

HYBRID METHODS FOR OPTIMISATION

E. S. FRAGA

*Centre for Process Systems Engineering, Department of Chemical Engineering
University College London (UCL), London WC1E 7JE, United Kingdom*

Computer aided design tools for industrial engineering typically require the use of optimisation. The optimisation problems in industrial engineering are often difficult due to the use of nonlinear and nonconvex models combined with underlying combinatorial features. The result is that no single optimisation procedure is typically suitable for most design tasks. Hybrid procedures are able to make use of the best features of any method while ameliorating the impact of the disadvantages of each method involved. This paper presents an overview of hybrid methods in engineering design. A simple case study is used to illustrate one hybrid optimisation procedure.

1. Introduction

Computers are used in industrial engineering throughout the whole life cycle. At the early stages of the cycle, computer aided design tools are used to identify good or promising design alternatives. Subsequently, further tools are used to refine these alternatives using more complex models as information becomes available and issues must be addressed. The earlier issues can be addressed, the greater the likelihood that the final design generated meets the criteria imposed on it (economic, environmental, societal). Therefore, there is constant pressure to have as complex a model as possible for the design problem under consideration as early as possible.

This constant pressure is resisted by the need for more powerful and capable optimisation tools to handle the increased complexity. Optimisation forms the core of many computer aided engineering tools. The types of optimisation models used in industrial engineering range from linear programming through to mixed integer differential/integral nonlinear programming. Generic technologies have been developed for most classes of optimisation problems with varying success. Commercial software is available, including, for instance, the set of solvers available through the NEOS server.¹

2. Hybrid Methods for Optimisation

Although there has been significant progress in the development of generic solvers, many problems in industrial engineering cannot be handled with these solvers. Models in industrial engineering, especially those in the processing industries, often exhibit nonlinear, nonconvex and discontinuous behaviour. Furthermore, the models may pose inherent numerical difficulties for computational tools due to behaviour in the limits of the domains of the variables (e.g. the *log*-mean temperature difference equation in heat exchanger design) or are valid only in a restricted domain and may be meaningless outside that domain (e.g. mole fractions). In some cases, models may also exhibit noise (e.g. due to online experimental measurements as part of the models used).

Therefore, for many industrial engineering applications, targeted optimisation procedures are developed. These targeted procedures are often based on stochastic methods, including evolutionary programming methods, such as genetic algorithms,² and simulated annealing.³ The appeal of this class of methods is their ease of implementation and their robustness with respect to the issues mentioned above, making them suitable for use by non-experts in the area of optimisation. Their greatest disadvantage, however, is the number of parameters that require setting and for which values are often difficult to ascertain based purely on the problem considered.

Although these stochastic methods can be successful in identifying good solutions, they often do not achieve the best solutions possible and also do not necessarily provide any insight into how far from the best the solutions obtained may be. The advantage of the more traditional, mathematical programming, approaches is that they can address some of these issues. Therefore, one reasonable approach is to consider the development of *hybrid* procedures that combine the best attributes of these classes of methods.

Hybrid methods are so called because they combine one or more methods to work together in solving a given problem:

Hybrid (*Hy"brid*), a.

derived by a mixture of characteristics from two distinctly different sources;⁴

There are two ways to combine two or more methods: sequential or embedded. Examples of both are presented in what follows.

3. Embedded Hybrid Methods

In an embedded method, an outer procedure is used to determine the values of all the decision variables or, possibly, a subset of these variables. An inner procedure is then invoked with the values determined by the outer procedure either to determine the values of the remaining decision variables or to refine the values of all the decision variables determined by the outer procedure. Once the inner procedure has completed, the outer procedure is given control again and another iteration performed until the appropriate stopping criterion is met.

The simplest example of an embedded hybrid method appears in the various modifications to the conjugate gradient method for handling the line search. The outer method determines a search direction and the inner procedure manipulates the decision variables subject to remaining on a line defined by the search direction. However, this example is arguably not a hybrid procedure in that the outer procedure does not actually define any values for the decision variables.

In computer science applications, a large number of what are known as *neighbourhood search* or *local search* algorithms have been developed. These are typically a combination of a backtracking algorithm used to search through a graph based representation of the solution space with a local embedded search procedure used to determine the best alternative path to choose at any point in the forward traversal. See Ahuja *et al.*⁵ for a general survey of these types of methods.

Shouraki & Haffari⁶ describe their experiences with different local search algorithms within the STAGE procedure for tackling combinatorial problems. STAGE combines local search methods with backtracking procedures, preserving the scalability of the local search methods while aiming for the exhaustive properties of backtracking methods. Prestwich⁷ describes the Incomplete Dynamic Backtracking method which combines local search with backtracking so as to preserve the advantages of both approaches without losing the scalability of the local search methods. More recently, van Hentenryck & Michel⁸ present a formulation for describing hybrid search procedures based on backtracking and local or neighbourhood search methods.

A more general embedded hybrid optimisation approach is the incorporation of local search or refinement techniques within a stochastic global optimisation procedure. Locatelli & Schoen⁹ describe formally how a local search algorithm affects a global optimisation procedure. They apply

such a method to the minimisation of potential energy as modelled by the Lennard-Jones equation.

Frequently, the stochastic optimisation procedure is a genetic algorithm² (GA) or simulated annealing³ (SA). Some examples of such approaches are described in the remainder of this section.

Thomsen¹⁰ describes the effects of incorporating a local search within a genetic algorithm to define a *Lamarckian* GA. A Lamarckian GA is one in which population members can be modified by a local search procedure (cf. the distinction between Lamarckian and Darwinian evolution¹¹). Three approaches are compared: one without any local search, one where the current best solution is refined and one where a randomly chosen solution from the current population is chosen.

Ganesh & Punniyamoorthy¹² describe a combined GA and SA procedure where, at the end of each generation of a genetic algorithm, each member of the current population is used as an initial guess for a simulated annealing procedure. The results of all the SA applications are used to define a new population (using standard selection procedures in the GA). A similar approach was used by Ponnambalam & Reddy¹³ for integrating lot sizing and sequencing in flow-line scheduling.

The local search procedure need not be deterministic. For instance, Tulpan & Hoos¹⁴ present a stochastic local search method based on a random walk procedure which has been extended with a local search procedure which resolves conflicts (i.e. constraint violations). They have applied this method to the DNA code design problem.

Theos *et al.*¹⁵ describe the PANMIN program which is based on two stochastic global optimisation methods which use local searches as intermediate steps and for refinement of solutions. One of the methods is similar to a controlled random search.¹⁶ The second method implements a topographical multilevel single linkage approach. This has some similarity to a controlled random search but with memory. The method also uses a Bayesian¹⁷ statistical method to provide a stopping criteria by estimating the number of global minimisers in the domain. The local search, which forms part of the core algorithm, uses the Merlin system¹⁸ which provides links to a number of local search methods include both direct search and gradient based methods.

Alternative methods for the outer global optimisation procedures have also been developed. Smyth *et al.*¹⁹ combine a tabu search with iterated local search. Jussien & Lhomme²⁰ also combine a tabu search with a local search procedure, this time a search over partial assignments, instead of

complete assignments, for open-shop scheduling problems.

Recently, another stochastic approach derived from observation of biology, based on analogies with ant colonies or particle swarms, has been investigated. For hybrid methods, Meyer & Ernst²¹ combine an *ant colony optimisation* (ACO) model with constraint propagation to tackle problems with hard constraints that would otherwise be inappropriate for ant colony models. Lee & Lee²² combine GA, ACO and heuristics to solve resource allocation models.

The commonality of all these methods is the combination of an outer global optimisation procedure with a targeted local search method. The aim is to enhance the convergence of the outer procedure using the fine tuning capabilities of local search methods. Without this tuning, many of the global optimisation methods used may converge to the global optimum in theory but in practice achieve less spectacular results.

Before continuing on to the other form of hybrid optimisation, it is worth noting that not all embedded approaches embed a local search method within a global optimisation procedure. In fact, Fraga & Žilinskas²³ present a family of embedded hybrid methods for the optimal design of heat-integrated process flowsheets in which the outer method is a direct search local optimisation procedure and the embedded method is a genetic algorithm. This particular combination is chosen because of the decomposition used for the process model. The outer procedure handles the NLP aspects whereas the inner procedure takes care of the combinatorial elements. The particular combination is shown to be highly effective and efficient.

4. Sequential Hybrid Methods

The procedures presented in the previous section demonstrate the wide range of applicability of the embedded form of hybrid optimisation. However, the alternative approach for combining optimisation procedures is more straightforward and can still achieve significant improvements over the use of a single method. In a sequential approach, one method is applied and a solution, or possibly a set of solutions, is generated. This solution or set of solutions is then used as the initial guess for a subsequent method. The solution from the second step can be fed into yet another method or back into the first method, forming the basis of an iterative procedure. These sequential hybrid methods are also known as *multi-start* algorithms²⁴ although, for some authors, multi-start methods imply a single method with multiple attempts using different initial guesses.

In principle, any combination of methods can be used. For instance, very recent work by Xia & Wu²⁵ presents a sequential hybrid procedure using a particle swarm optimisation (PSO) method to initialise a simulated annealing procedure. Fraga & Papageorgiou²⁶ use an interval analysis based stochastic procedure to provide feasible or close to feasible initial solutions for the following mathematical programming stage for the design and optimisation of water distribution networks.

Instead of attempting to enumerate further even a small number of such approaches, the rest of this paper is devoted to a simple case study which demonstrates the potential benefits of using sequential hybrid methods.

5. Illustrative Case Study

A process plant will typically have large cooling and heating demands. For instance, a popular technology for separating liquid mixtures is distillation. A distillation unit operates by boiling liquid at the bottom of the unit and condensing vapour at the top. Meeting the heating and cooling requirements can involve large amounts of utilities, such as steam and cooling water. Besides the obvious economic impact, there are also significant environmental issues from utility consumption. Therefore, it is beneficial to reduce utility consumption whenever possible.

Utility consumption can be reduced by using excess heat in one part of a process plant to meet the heating requirements elsewhere in the same process plant, subject to the laws of thermodynamics. Using heat in this way is known as process integration. Identifying the optimum integration between all the processing units in a process plant is known as the *heat exchanger network synthesis* (HENS) problem.

The definition of a HENS problem is a set of cold streams, a set of hot streams and the set of utilities available for meeting any cooling and heating demands not satisfied by integration. Mathematically, the aim is to minimise, for instance, an annualised cost for meeting the heating and cooling requirements of a process plant taking into account not only the utility consumption but also the cost of equipment.

As an optimisation problem, all possible integrations must be considered. This is a combinatorial problem and is particularly challenging when we allow for streams to be split so that, for instance, a hot stream may exchange heat with two cold streams in parallel. Previous attempts at solving the full heat exchanger network synthesis problem with stream splitting have been based on the *a priori* definition of a superstructure.^{27,28}

For larger problems, an efficient superstructure can be difficult to generate. By efficient, in this case, we mean a superstructure that contains hopefully all solutions of interest with minimal coverage of solutions that are less likely to be good. A tighter superstructure will lead to easier to solve optimisation problems, in some cases making the difference between a problem which is solvable and one which is intractable. Recently, with this aim, we have developed a multiple ant colony model approach for identifying a suitable superstructure as the first step in a multi-step sequential hybrid optimisation method.²⁹ In what follows, we illustrate the hybrid procedure used to solve the nonlinear programme defined by the superstructure generated by this ant colony method.

Table 1. Heat exchanger network synthesis case study.

Stream	T_{in} ($^{\circ}C$)	Process Streams		
		T_{out} ($^{\circ}C$)	Q (kW)	h ($\frac{kW}{K \cdot m^2}$)
H1	200	40	6400	0.8
H2	120	60	600	0.8
H3	90	50	200	0.8
C1	25	180	3100	1.6
C2	80	210	3250	1.6
C3	35	160	2250	1.6

Type	T_{in} ($^{\circ}C$)	Utilities		
		T_{in} ($^{\circ}C$)	h ($\frac{kW}{K \cdot m^2}$)	c_u ($\frac{\pounds}{kW \cdot y}$)
Steam	220	219	1.6	700
Water	30	40	0.8	60

Note: Q is the amount of heating or cooling required for each stream, h is the heat transfer coefficient for each process stream and each utility, and c_u is the cost of each utility.

The problem we consider is a generalisation of the stream splitting case study presented by Morton,³⁰ shown in Table 1. The resulting superstructure identified by the ACO step,²⁹ and which forms the basis of the subsequent optimisation steps, is shown in Fig. 1.

The nonlinear programming model has 13 continuous variables: 7 heat exchanger duties and 6 split fractions. Heat exchanger duties are represented by X_{mn} in Fig. 1, where m indicates the cold stream index and n

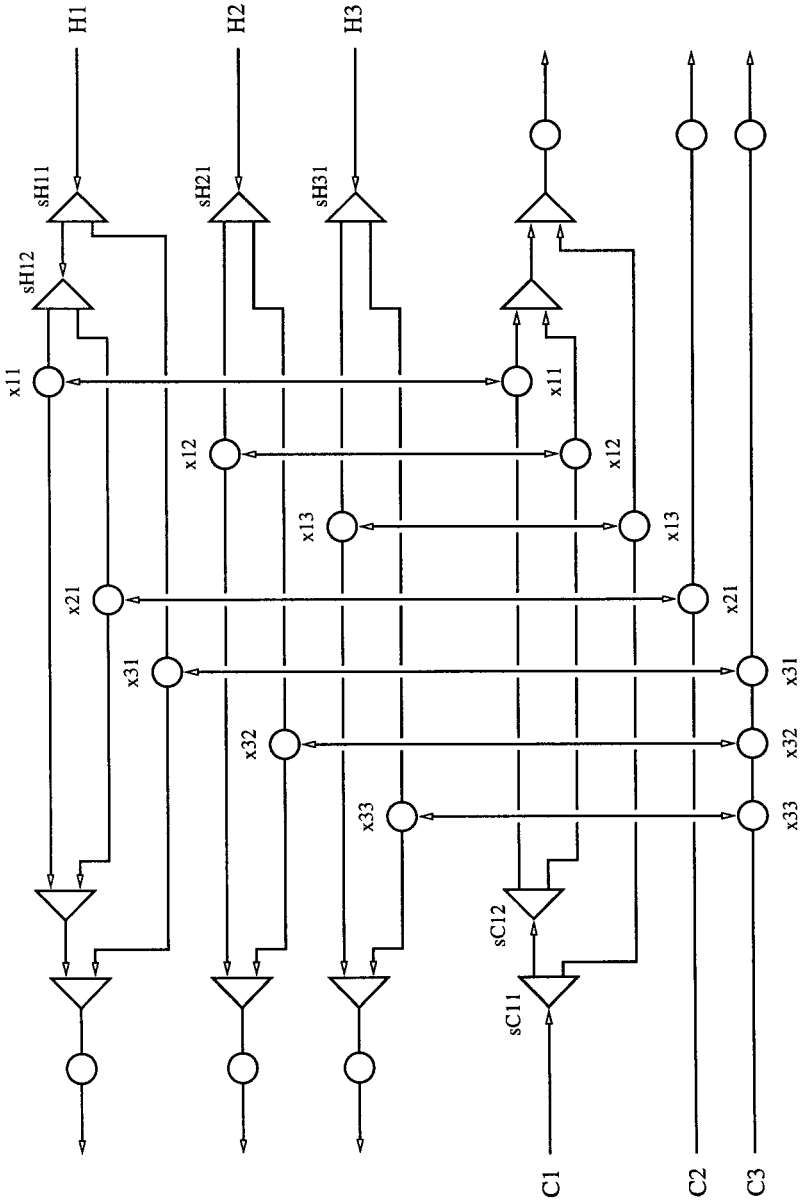


Figure 1. Superstructure for Morton case study obtained using an ant colony optimisation procedure.

the hot stream index. The split fractions are represented by $sHab$ for hot streams and $sCab$ for cold streams, where a is the index of the hot or cold stream and b is a counter to ensure unique labels for these splitters. All exchange variables are normalised so that all the variables take values $\in [0, 1]$.

The exchange variables represent the amount of exchange as a fraction of the maximum possible for that particular match. For a given match, the maximum possible is the minimum of the amounts available on each stream involved. The amounts available depend on the values of the split fractions. For instance, the match between cold stream C2 and hot stream H1, indicated by $x21$ in the superstructure, would have a maximum amount Q_{\max} defined as:

$$Q_{\max} = \min \{Q_{C2}, sH11 \times (1 - sH12) \times Q_{H1}\} \quad (1)$$

where $Q_{C2} = 3250$ kW and $Q_{H1} = 600$ kW, as given in Table 1, and where it is assumed that the split fraction variables specify what proportion of the duty goes to the upper stream as shown in the superstructure diagram.

This formulation is used to provide a feasible search space which is large in comparison with the full domain $\underline{x} \in [0, 1]^{13}$. Having the feasible space as large as possible with respect to the variable bounds makes searching more effective for stochastic methods. A second advantage of this formulation is that the point $\underline{x} = \underline{0}$ is a feasible point, one which corresponds to the non-integrated solution and which is a good starting point for any search from an engineering perspective.

The procedure for the solution of this NLP is shown in Algorithm 1. This procedure iteratively applies a number of direct search methods, specifically the Hooke & Jeeves, Implicit Filtering and BFGS methods from Kelley,³¹ followed by two stochastic methods, a genetic algorithm and a simulated annealing procedure. The direct search methods are applied in parallel and the best of the results used as the initial population, size 1, for the genetic algorithm. The best solution obtained by the GA is used as an initial guess to a simulated annealing procedure. Finally, the best solution obtained by the SA is presented to the engineer for direct fine tuning, if desired. The result is then compared with the initial starting guess before the direct search methods were applied and convergence determined.

This iterative multi-start procedure uses direct search for fine tuning and the two stochastic methods for global searching. The results obtained for this case study are shown in Table 2 with the final heat exchanger network presented in Fig. 2 with the exchangers annotated with the actual

Algorithm 1 Multi-start hybrid procedure for solving the heat exchanger network synthesis problem.

```

1: Let  $\underline{x} \leftarrow \underline{0}$  ▷ Guaranteed feasible point
2: converged  $\leftarrow$  false
3: while not converged do
4:    $\underline{x}^{(1)} \leftarrow \underline{x}$ 
5:   improved  $\leftarrow$  true ▷ For direct search only
6:   while improved do
7:      $\underline{x}^{(2)} \leftarrow$  best solution from direct search methods applied to  $\underline{x}^{(1)}$ 
8:     improved  $\leftarrow$   $\|\underline{x}^{(2)} - \underline{x}^{(1)}\| > \epsilon$ 
9:      $\underline{x}^{(1)} \leftarrow \underline{x}^{(2)}$ 
10:  end while
11:   $\underline{x}^{(3)} \leftarrow$  GA : initial population =  $\{\underline{x}^{(2)}\}$ 
12:   $\underline{x}^{(4)} \leftarrow$  SA :  $\underline{x}^{(3)}$ 
13:   $\underline{x}^{(5)} \leftarrow$  engineer interaction!
14:  converged  $\leftarrow$   $\|\underline{x} - \underline{x}^{(5)}\| < \epsilon$ 
15:   $\underline{x} \leftarrow \underline{x}^{(5)}$  ▷ New iterate
16: end while

```

amounts exchanged, in kW.

The table shows how the values of each of the decision variables evolves during the iterative process. All the methods contribute to the improvement of the objective function at some point in the iteration and user interaction is used twice. User interaction is particularly useful for forcing variables to their bounds. The values changed by the user are indicated in bold in the table and without any trailing numbers after the decimal point. Any values before user interaction that appear to be at one of the bounds (e.g. “1.000”) are in fact very close to the bound but not exactly on the bound.

Although it is often possible to design and implement optimisation procedures that attempt to push variables to bounds, this is difficult to do in a generic manner. The user, however, can often tell whether it is reasonable for a variable to actually be at the bound and, in this case, this proves to be true.

It is useful to analyse the evolution of a selection of the variables. Figure 3 shows graphically how these variables evolve. Specifically, we see that **x32** and **sh31** swing from one end of their domain to the other when the stochastic optimisation procedures are invoked, demonstrating the global search capabilities of these methods. We also see that variable **sh12** evolves

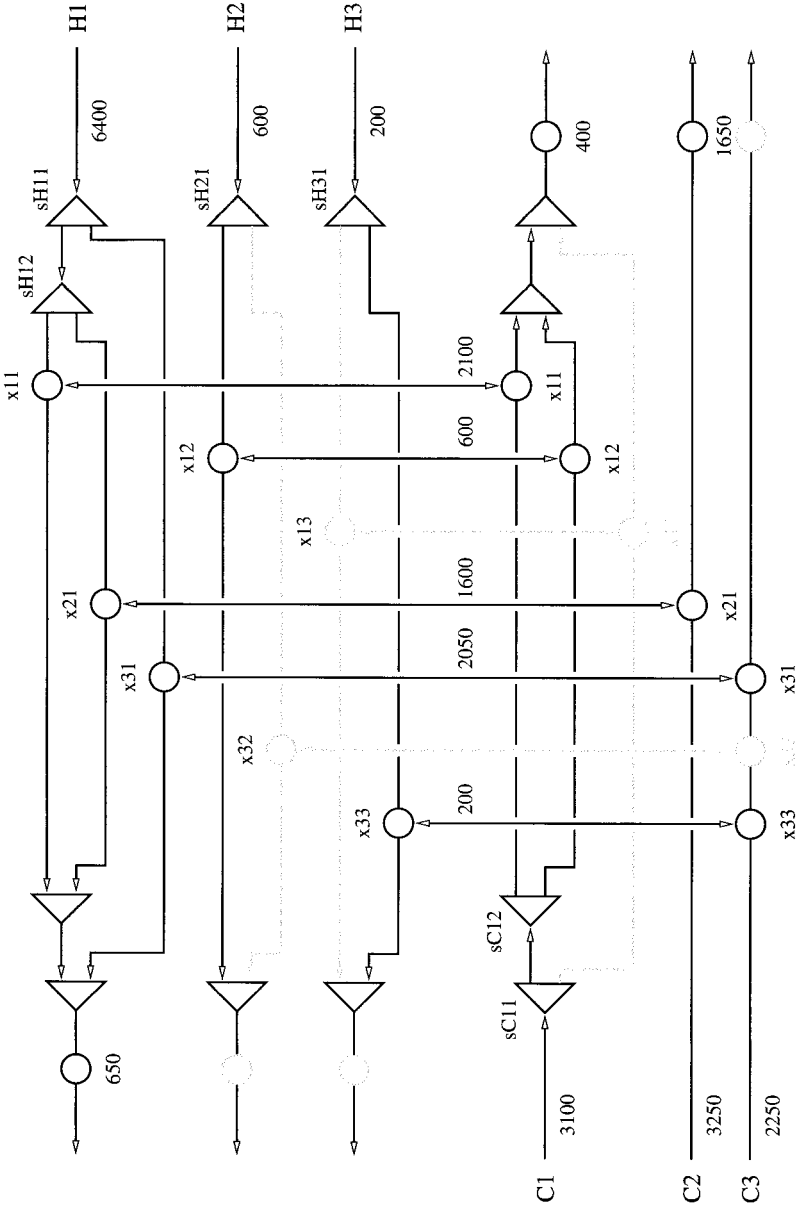


Figure 2. Final heat exchanger network design obtained for the case study. The elements of the original superstructure which are not present in the final network have been shown in light grey.

Table 2. Evolution of best solution during the application of the hybrid multi-start algorithm.

x_i	Step							
	Initial	IF	HJ	BFGS	GA+SA	UI	HJ	UI
x11	0.	.807	.998	1.000	1.000	1.	1.	1.
x12	0.	.697	.998	.999	1.000	1.	1.	1.
x13	0.	.695	.999	1.000	.998	1.	1.	1.
x21	0.	.806	.744	.744	.745	.745	.745	.745
x31	0.	.962	.998	.999	1.000	1.	1.	1.
x32	0.	.695	.999	.999	.320	.320	.320	.320
x33	0.	.694	.998	.998	1.000	1.	1.	1.
sH11	0.	.688	.649	.649	.663	.663	.663	.663
sH12	0.	.193	.373	.374	.489	.489	.493	.493
sH21	0.	.695	.999	1.000	1.000	1.	1.	1.
sH31	0.	.694	.999	.999	.039	.039	.004	0.
sC12	0.	.693	.599	.600	.674	.674	.678	.678
sC11	0.	.693	.834	.834	.994	1.	1.	1.
$f(\underline{x})$	3.306	1.211	.914	.912	.908	.893	.892	.888

slowly to its final value, indicating the contribution made by several methods in combination.

6. Discussion

This paper has presented a brief overview of hybrid methods for optimisation. The methods have been placed into two categories, embedded and sequential. A key distinction between the two is that the former tend to be defined and implemented for specific problems or problem areas whereas the latter can make use of generic implementations.

A case study on the design of a heat exchanger network has been presented to demonstrate the power of a hybrid optimisation procedure. In this case, a multi-start, or sequential, procedure has been used. This procedure combines both stochastic and deterministic methods and the case study shows the contribution made by both types of methods. The case study also demonstrates the potential benefit of including the engineer in the iterative procedure.

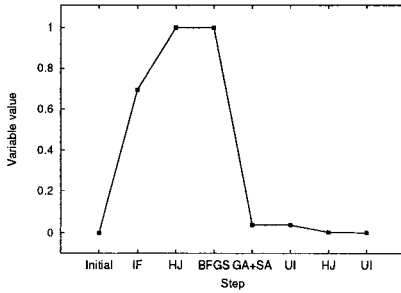
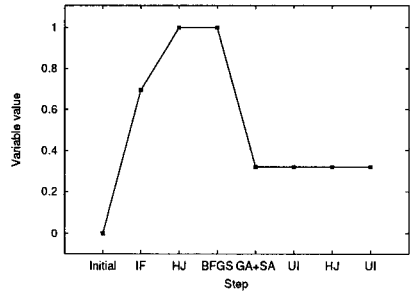
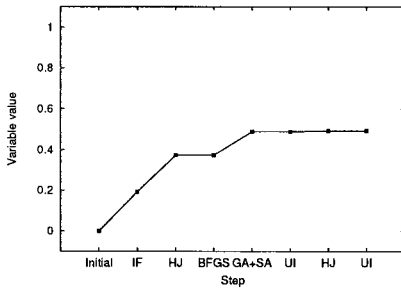
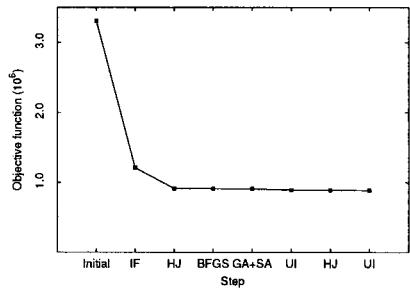
(a) x_{32} (b) sH_{31} (c) sH_{12} (d) $f(\underline{x})$

Figure 3. Evolution of a selection of decision variables and objective function for the case study.

References

1. J. Czyzyk, M. P. Mesnier and J. J. Moré, *IEEE Computing in Science and Engineering* **5**(3), 68–75 (1998).
2. J. E. Smith, In: P. M. Pardalos and H. E. Romeijn (eds.), *Handbook of Global Optimization Volume 2*, Kluwer Academic Publishers, 275–362 (2002).
3. P. J. M. van Laarhoven and E. H. L. Aarto, *Simulated Annealing: Theory and applications*. Kluwer Academic Publishers, Dordrecht (1987).
4. Published on the Internet. <ftp://ftp.gnu.org/gnu/gcide/>.
5. Ahuja, R. K., Ö. Ergun, J. B. Orlin and A. P. Punnen, *Discrete Applied Mathematics* **123**(1-3), 75–102 (2002).
6. S. B. Shouraki and G. Haffari, *Lecture Notes in Computer Science* **2510**, 102–109 (2002).
7. S. Prestwich, *Annals of Operations Research* **115**(1-4), 51–72 (2002).

8. P. van Hentenryck and L. Michel, *Lecture Notes in Computer Science* **3524**, 380–395 (2005).
9. M. Locatelli and F. Schoen, *Computational Optimization and Applications* **26**(2), 173–190 (2003).
10. R. Thomsen, *Biosystems* **72**(1-2), 57–73 (2003).
11. <http://en.wikipedia.org/wiki/Lamarckism>.
12. K. Ganesh and M. Punniyamoorthy, *International Journal of Advanced Manufacturing Technology* **26**(1-2), 148–154 (2005).
13. S. Ponnambalam and M. Reddy, *The International Journal of Advanced Manufacturing Technology* **21**(2), 126–137 (2003).
14. D. C. Tulpan and H. H. Hoos, *Lecture Notes in Computer Science* **2671**, 418–433 (2003).
15. F. V. Theos, I. E. Lagaris and D. G. Papageorgiou, *Computer Physics Communications* **159**(1), 63–69 (2004).
16. W. L. Price, *Journal of Optimization Theory and Applications* **40**(3), 333–348 (1983).
17. J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*. John Wiley (2000).
18. D. G. Papageorgiou, I. N. Demetropoulos and I. E. Lagaris, *Computer Physics Communications* **109**(2-3), 227–249 (1998).
19. K. Smyth, H. H. Hoos and T. Stützle, *Lecture Notes in Computer Science* **2671**, 129–144 (2003).
20. N. Jussien and O. Lhomme, *Artificial Intelligence* **138**(1), 21–45 (2002).
21. B. Meyer and A. Ernst, *Lecture Notes in Computer Science* **3172**, 166–177 (2004).
22. Z.-J. Lee and C.-Y. Lee, *Information Sciences* **173**(1-3), 155–167 (2005).
23. E. S. Fraga and A. Žilinskas, *Advances in Engineering Software* **34**, 73–86 (2003).
24. E. C. Laskari, K. E. Parsopoulos and M. N. Vrahatis, *Numerical Algorithms* **34**(2-4), 393–403 (2003).
25. W. Xia and Z. Wu, *International Journal of Advanced Manufacturing Technology*, in press (2006).
26. E. S. Fraga, L. G. Papageorgiou and R. Sharma, In: A. Kraslawski and I. Turunen (eds.), *European Symposium on Computer Aided Process Engineering – 13*, Elsevier Science B.V., Amsterdam, *Computer-Aided Chemical Engineering* **14**, 119–124 (2003).
27. J. Aaltola, *Applied Thermal Engineering* **22**(8), 907–918 (2002).
28. G. F. Wei, P. J. Yao, X. Luo and W. Roetzel, *Chinese Journal of Chemical Engineering* **12**(1), 66–77 (2004).
29. G. W. A. Rowe and E. S. Fraga, In: I. C. Parmee (ed.), *Proceedings of Adaptive Computing in Design and Manufacture, ACDM'2006*, in press (2006).
30. W. Morton, *Proc. Inst. Mech. Eng. Part E* **216**(2), 89–104 (2002).
31. C. T. Kelley, *Iterative Methods for Optimization*. SIAM (1999).