportunity to develop a flexible automated design tools in partnership with WA companies. This would provide industry with a cost-effective way to benefit from the expertise of researchers in the field, and would provide financial support and a real-world focus for the research and development work.

If there is in-principle support for this concept, we would like to meet with company representatives to formulate a proposal for an initial prototype development project.

Evolutionary Computation for Industrial Design

A background briefing on the use of evolutionary computation to optimise industrial designs

Introduction

Nature has a process that optimises the design of living organisms. This process is called evolution. Evolution is based on a set of key principles including the use of a genetic code and Darwinian natural selection. Today's scientists and engineers can use the same principles to solve difficult design optimisation problems using computers to simulate evolution.

So-called *evolutionary algorithms* are now routinely used to solve real-world problems that are impractical to solve by traditional analytical methods. This may be because the problem is computationally intractable, or because it involves a messy, highly non-linear, complex system. A key advantage of evolutionary algorithms is that they operate on general principles, requiring only a way to calculate which potential solutions are better ("fitter"). This is enough to guide the algorithm towards high quality solutions.

A particular application area where evolutionary algorithms can shine is the creation of industrial designs. The purpose of this document is to provide a technically able reader with a background briefing on evolutionary algorithms and how they are used to solve such problems, and to explain why we believe this approach should be applied to the problem of flowsheet design.

As an example relevant to WA industry, take the design of a processing plant for a mine. The quality of this design can have a major economic impact on a mining operation. A small percentage improvement represents a great deal of added value, and any mistakes mean lost value. Yet the process of creating and improving flowsheets for plant is largely intuitive, relying critically on the experience of the individual engineer. An automated tool that could improve the quality and consistency of flowsheet design would bring continuing benefits and competitive advantage.

A flexible automated design tool could be used in greenfields projects to create alternative high quality designs from scratch given desired quality criteria. A consulting engineer could also use such a tool in scenarios such as the following:

- The engineer is asked to evaluate the design of an existing plant — the tool could be given the current design, and would search for improvements. The suggested improvements could then undergo more detailed scrutiny and evaluation, using detailed process modelling and the engineer's practical judgement. In a similar way, proposed designs submitted by tenderers could be evaluated.

- Retrofit design — an existing plant is to be extended — the tool could be given the existing design plus proposed extensions, and would search for variations that minimise capital expenditure whilst still achieving desired performance.

- An existing plant is not performing at the level that is expected — the tool could be given the current design and operating conditions, and would search for improvements to overcome the problem.

The benefits of using such a tool include financial benefits from improved plant performance, improving use of existing capital investments, reduced risk and greater confidence in proposed designs, and time efficiency in allowing the engineer to consider more options in a given time.

We believe that such a tool can be developed using evolutionary algorithms. Evolutionary computation methods, such as *genetic algorithms*, have been used with great success in a variety of industrial design tasks, particularly for network-like designs. Examples include:

- Jet nozzles: Optimisation of the internal shape of two-phase jet nozzles, yielding a 22% improvement in efficiency over the original design [5].

- Heat-exchanger networks: Cost minimisation of heat exchanger networks, yielding savings in the order of US$100,000 - US$140,000 for each network built [3, 7].

- Nuclear reactor core: Optimisation of fuel assembly selection and arrangement, yielding solutions 2-5% better than known existing solutions [1, 9].

- Chemical engineering process plants: Optimisation of the choice and layout of components used in chemical processes [2].

- Telecommunications networks: Design of a system to evolve communication networks for widely varying criteria; system produces networks that are 3-4% better than those created by human experts [10].

- Water distribution networks: Optimisation of the layout of existing water distribution networks yielding larger costs savings than traditional programming techniques [4].

Several features of evolutionary design methods make them well suited for such applications. These features include:

- Can handle complex, non-linear problems where it is not feasible to develop analytical models — the ÒfitnessÓ calculation used to evaluate designs can be arbitrarily complex.

- Do not become ÒstuckÓ as simpler hill-climbing methods can (see later).

- Can produce superior designs that are non-obvious for a human designer.

- Can perform multi-objective optimisation, balancing different or even conflicting criteria, such as risk, throughput, capital cost, NPV, expandability and so on.

- Can produce more than one high quality solution, so that the engineer can carry out more detailed validation on a small set of candidate designs.

- Are relatively cheap to develop and run on standard computers.

- Can easily be hybridised/integrated into other systems.

In the following sections, we describe the operation of a typical evolutionary algorithm, and present some case studies showing how such algorithms are used to solve real-world design problems.

Natural Evolution

We begin with a highly simplified and abstracted description of evolution. Evolution is the process whereby successive generations of an organism are modified so as to operate better in their environment. We can think of this as a process of refining a design (the "recipe" for the organism).

The key point of Darwin's theory of natural selection is that a creature's traits (eye colour, length of toes, intelligence etc) tend to be inherited by its descendants, and that these traits affect the creature's chances of survival and reproduction. Once these principles are accepted, the process of evolution by "survival of the fittest" is assured. Those creatures whose traits help them to survive better, and thus produce more offspring, inevitably become more numerous generation after generation, and good design features accumulate.

We know now how traits are inherited (though Darwin did not). It wasn't until Watson and Crick's discovery of the structure of DNA in 1953 that we had the knowledge we needed to understand the mechanisms of chromosomes and genes.

Putting these together, we can describe the process of evolution:

> *Every creature has within its cells a "recipe" according to which its body is created - its* genotype. *In humans, this consists of a sequence of 23 pairs of chromosomes — our DNA. When a creature reproduces, part of its DNA is passed on to its offspring. In sexual reproduction, this part of the creature's DNA is combined with part of the other parent's DNA. This is called* recombination. *Recombination allows traits of the parents to be combined in different ways in their offspring. Very occasionally, an error happens when the DNA is copied from parent to offspring. This is called a* mutation. *Mutations are usually deleterious to the offspring, but, rarely, may be beneficial.*
>
> *While the offspring is developing into a new creature, parts of the DNA determine just how the body of new creature (its* phenotype*) will develop. The part of the DNA that determines a particular trait is called the* gene *for that trait, and different configurations of that gene are called* alleles.
>
> *When the new creature ventures into the world, its chances of surviving long enough to reproduce (its* fitness*) depend in part on these inherited traits. Without mutation, this process simply juggles different combinations of existing alleles. Through mutation, new alleles can be introduced, producing truly different new creatures. Over many generations, alleles that tend to produce fitter creatures will predominate, ensuring "survival of the fittest".*

Simulated Evolution

Simulated evolution mirrors this process. There are several different kinds of evolutionary algorithm that vary in approach and detail. The most common are evolutionary strategies, evolutionary programming, genetic programming and genetic algorithms. We will describe genetic algorithms since they are the variant most often used in industrial design.

Genetic algorithms (GAs) are used to find high quality solutions to a particular problem. The GA maintains a population (usually a few 10's or 100's) of potential solutions. Individual solutions are randomly selected for reproduction from this population based on their *fitness* (how well they solve the problem). Performing recombination and mutation on the selected individuals creates a new generation of solutions. This process is repeated until a sufficiently high quality solution is found (usually a few 10's or 100's of generations), and the best solutions seen are kept. To make this clearer, here are the steps in a "vanilla" GA:

1. Create an initial population of potential solutions;
2. Evaluate the fitness of each potential solution;
3. Select parents for the next generation of solutions based on their fitness;
4. Apply recombination (crossover) operators to create new solutions;
5. Apply mutation operators to promote diversity;
6. Repeat steps 2-5 until done.

There are also specially designed variants of this basic process for multi-objective problems and for dynamically changing problems.

To apply a genetic algorithm to a real world problem, the details of these steps need to be filled in. This means determining a representation scheme, a fitness function, any non-standard genetic operators, and specifying an initial population.

Representation Scheme

A Ògenetic codeÓ must be designed to suit the problem. In nature, the DNA sequence (a very large sequence of symbols from a 4-letter alphabet of amino-acids) is used. We need to find a suitable way to encode possible solutions to the problem as sequences of symbols from some alphabet, often just 0Õs and 1Õs. The representation chosen can have a large impact on the success of the GA.

Fitness Function

This is a method for calculating a single number that indicates how well a potential solution solves the problem. It could be as simple as plugging solution values into a formula, or as complex as running a simulation using the suggested solution. The fitness function provides direction to the search.

Genetic Operators

There are Òstandard Ó operators that mimic crossover and mutation in nature, but some problems are better solved using specially created operators that take account of knowledge of the problem. For example, special operators are often used to solve network design problems.

Initial Population

The initial population can be created randomly or can be ÒseededÓ with reasonable solutions that were kept from past problems or were created using some other method.

We will now illustrate the approach with some relevant case studies.

Case Study 1: Jet Nozzle Design

The first case study that we consider is a classic in the field, dating from 1970 [5]. This case illustrates the basics of the evolutionary approach and shows how it can lead to excellent designs of a type that would never be suggested by a human designer.

The task is to determine the internal shape of a two-phase jet nozzle with maximum efficiency under constant starting conditions. Some minor structural constraints were placed on the design: the incoming diameter was fixed at 32mm, and the smallest diameter was fixed at 6mm, resulting in a convergent-divergent (Ò two-phase Ó) structure for the nozzle.

Methodology

In accordance with the methodology described above, the first step in the design process is to define a Ògenetic codeÓ capable of representing nozzle designs. A given nozzle is constructed from a number of conical segments: each segment is 10mm in length, and each is specified by its out-going diameter (always a multiple of 2mm). The genotype thus has the form

$$[z_1, z_2, D_{z1}, \ldots, D_{-1}, D_1, \ldots, D_{z2}]$$

where z1 is the number of segments in the convergent part of the nozzle, z2 is the number of segments in the divergent part, and the D_k are the diameters of the segments. For example, the sequence

[4, 3, 26, 20, 14, 8, 12, 16, 20]

represents a nozzle whose in-coming diameter is 32mm, reducing smoothly in four segments to 8mm, then expanding smoothly in three segments to a final diameter of 20mm. This is somewhat similar to the initial nozzle used in the actual execution, shown in Figure 1 (taken from [11]). This code can represent nozzles up to an arbitrary length and with arbitrary internal shapes.

Figure 1. The starting nozzle

The fitness function used was simply the thrust to mass velocity ratio (a measure of efficiency) achieved by each nozzle. Of particular interest is the fact that the designers actually built the nozzles that appeared during the execution: the computers of the day wouldnÕt support an adequate nozzle simulation!

The genetic operators used in the execution had to be adapted to the discrete nature of the genotype. The mutation operator in particular could add or subtract segments from the nozzle, or vary the out-going diameter of any segment (by 2mm) at each generation of the execution.

Finally, the nozzle used in the initial population had the shape shown in Figure 1. The best nozzle that appeared is shown in Figure 2.

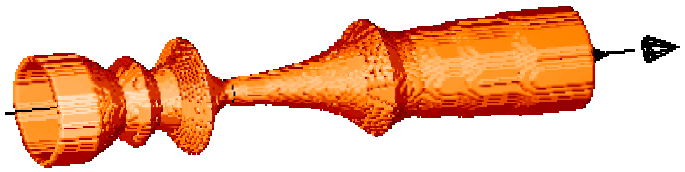


Figure 2. The resulting nozzle

Results

The resulting nozzle was found to be about 22% more efficient than the conical nozzle. Other experiments were also carried out using the identical algorithm with different fitness functions, resulting in designs optimised for thrust rather than efficiency.

Case Study 2: Heat Exchanger Network Design

Heat exchanger networks (HENs) are used widely in engineering plants due to their potential for large cost savings. HENs reduce the energy consumption of a plant by transferring heat between streams of fluids at different temperatures that would otherwise need to be heated or cooled via external sources, leading to energy cost savings. However, since using a HEN introduces additional capital and running costs, a balance between these costs and effectiveness of the HEN is required.

Figure 3 (originally described in [6]) depicts a four-stream HEN with two hot streams (denoted H_1 and H_2) and two cold streams (C_1 and C_2). A hot stream is one that needs to be cooled; similarly a cold stream is one that needs to be heated.

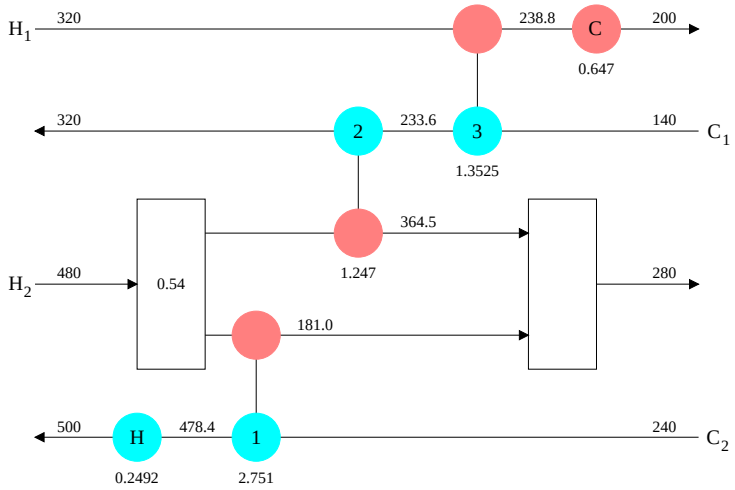


Figure 3. An example HEN

Arrows indicate the directions of flow of fluid streams, and numbers at the ends of a stream represent the incoming and target temperatures of the stream. The coloured circles labelled *H* and *C* represent external heaters and coolers. Numbered circle pairs represent heat exchangers acting between two fluid streams. The numbers directly below each heat exchange represents the enthalpy transfer occurring in the exchange (x 10^6). The large boxes represent splitters and mixers that split and combine streams respectively.

As described in [3], the design of a HEN comprises two tasks:

1. Design of the physical structure of the HEN, and,

2. Determination of HEN internal parameters (enthalpy transfer rates of heat exchangers and splitting ratios of splitters) to maximise the performance of the HEN.

Standard optimisation techniques are not feasible for completely automated HEN design. Traditionally, engineers design the structure of a HEN, which is then optimised using an algorithmic approach. This procedure does not necessarily find the best overall design. An automated tool to optimise both the HEN internal parameters and design the physical structure of the HEN is needed.

Methodology

We now revisit the steps in constructing an evolutionary algorithm approach to solve a problem. Firstly, in order to use an evolutionary algorithm approach, a representation must be determined that can encode possible HEN designs. Since a HEN can be thought of as a collection of heat-exchanging components connecting fluid streams, a graph representation seems most appropriate.

Pictured below is the corresponding graph representation of the HEN shown previously. Each node of the graph represents a component of the HEN with each edge representing a connection between two components. Associated with each node and edge are the corresponding physical data (e.g. heat-capacity flow rates, splitting ratios) of the component or connection. The heat exchangers are numbered *1* through *3* with the splitter and mixer labelled S_1 and M_1 respectively.

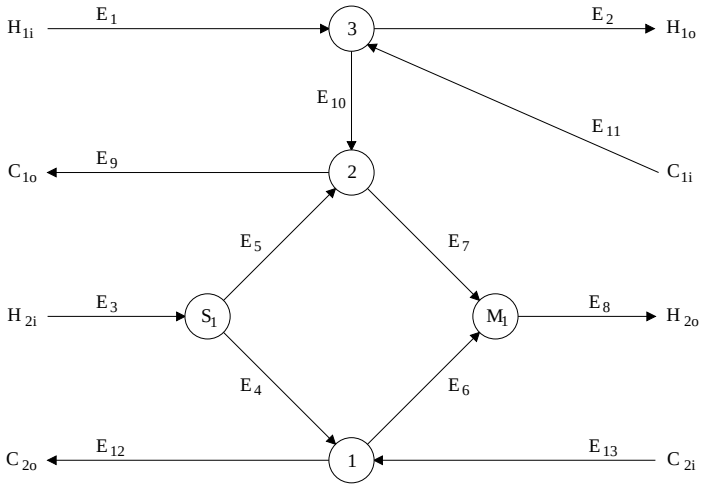


Figure 4. A graphical representation of a HEN

The fitness function for this problem measures the economic rewards of using the HEN. It consists of components measuring the effectiveness of the HEN, the investment cost of using the HEN, and the ongoing running costs of the HEN. The fitness function is highly geared towards overall cost minimisation.

A specialised mutation operator is required to modify existing HEN designs as both the internal HEN parameters and the structure of the HEN itself need to change under this operation. This is achieved by having the operator consist of two parts: the first modifies the real-valued internal parameters of the HEN with the second part accomplishing the structural changes to the graph representation. Adding and deleting nodes to and from the graph (with low probability) achieves structural changes to the graph.

The initial population consists of only empty HENs (i.e. HENs in which only external heaters and coolers are used). The population is subjected to the evolutionary process outlined previously. The population evolves to optimise the fitness function — in this case to minimise costs.

Results

Siemens AG is a worldwide engineering company interested in automated HEN design. As outlined here, an evolutionary algorithm approach to this problem yielded a system that outperformed all existing methodologies known to Siemens AG for optimising heat exchanger networks. Using this approach, Siemens AG realised savings in the order of US$100,000 - US$140,000 for each heat exchanger network designed [7].

Comparison with traditional optimisation methods

Automated optimisation of flowsheet design represents a severe challenge for traditional operations research/numerical approaches.

Many classical methods begin by posing the problem as finding the best values for a fixed number of unknown variables. But for flowsheet design, the numbers of units required is not known a-priori. So these methods can only be used to solve part of the whole problem, or an approximation, or a restricted version of the problem.

Numerical methods also usually rely on the problem being not too complicated. For example, some methods require the problem to be stated in terms of linear equations and inequalities. Some allow at most quadratic equations. Most require that the problem can be described in terms of functions that are differentiable. This is not the case for flowsheet design (for example, units are only available in certain sizes, so the problem is not even continuous).

Many existing methods are ÒlocalÓ optimisers, meaning that they can be fooled into settling for solutions that cannot be improved by making small changes, but are far from the true best solution (the global optimum). This is illustrated in the Figure 5, which is a plot of fitness values for a problem with two independent variables. The height of the surface represents fitness. There is no way to move from the local optimum at ÒAÓ to the true optimum at ÒBÓ via small hill-climbing steps as used in local optimisers. By contrast, evolutionary methods are true global optimisers (along with other methods like simulated annealing, tabu search, ant colony optimisation etc).

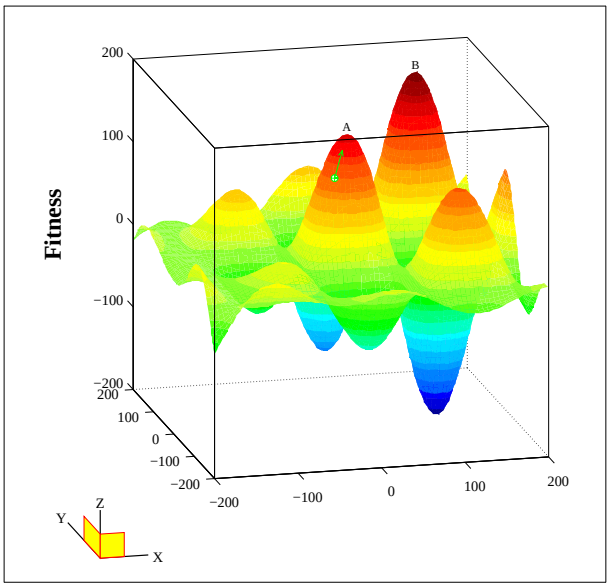


Figure 5 A fitness landscape with local optima

Because of these problems, numerical methods are usually used only to find good values for numerical parameters of a design (e.g. unit sizes) after the layout of the flowsheet has been designed manually.

Another type of artificial intelligence based approach is to attempt to embed engineering knowledge into rules for creating designs. The difficulty is in determining an appropriate set of rules. Such systems are notoriously ÒbrittleÓ — they work well in most situations but unexpectedly produce nonsensical results in some circumstances. Finding a set of rules that produces sensible designs in every situation is virtually impossible.

There are at least two means by which professional expertise may be incorporated in an evolutionary algorithm. For example, suppose that designs with certain combinations of features were deemed to be always undesirable. These designs can be avoided in either of the following ways.

- Directed mutation and crossover: the mutation and crossover operators could be defined to never create these designs.

- Fitness function: these designs could be assigned a poor evaluation in the fitness function.

These two approaches correspond roughly with the Òinfeasible vs. feasibleÓ discussion in Prince's method. A second kind of approach is to have the evolutionary algorithm produce

a small number of alternative designs, perhaps as the result of multi-objective optimisation, and for the engineer to make the final choice between the alternatives.

In summary, existing approaches suffer from a number of inherent faults that can be overcome by true evolutionary algorithms.

Suggested plan of action

We believe there is an opportunity to develop a flexible automated design tools in partnership with WA companies. This would provide industry with a cost-effective way to benefit from the expertise of researchers in the field, and would provide financial support and a real-world focus for the research and development work.

Our suggestion is to proceed with a staged approach, to manage the risk to industry partners. The first stage would be to develop a prototype demonstrating the feasibility of the technology, using a small set of test scenarios. Subject to a positive evaluation of the prototype, a second stage would be to develop a tool for use in the field.

If there is in-principle support for this concept, we would like to meet with company representatives to formulate a proposal for an initial prototype development project.

About the authors

Dr Philip Hingston

Philip was employed by Rio Tinto in research and development roles between 1995 and 2000. In this time, he participated in, carried out or managed a variety of projects ranging from the use of artificial neural networks in exploration, through mine planning and scheduling optimisation projects, to knowledge management initiatives. Outside of Rio Tinto, he has successfully carried out and supervised research in artificial neural networks, database and information retrieval, and evolutionary computation, as well as in other areas of artificial intelligence and computer science. Philip is currently a Senior˚Lecturer in Computer and Information Science at Edith Cowan University.

Dr Lyndon While

Lyndon started his career at INMOS Ltd. in the UK, working on the processor and memory architecture of the transputer. In 1992, he joined the Department of Computer Science & Software Engineering at The˚University of Western Australia (UWA). His recent research has been in applications of evolutionary algorithms, the design of parallel algorithms, and the implementation of declarative programming languages. He has published a number of papers and supervised several projects in the area of evolutionary computation. Lyndon is currently a Senior Lecturer in Computer Science & Software Engineering at UWA.

Dr Luigi Barone

Luigi recently completed his PhD at UWA in the area of evolutionary computation. He has twice received scholarships from IEEE to present this work at international conferences. Luigi is currently lecturing in Computer Science & Software Engineering at UWA.

Selected Publications in Evolutionary Computation

S. Huband, P. Hingston, R.L. While and L. Barone. An Evolution Strategy with Probabilistic Mutation for Multi-objective Optimisation. *To appear in 2003 Congress on Evolutionary Computation*, Canberra, December 2003.

L. Barone, L. While and P. Hingston. Designing Crushers with a Multi-objective Evolutionary Algorithm. *Proceedings of the Genetic and Evolution Computation Conference, GECCO-2002*, New York, July 2002.

P. Hingston, L. Barone and R.L. While. Evolving Crushers. *2002 Congress on Evolutionary Computation*, Honolulu, May 2002.

P. Hingston. A Genetic Algorithm for Regular Inference. *Proceedings of the Genetic and Evolution Computation Conference, GECCO-2001*, San Francisco, July 2001.

L. Barone and R.L. While. An Adaptive Learning Model for Simplified Poker Using Evolutionary Algorithms. *1999 Congress on Evolutionary Computation*, Washington DC, July 1999.

R. Salama and P.Hingston. Evolving Neural Network Controllers. *Proceedings of the IEEE International Conference on Evolutionary Computation*, Perth, December 1995.

L.Barone, P.Hingston and R.L.While. Computer Modelling of Evolution. *Proceedings of the 1994 Western Australian Computer Science Symposium*, September 1994.

References

[1] T. Back, J. Heisternmann, C. Kappler, and M. Zamparelli. Evolutionary Algorithms Support Refueling of Pressurized Water Reactors. *Proceedings of the Third IEEE Conference on Evolutionary Computation*, 104-108, IEEE Press, 1996.

[2] M. Groetzner, M. Schutz, P. Roosen, M. Emmerich, and B. Gross. A Mixed Integer Evolution Strategy for Chemical Plant Optimization. *Evolutionary Design and Manufacture*, 55-67, Springer, 2000.

[3] B. Gross, U. Hammel, P. Maldaner, A. Meyer, P. Roosen, and M. Schutz. Optimization of Heat Exchanger Networks by Means of Evolution Strategies. *Proceedings of the Sixth Conference on Parallel Problem Solving from Nature*, 1002-1011, Springer, 1996.

[4] I. Gupta, A. Gupta, and P. Khanna. Genetic Algorithm for Optimization of Water Distribution Systems, *Environmental Software*, 14(5): 437-446, Elsevier Science, 1999.

[5] J. Klockgether and H. Schwefel. Two-Phase Nozzle and Hollow Core Jet Experiments. *Proceedings of the 11th Symposium on Engineering Aspects of Magnetohydrodynamics*, 141-148, 1970.

[6] K. F. Lee, A. H. Masso, and D. F. Rudd. Branch and Bound Synthesis of Integrated Process Designs. *I&EC Fundamentals*, 9(1): 48-58, 1970.

[7] Siemens AG - Optimization of Chemical Engineering Plants (CEPs). *Nutech Solutions Case Study*. As of: 20 March 2001. Available from: http://www.nutechsolutions.com

[8] R.G.H. Prince. Computer-aided design of mineral processing flowsheets, AMIRA Project P452, December 1997.

[9] Kraftwerksunion (KWU) - Optimization of Fuel Reload Designs for a Nuclear Reactor. *Nutech Solutions Case Study*. As of: 20 March 2001. Available from: http://www.nutechsolutions.com

[10] Case Study - Development of an Optimization System for Designing Networks. *Nutech Solutions Case Study*. As of: 20 March 2001. Available from: http://www.nutechsolutions.com

[11] Optimisation of a 2-phase nozzle with an ES. From the Evo-Net website as of: 20 March 2001. Available as: http://ls11-www.informatik.uni-dortmund.de/people/kursawe/Demos/Duese/dueseGIFE.html