J Glob Optim (2009) 45:169–185
DOI 10.1007/s10898-009-9401-7

Modular global optimisation in chemical engineering

S. Balendra · I. D. L. Bogle

Received: 8 January 2009 / Accepted: 19 January 2009 / Published online: 5 February 2009

Abstract Chemical Engineering design and analysis is dominated by the use of modular computational systems restricting the use of rigorous global optimisation techniques. Other engineering domains also exploit modularity in order to break down complex tasks to allow the use of legacy codes, to protect intellectual property, and to allow large teams to work on problems. By casting modules in a generic form such systems could be recast to incorporate interval based methods. In this paper we explore the use of five interval contraction methods to improve the performance of interval based optimization of modular process design systems: consistency methods, constraint propagation, Interval Gaussian elimination, Interval Newton, and Linear Programming. It is shown that the Linear Programming contractor provides the greatest value in contracting the intervals and that constraint propagation and Interval Gaussian elimination (as implemented here) provides less of an impact. Other contractors do provide value and the LP contractor will be of less value as the problem size increases so it is necessary to include a number of contractors which can be done at small computational cost. A number of challenges are outlined which need to be addressed before there can be routine use of interval global optimization in modular systems.

Keywords Chemical engineering design · Interval based optimisation · Modular systems · Interval contractors · Pooling problem

S. Balendra · I. D. L. Bogle (✉)
Department of Chemical Engineering, Centre for Process Systems Engineering,
University College London, London WC1E 7JE, UK
e-mail: d.bogle@ucl.ac.uk

Present Address:
S. Balendra
MW Kellogg, Greenford Road, Middlesex UB6 0JA, UK

1 Modular chemical process design

As in all branches of Engineering, optimisation of new designs and of the operation of manufacturing plant is critical for a competitive business. In the field of Chemical Engineering this has become more clearly codified in the computational systems used for design and for management of operations. Most design systems now have the capability to optimise the design, almost always using gradient based methods, and the control systems on process plant have on-line optimisers, again using gradient based methods, which try and ensure that the operations are running in the most efficient manner. This capability is now used routinely [8].

The need for process optimisation is becoming more widespread as the manufacturing objectives become more complex. Financial efficiency, however defined, is no longer the only optimisation criterion used. Safety is considered to be the number one priority for any design of new plant or modification of existing plant. Now increasingly the environmental performance of the plant is becoming a key issue in its acceptance by the regulatory authorities for permission to operate. Of course this comes from increasing pressure from society in general. This often results in a trade off between economic performance and environmental performance but both can be formulated as optimisation problems.

A typical design problem is to find an optimal solution as defined by a quantified objective function based on cost, environmental performance, or safety. A chosen collection of variables will be selected as design variables and a solution found which maximizes the given objective function. Inequality constraints may also be imposed on the optimization problem. For example, a lower limit may be placed on the purity of a stream. It may also be necessary to add equations to evaluate the objective function, such as cost correlations, to the system of equations.

One of the big technical issues restricting the effectiveness of process optimisation is that the process models are frequently non-convex. These non-convexities arise from combinations of nonlinear functions and from discontinuities in the describing equations and their derivatives. This of course means that the likelihood of finding a local minimum is very high. Since all the problems of any significance are large it becomes impossible to exploit specific mathematical features of a particular problem to avoid finding local minima and so a systematic means of finding genuinely global minima, particularly when there are significant differences between local and global minima, becomes very important.

Reviews of global optimisation methods have been provided by Torn and Zilinskas [34] and Horst and Tuy [21]. There has recently been increasing interest in deterministic and stochastic global optimisation methods in the Chemical Engineering literature. Floudas [10] gave a recent review of the use of the development of global optimisation techniques for process engineering problems and many contributions have been gathered in Grossmann [15]. Floudas and Visweswaran [12] presented a 'primal-dual' decomposition approach based on the Generalised Benders Decomposition. Vaidyanathan and El-Halwagi [35] and Byrne and Bogle [6] have proposed modifications to interval methods [16,27]. Ryoo and Sahinidis [31] used convex underestimators to solve a large number of engineering problems. Adjiman et al. [1] present a global optimisation method for solving general chemical process design problems. Quesada and Grossmann [29] developed specialised convex underestimators for solving a class of heat exchanger network problems and bilinear mass transport problems. Garrard and Fraga [13] use stochastic methods for seeking the global optimum of a synthesis problem.

The computational systems for designing or analysing such plants have mostly been developed to mimic flowsheet structure. Individual modules or procedures have been developed for each unit and are connected together in an appropriate calculation order. The recycle is handled iteratively and there are computational strategies for handling specifications placed on streams other than inputs, such as a desired product quality. These systems are called Sequential Modular systems [3,32]. More recently Equation Oriented systems have appeared commercially where the equation set is assembled and solved simultaneously [28]. A thorough comparison is provided by Biegler et al. [4]. The second class has greater flexibility in that a wider range of numerical codes can be used and the extension to dynamical systems is straightforward. However, the natural structure of the problem is lost during the solution phase and this causes difficulties in trouble shooting when problems arise in finding the solution. Sequential Modular systems dominate the market.

The Sequential Modular approach to flowsheeting links together sequences of models, implemented as computational procedures or modules, for the unit operations and calculations follow this sequence. Recycles and design specifications are handled by a number of alternative methods. Modularity is one of the oldest of software engineering paradigms. It has a number of advantages: it aids understanding by presenting a system in distinct functional chunks, modules are reusable, and modifications or a replacement can be made to one module without affecting others.

The procedures can be broken down into two types.

Algebraic procedures: A procedure P computes its outputs, y, by solving a set of nonlinear algebraic equations where some variables, w, are internal to the procedure itself, i.e. they do not occur in the mathematical optimisation problem. The equations are solved to determine y and w for given values of the input variables, u, and the values of w are discarded while those of y are returned as the outputs of the procedure, effectively establishing the required mapping:

$$y = P(u).$$

An example of an algebraic procedure is a computational module that describes the steady-state behaviour of a distillation column. In this case, the input variables u could be associated with the properties of the feed stream(s) to the column, the reflux ratio and the boil-up rate; the output variables y would include the properties of the column output streams; and the internal variables w would comprise all the variables describing the composition, temperature and flowrates internally to the column.

General procedures: Procedures of this type compute their outputs by executing a sequence of computational instructions which may involve not only the evaluation of simple numerical expressions but also iterative loops, conditional statements and so on. Such procedures typically take the form of Fortran subroutines or equivalent constructs in other procedural computer languages. An example is provided by physical property prediction, e.g. the specific enthalpy of a mixture is the output of a subroutine which computes its value from given values of the mixture temperature, pressure and composition as inputs. These often involve internal iterations and are usually provided as a separate program from the simulator but linked computationally.

Previous work has applied global optimisation techniques to problems where the equations are explicit. Here we have addressed the problem of applying global optimisation using interval methods to modular approaches to chemical process design, often called process flowsheeting. Interval analysis has also been applied to many applications in chemical engineering [25,33] but the focus here will be towards solving modular flowsheet design problems. We have explored the basic premise [7] but many challenges remain.

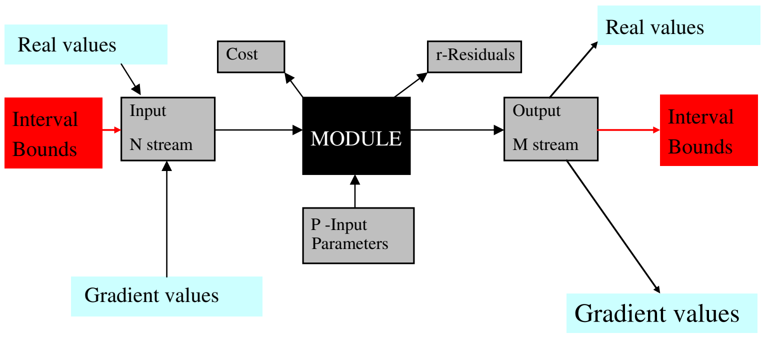

Fig. 1 A Generic module allowing multiple types

2 Applying interval analysis to modular flowsheets

Byrne and Bogle [7] showed that the sequential modular approach can be formulated in a way to take advantage of interval methods. Modules are connected in the same way but can be modified to handle interval arithmetic. A generic module can be formulated to allow point values or intervals for all the input streams and unit parameters. The module calculates the conditions for the output streams as either values or as intervals. Modular flowsheets are constructed with generic unit modules that can provide the interval bounds, linear bounds, derivatives and derivative bounds using extended arithmetic types. An extended arithmetic type is a compound type together with a set of rules that define operations on it [27]. In this case when an interval compound is used according to the rules of interval arithmetic, the extended type (T) used is an interval. Using interval analysis and automatic differentiation as the arithmetic types, lower bounding information is used for optimization in a branch and bound framework.

Figure 1 illustrates a generic module allowing multiple types. This module must allow the transformations that need to be applied to some underlying type, T, to obtain the output. Each unit has input and output streams which are the feeds for the successive units. Input parameters determine how the unit operates, for example the split fraction in a splitter, and a cost can be evaluated based on calculated outputs. Residuals represent constraint violations.

Such modules can be developed to describe the operations in a process plant and then the appropriate operations are applied to data of type T. For example, a module which adds its inputs should use the interval arithmetic operations if the underlying type is an interval (T = interval). This formulation also includes models using traditional arithmetic. We have used interval operations within Matlab to test our method but some Sun compilers support interval operations.

Flowsheets can be built up from generic units and then evaluated using the operations for an extended type, T, which provides the information necessary for global optimization using interval methods where the variable type <T> used is interval. Each unit calculates the output streams (a vector of physical variables such as flowrates and temperature) and the cost associated with the unit as intervals. The summation of the costs provides the objective function. Design constraints are added outside the module to the optimization algorithm. A simple example is shown in the Appendix.

Interval gradient types have been used, and the global optimisation algorithm uses the NE (natural extension) and MV (mean value) underestimation schemes [6] to construct a linear relaxation of the objective function and constraints in terms of the optimisation variables. The solution of this linear relaxation provides the necessary bounds on the optimisation problem. The variable types <T> used were intervals with interval arithemetic and interval gradients using rules of automatic differentiation.

3 Interval global optimisation methods

The key to finding the global solution is the identification of the upper and lower bound to the problem. The algorithm used by Byrne and Bogle [6] uses bound constrained linear programs which can be created from a modularly constructed flowsheet. The relaxed problem formulation allows convex constraints formed from non convex problems (NCP) to be included in the lower bounding procedure.

3.1 The linear lower bound approach

The linear lower bound approach [5] generates linear lower bounds for a function, f. This is conducted by decomposing f into simpler functions, such as monotonic convex functions. Once linear lower bounds for each of the simpler functions are found, these bounds are recombined to obtain a linear lower bound for the original function. This approach assumes that f can be decomposed into simpler functions.

When relaxing constraints to find a lower bound it is important not to create too many additional variables. Thus, in place of introducing additional variables for nonlinear intermediate expressions, it is possible to relax the original constraints directly. Constraints are formulated as a difference of convex functions. Forward propagation is obtained in an automatic differentiation like manner, described in Kolev and Nenov [24]. This type of method is described in Byrne and Bogle [6].

3.2 Interval linear underestimators

Linear models have been proposed in Byrne and Bogle [6]. They are the MV and NE underestimators. The NE underestimators are linear under/over relaxations which can be used to obtain a linear relaxation of any variable of the problem. Because they are linear they can easily be characterised by the coefficients of the different components and a Linear Programming problem can be constructed. The MV relaxation also provides linear underestimators. The underestimators are constructed from gradient information in a similar manner to linearisation. This method has a particular advantage that one set of gradient bounds obtained by automatic differentiation can be used to construct as many underestimators as necessary.

Reformulation of the lower bounding problem is used to increase the number of constraints and terms in the objective function which can be retained in the lower bounding problem. This reformulation opens the possibility of adding more complementary relaxations (MV or NE) to the lower bounding problem. The MV relaxation is more general and more widely applicable, although less tight, than the $\alpha-$ BB relaxation [1] providing a linear relaxation of any once differentiable term in the problem by using interval analysis to bound the gradient. Though it is possible to construct convex relaxations using these techniques we have chosen to use only linear relaxations of NCP. The decision to use linear relaxations is based on the need to solve relaxed problems efficiently and reliably in preparation for the modular systems. This work can be found in Byrne and Bogle [6].

3.3 The interval optimization algorithm

The algorithm used here in optimising modular flowsheets can be found below. The algorithm relies on bounding the objective function over subsets, χ_k, of the feasible region, and maintaining an upper bound (indicated by a superscript u) on the value of the global optimum (y^u). If the lower bound (indicated by a superscript l) for χ_k is greater than y^u then χ_k cannot contain a global minimiser. As the partitions are refined the lower bounds become tighter and form a closer approximation of the objective function. When the difference between the lower and upper bound is within a certain tolerance then the global minimiser is located in the current partition. The algorithm used is as follows.

Algorithm: [6]

```
1. Initialise
    (a) a counter, $k = 0$
    (b) an initial region, $X_o \supseteq A$ where A is the full range of all variables.
    (c) store a triple $\chi = \left\{X = X_o,\ y_o = f(X)^l,\ x_o\right\}$ where $x_o$ is not set.
    (d) an upper bound on the global minimum $y^u = \text{infinity}$
2. Select and remove a stored triple, $\chi_{k,}$ with the lowest stored value of $y_k$.
3. If $x_k$ is not set then set $x_k$ equal to the local minimiser or a feasible point of $A \cap X_k$ if
   one can be determined by the local minimization phase.
4. If $|f(x_k) - y_k| \le \varepsilon$ then
    (a) $x_k$ is a global minimiser
    (b) Terminate
5. Set $y^u = \min\{f(x_k), y^u\}$
6. Partition $\chi_k$ giving $\chi_L$ and $\chi_R$.
7. Update $y_L^l$ and $y_R^l$. Store $\chi_L$ and $\chi_R$.
8. Remove any stored triples, $\chi_j$, j= 0 . . . k for which:
    (a) $X_j$ is completely infeasible
    (b) $y^l > y^u$
9. Increment k and return to step 2.
```

The algorithm can be modified to find all the global optima [2].

Byrne and Bogle [7] found modular optimisation computationally demanding. A variety of approaches to interval contraction have been applied and tested on flowsheeting problems here to explore ways of improving computational performance.

4 Interval contraction methods

The potential exists to improve interval global optimisation techniques within flowsheeting programs. Jaulin et al. [23] reviewed many interval contraction methods which tighten interval bounding. Our goal here is to investigate the broad range and variety of contraction methods available in the context of modular flowsheeting. In this work contractors based on the following operations have been used: Consistency techniques, Constraint propagation, Linear Programming contractors, Interval Gaussian elimination, and the Interval Newton contractor. The Contractors were inserted into the global optimisation algorithm above before step 6.

4.1 Consistency techniques

To determine the precise hull which encloses the solutions of a set of interval equations is NP-hard. However, contractors can be determined which enclose the region. Take an equation $f(x,y) = 0$ and assume that x and y are in intervals X and Y, respectively. The values of $x \in X$ and $y \in Y$ such that $f(x,y) = 0$ and for any $y \in Y$ there exist $x \in X$ such that $f(x,y) = 0$. If a scenario arises where $x \in X$ and there is no $y \in Y$ such that $f(x,y) = 0$, then these values of x can be excluded from consideration when seeking solutions for $f(x,y) = 0$. This concept of consistency can be applied to more than one variable which allows the elimination of sub-boxes.

There are two main consistency procedures known as Box and Hull consistency [26]. These methods are diverse and have been applied to other types of engineering problems [36]. The version and implementation we have used is the 'box-consistency' method taken from Hansen and Walster [18]. Hull consistency methods have been proposed but they are NP-hard and have been excluded.

Assume the solution of a given problem must satisfy the nonlinear constraint

$$f(x_1, \ldots, x_n) = 0.$$

Suppose that we seek a solution in a box X. We can use box consistency to eliminate sub-boxes of X that cannot contain a point satisfying the above equation.

If we replace all the variables except the i-th by their interval bounds (i.e. components of X), we obtain the following equation

$$q(x_i) = f\left(X_1, \ldots, X_{i-1}, x_i, X_{i+1}, \ldots, X_n\right) = 0.$$

If zero is not within $q(x_i)$ in some subinterval X_i' of X_i, then we do not have consistency for $x_i \in X_i'$ and the sub-box $\left(X_1, X_{i-1}, x_i, X_{i+1}, \ldots, X_n\right)$ of X can be deleted.

4.2 Constraint propagation

There are various types of constraint propagation discussed in Jaulin et al. [23]. The type of propagation we have used is the forward-backward contractor because it is more adaptable to modular flowsheets. This method takes one constraint in isolation, say $\left\{q_i(x_i, \ldots, x_n) = 0\right\}$ then deduces better bounds for a particular variable using other variable bounds by rearranging the constraint. The q_k chosen is most often a linear or quadratic function of a single variable only.

This technique may save a number of branching steps and thus speed up the interval algorithms. Special care should be taken in presenting (or transforming) the problem in a form which has as much separability as possible.

4.3 Linear Programming contractor

A linear Programming (LP) contractor bounds the solution set of the linear interval equation system. Given an upper bound, y^u on the solution of the non convex problem (NCP) and an enclosing set of lower bounding constraints this forms a LP with the reformulated non linear variables incorporated into the vector x of independent variables. This model is created by MV underestimators and overestimators of the constraints and objective function.

The contraction step solves a set of LPs to tighten the bounds on each of the variables from the NCP. For each x_i that appears non linearly in NCP it is necessary to solve

$$\begin{aligned}&\min x_i\\ &\text{s.t. } c^T x \le y^u,\ Ax \le b\end{aligned}$$

to tighten the lower bound on x_i and an equivalent problem for the upper bound. If either of these problems is infeasible then the box, X^k, can be deleted.

This method can be particularly useful when the initial interval boxes are much larger than the feasible region.

4.4 Interval Gaussian elimination methods

An interval version of Gaussian elimination is obtained from real arithmetic by simply replacing each real arithmetic step by the corresponding interval arithmetic step. This produces for a linear system, in this case the linearised constraints, an enclosure of the bounds on the solution. This is discussed in Hansen and Walster [18]. The linear system $Ax = b$ becomes an interval equation.

Using interval arithmetic we replace the real rational function by an inclusion isotonic interval extension of the rational function. An interval function F is said to be inclusion isotonic if $X_i \subset Y_i$ $(i = 1,\ldots, n)$ implies $F(X_1, \ldots, X_n) \subset F(Y_1, \ldots, Y_n)$. Each such component of the real solution constitutes a point, s, and the interval solution must contain s.

However, simply replacing real Gaussian Elimination by an interval version is generally inefficient. Bounds of intermediate quantities tend to grow very quickly because of accumulated rounding errors and especially because of dependence among the generated intervals. Preconditioning can help with this problem.

4.5 Interval Newton contractor

The Interval Newton method is concerned with finding and bounding all the solution vectors of $f(x) = 0$ in a given box X^0. The method was derived by Moore [27].

Hansen and Walster [18] applied the Interval Newton method using the Fritz–John condition to contract intervals with problems involving equality constraints. They show that this requires a substantial amount of computing due to extra interval variables of the Lagrange multipliers which are needed to solve the Fritz-John condition. This type of information will not be readily available in modules of modular process flowsheeting systems and thus a decision was made to avoid using the Fritz–John conditions.

An alternative technique is available for the Interval Newton method. If we fix a subset of the independent variables to obtain a square system as discussed in Hansen and Walster [18], both the Interval Gaussian Elimination and Interval Newton methods can be applied to this square system. The Interval Newton and Interval Gaussian methods are classified as fixed point contractors.

The Interval Newton procedure has some unique properties: for instance if there exists a solution x^* of f in X^n before contraction then x^* is also in the contracted interval. Also the Interval Newton algorithm always converges if there is a solution [16]. Other properties include that every discrete zero of f in X^0 is isolated and bounded to arbitrary accuracy and that if the intersection of the contracted interval and X^n is empty there is no zero of f in X^n. Other properties can be found in Hansen [17].

4.6 The implementation of fixed point contractors

Interval Newton and Interval Gaussian methods are easily applied to a problem containing an equal number of variables as constraints. These fixed point contractors can be applied to a non-square system once some variables are fixed to make the system square. The aim now for these fixed point contractors is to obtain a feasible point in the box. This fixed point is regarded as the upper bound. If the lower bound of a box is greater than a feasible point (upper bound) then the box can be deleted and as a result regions can be removed. This is still a form of contraction, but regions are removed as opposed to reducing the interval range. Hansen and Walster [18] describe this method in more detail.

In an optimization problem the number r of equality constraints is less than the number n of variables. Otherwise, it is generally possible for the constraints themselves to determine the solution point(s) with no degrees of freedom remaining to minimise f. Suppose we want to determine whether there is a feasible point in a given box X. Let x denote a variable value in X, and let c be the centre of X. We fix $n - r$ of the variable components of x by setting $x_i = c_i$ for $i = r + 1, \ldots, n$.

To choose which variables to fix we use the method discussed in Hansen and Walster [18]. q constraints are linearised about the centre x of a box X by using the Taylor series expansion. The following steps are taken to decide which variables to fix.

1. Compute a real matrix J^c, which is the approximate centre of the r by n matrix J(X).
2. Carry out Gaussian elimination on J^c using both row and column pivoting to manipulate J^c into a form in which elements in position (i, j) are zero for $1 \leq i \leq r$ and $1 \leq j \leq r$ except for $i = j$. In the process, the final column pivoting should ensure that the final element in position (r, r) is largest in magnitude among the elements in position (r, j) for $j = r, \ldots, n$.
3. Select the variables corresponding to those now in the final columns $r + 1, \ldots, n$ to be those to be replaced by their fixed points.

Once the variables have been chosen to apply a step of the fixed point contractors let Z' denote the solution box for Z, the remaining interval members of X. If $Z' \subset Z$, then there exists a solution of the constraint set in Z'. This implies that there exists a feasible point in the n-dimensional box

$$X' = \left(Z_1, \ldots, Z'_r;\ c_{r+1}, \ldots, c_n\right)^T$$

which is a partially degenerate sub-box of X, because of the fixed points.

If $Z' \subset Z$, then any of the fixed point contractors can be changed to reduce the size of the box bounding the feasible point. The smaller the box, generally the smaller the upper bound on f over the box.

If we have proved that a feasible point exists in a box, X, then the contracted intervals and fixed points are substituted into the objective function to give an upper bound. We evaluate f(X) producing upper and lower bounds. $f(x_k)^u$ is an upper bound for the global minimum. The global optimisation algorithm is updated every time an upper bound is calculated.

This will be inserted into the global optimisation algorithm to find a feasible point, and will be used as an upper bound to the problem. As the algorithm is being processed the interval boxes are being reduced in size and the Interval Newton method will provide better feasible points updating the upper bound.

The Interval Gaussian method is highly dependent that a zero does not occur in the denominator at any stage of the Gaussian Elimination method. If a zero does occur in the denominator then the method will not be used in the iteration. Methods are available to tackle this problem, but the complexity of the Interval Gaussian algorithm will increase and will be impractical when applied to flowsheeting problems. For this the reason both Interval Newton and Interval Gaussian Elimination methods have been investigated.

5 Numerical results

5.1 Mathematical problem set

The contractors were inserted in the global optimisation algorithm and were first applied to a set of mathematical problems. Each problem has a variation in the degree of complexity and reflect the difficulty found in flowsheeting problems. The problems contain equality constraints because they occur frequently in modules of flowsheeting problems. From the results obtained we should be able to identify the influence of interval contraction techniques on global optimization. The tolerance of all the problems tested is 0.001, unless specified. The computer used was a Pentium IV 1.5 GHz and the programme was Matlab 6.5 and the interval arithmetic package for Matlab (Acsysteme) by Houizot [22] was used which is based on the algorithms in Jaulin et al. The maximum designated time the simulation is allowed to run for was 12 h.

We have taken three problems from the set published by Hock and Schittkowski [20]: problems 39, 31 and 79 (all polynomial functions up to order 4 with 4, 3 and 5 variables, respectively). Table 1 demonstrates the influence of each contractor. Any combination of contractors can be used simultaneously. Rather than report the performance of each contractor individually we have presented the results with no contractors, with all contractors, and with each contractor removed individually. The number of ‘iterations’ is presented where this corresponds to box divisions along with the computational time taken to find one global minimum. A—indicates that no solution was found in the maximum time allowed.

From the results it can be seen that interval contractor techniques are definitely improving global optimisation. The most effective contractor is the Linear Programming contractor as considerable decreases were seen in numbers of iterations and global optimisation times using this contractor. However, the Linear Programming Contraction is NP-hard and may not be as effective for larger problems [6]. If the initial bounds are unnecessarily bigger than the feasible region then the Linear Programming contractor has the ability to eliminate these unnecessary regions. For the third problem the Consistency Method provided some

Table 1 Results of global optimisation algorithm using different contraction techniques

	HS Problem 39		HS Problem 31		HS Problem 79	
	Iterations	Time	Iterations	Time	Iterations	Time
No contractors	168	177.75	70	65.98	–	–
w/o consistency method	5	5.12	13	14.02	249	531.43
w/o constraint propagation	5	5.23	13	12.92	252	523.33
w/o Interval Gaussian	5	5.06	13	12.45	324	596.0
w/o Interval Newton	5	5.51	13	13.23	254	479.0
w/o Linear Programming	123	206.7	70	78.0	455	1490
All contractors	5	5.96	13	14.42	249	532.33

contraction in the global optimisation algorithms. Constraint Propagation provided some interval contraction. On the third problem (Hock and Schittkowski problem 79) the Interval Gaussian contraction performs better than the Interval Newton method. There is no conclusive best method here particularly since the problems are all small. The other contraction methods (apart from the LP contractor) may make a more significant contribution on larger problems and in flowsheet design problems.

5.2 Flowsheet design problems

5.2.1 Haverly pooling problem

The Haverly Pooling problem (Haverly [19], with details of the specific problem in Floudas and Pardalos [11]) has been used as a test case for global optimisation by many authors. This problem is to blend four feeds into two products to minimize the total cost (Fig. 2). Each feed has a composition and cost and each product has a cost, required composition, and flowrate. The feeds are mixed to produce products that satisfy the quality requirements using mixer and splitter units to represent the blending tanks. Further details of the problem can be found in Quesada and Grossmann [30]. The problem is a small scale blending problem and is non-convex due to bilinear terms.

The procedure presented here can be used to obtain the global optimum of large scale non-convex blending problems by reusing a very small number of generic units (mixers, splitters, feed and product units). This pooling/blending problem, and any other pooling or blending problem, can thus be formulated as a modular problem. It is natural to view the problem in terms of interconnected units with each unit performing some transformation of the input stream to provide the output stream. The flowrate is determined by a single input parameter and the cost by multiplying the unit cost, by the flowrate. The quality constraints placed on the two product streams become residuals in the product modules which have

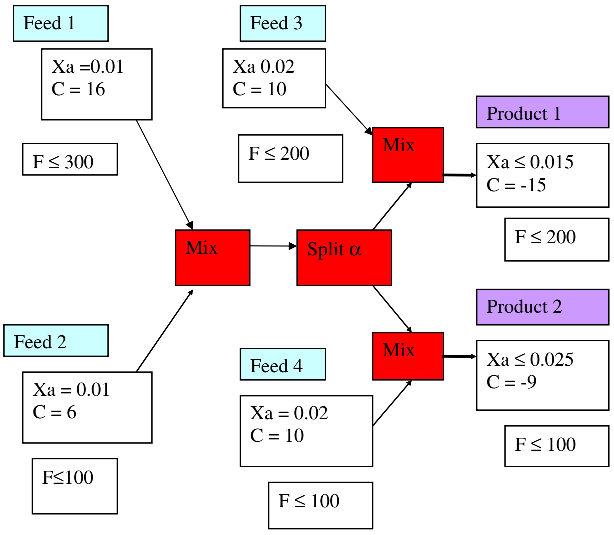

Fig. 2 The Haverly pooling problem (F—flowrates, Xa—concentration of component A, C—costs)

Table 2 Results for the Haverly pooling problem

	One solution		All solutions	
	Its	s	Its	s
No contractors	225	32	591	718
w/o consistency method	60	93	140	193
w/o constraint propagation	60	92	140	193
w/o Interval Gaussian	60	89	140	191
w/o Interval Newton	60	90	140	192
w/o Linear Programming	225	318	591	734
All contractors	60	93	140	193

one input stream and no outputs. In this formulation there are four independent variables partitioned and bounded by the interval optimization algorithm.

The modular global optimization algorithm is used together with a local minimization procedure to find a local upper bound. As the intervals are passed through the flow-sheet, if the intervals do not satisfy the product constraints then the interval box can be deleted (the consistency method). Constraint propagation uses the equations which produce the product residuals. The Interval Newton and Interval Gaussian procedure require specific information from certain parts of the modular flowsheet. As constraints occur only in the product specifications, the differential information and residual information will only be required at this point. The Linear Programming contractor requires cost information from the flowsheet and derivative information for both cost and variables. Finally the lower bound of the cost will be calculated by forming a linear model with underestimators.

The results for this problem can be found in Table 2. This time we have also included the results for when the algorithm is permitted to find all global optima. As with the mathematical problem set it is the LP contractor which provides all the contraction with little computational overhead.

5.2.2 Recycle flowsheet problem

The recycle problem taken from Floudas [9] was cast in a modular fashion and globally optimised in Byrne and Bogle [7] and this formulation was used here.

The problem is a plant involving the production of monochlorobenzine. Each unit has a capital cost and an operating cost which is incorporated into the objective function through a pay back time of 2.5 years. The principal units are a continuous stirred tank reactor (CSTR) and two separation columns, and unreacted feed is recycled back to the reactor. The reactor models the reaction between chlorine and benzene to produce monochlorobenzene and dichlorobenzene at a constant temperature.

The reactor model has a single input stream. The three parameters associated with this model are two rates of reaction and the volume. The cost of the feeds and products are: the purchase price of benzene \$27.98/kmol and chlorine \$19.88/kmol, and the sale price of monochlorobenzene, \$92.67/kmol for which there is a demand of 50 kmol/h. Both distillation column models assume that the columns performs sharp splits. A splitter module is used to purge a fraction of the recycle stream. There are six independent variables in this problem which are partitioned and bounded by the interval optimization algorithm.

Table 3 Results for the recycle problem

	One solution		All solutions	
	Its	s	Its	s
No contractors	–	–	–	–
w/o consistency method	240	1,930	246	2,083
w/o constraint propagation	154	1,020	160	1,170
w/o Interval Gaussian	154	996	160	996
w/o Interval Newton	166	927	172	943
w/o Linear Programming	–	–	–	–
All contractors	154	1,060	160	1,210

Table 4 Results for the reactor network problem

	One solution		All solutions	
	Its	s	Its	s
No contractors	532	1,049	1,325	2,600
w/o consistency method	283	614	379	634
w/o constraint propagation	52	77	111	169
w/o Interval Gaussian	52	68.14	111	135
w/o Interval Newton	83	82	111	135
w/o Linear Programming	76	80	183	242
All contractors	52	79.62	111	170

Table 3 shows that again the LP contractor proves to be the best contractor since without it no solution is found within the time constraint. However, this time the other contractors do have an appreciable effect with the consistency method proving to give the most value. Constraint propagation and Interval Gaussian have no effect.

5.2.3 Reactor network problem

The last example, taken from Ryoo and Sahinidis [31], is a reactor network design problem. This problem was chosen because it involves two complex reactors in series. It is natural to view the problem in terms of interconnected units with the reactor units performing some transformation of the input stream to provide the output stream. Here there are three independent variables.

For this problem it is not the LP contractor but the consistency method which provides the largest impact. Again the constraint propagation and Interval Gaussian methods have no effect. The LP contractor is still making a significant difference (Table 4).

From the results obtained it is evident that interval contractor techniques are definitely improving the interval global optimisation algorithm. The most effective contractor is the Linear Programming contractor as seen also with the mathematical problems. The approach applying the interval Newton method has a positive impact also. Constraint propagation and the Interval Gaussian method seem to be ineffective for modular flowsheet problems while the consistency method in some cases gives value. We believe that this is due to exacerbation of the interval dependence problem. The computational overhead for these contractors is in many cases not significant so it is useful to include any that do contribute to the contraction. In the case of these small problems the price to pay for finding all solutions is not large. However, this may be considerably more significant for larger and more complex problems.

6 Conclusions

The main conclusions of the paper are:

- An interval global optimisation algorithm has been applied to solving modular flowsheets incorporating the interval contraction techniques which help to improve computational performance
- Constraint propagation and the Interval Gaussian elimination procedure proved to be ineffective in solving these flowsheeting problems.
- Linear Programming techniques have proved to be the best contractor for solving these flowsheeting problems. However, other contractors do make a contribution and the combinatorial problems that the LP method has for large problems make it important to retain a combination of contractors. Also, this relies on derivatives which if determined numerically can be noisy which may make the LP contractor unuseable.

We have demonstrated that interval global optimization methods can be used with modular systems to determine the globally optimal solution(s) to design and operational problems, and that contraction methods can improve the computational performance. Existing systems do not currently support interval techniques so to use these ideas commercially would require modification of the modules and data structures within the simulation system. This would require converting data types to be of interval type within modules and the integrating simulation program, to provide derivatives (but see the final bullet point below), and to use a compiler that supported interval operations.

A number of major challenges remain before this capability can be incorporated in commercial systems.

- Any implementation would require the support of interval variable types.
- Many modules have internal iterations which would need to be solved perhaps by the interval Newton algorithm.
- Physical property calculation modules would have to be formulated to handle intervals. In principle these are just complex general procedures of the type outlined above.
- Handling discontinuities formally within flowsheet modules in a generic way is required.
- It is unrealistic to expect all modules to provide partial derivatives (although many currently do) so a method for determining derivatives of procedures automatically is required. Griewank et al. [14] have developed methods for automatic differentiation of procedures so there are ways forward to be explored.

Acknowledgements The authors acknowledge financial support for this project from the Engineering and Physical Sciences Research Council of the UK and Aspentech (and Hyprotech in the early stages of the project).

Appendix: Interval propagation for component 1 for the Haverly Pooling problem

For a modular system intervals for the independent variables are passed through the modules of the flowsheet. In the case of the Haverly Pooling problem there are just two types of module, the mixer and the splitter. In this case each module only consists of one equation but often they are much more complex with many equations, branches, and iterations. This is the simplest problem of its type but is still a non-convex optimization problem and represents a class of problems which can be of any size. Many pooling problems have many mixers and splitters and more complex units.

The vectors of interval stream variables (S_i), consisting of intervals of flowrates for each of the two component, are fed to the modules. For the first component these intervals are: S1 = [250 300], S2 = [150 300], S3 = [0 200], S4 = [0 100], and α = [0.5 1].

All Mixer modules implement the interval equation

$$S_{j,out} = \sum F_{ij} \quad \text{for each component, j, all i input streams}$$

which since all mixer cases here have only two inputs reduces, for each component, to

$$S_5 = S_1 + S_2.$$

Substituting the interval vectors above stream 5 becomes S_5 = [400 600]. S_5 passes to the splitter module which implements the equations

$$S_{out1} = \alpha\, S_{in}$$
$$S_{out2} = (1 - \alpha)\ S_{in}$$

or in this case

$$S_6 = \alpha\, S_5$$
$$S_7 = (1 - \alpha)\ S_5$$

and for the first component S_6 becomes [200 600].

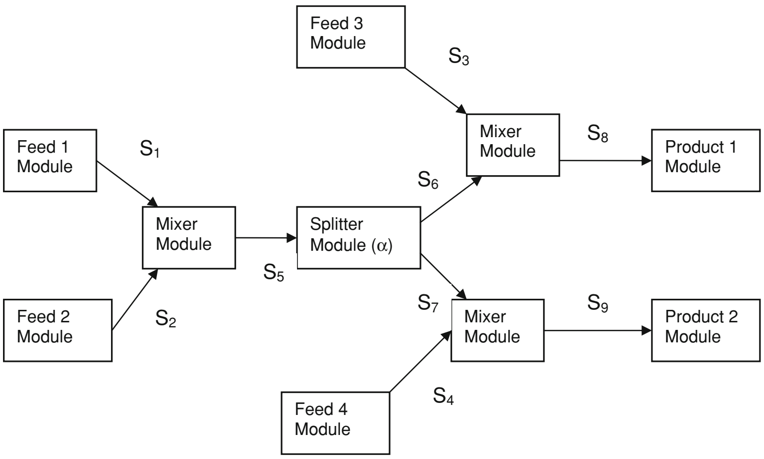

Fig. A1 Modular flowsheet for Haverly pooling problem showing stream variables

Mixer 2 uses the generic mixer equation above by mixing with S_3 [0 200]. The initial interval bound vector produced for stream S_8 is therefore [200 800]. However, the product constraint for this product is that the product concentration should be less that 200. No part of the interval satisfies the product constraint so this interval can be deleted from the candidate set of solutions. If the interval did completely satisfy the constraint it would be accepted by the algorithm, the objective function evaluated and the interval bisected or contracted by one of the contraction methods.

This can be repeated for component 2. If there were many components, as would normally be the case for industrial pooling problems with many feeds and products, the process would be identical. Although a very simple example this demonstrates how the propagation would work through modules.

References

1. Adjiman, C.S., Androulakis, I.P., Maranas, C.D., Floudas, C.A.: A global optimisation method, αBB, for process design. Comput. Chem. Eng. **20**, S419 (1996)
2. Balendra, S.: Global optimisation for modular process design. PhD Thesis University of London (2007)
3. Biegler, L.T.: Simultaneous modular simulation and optimization. In Proceedings of the Second International Conference on Foundations of Computer-Aided Process Design, CACHE (1983)
4. Biegler, L.T., Grossmann, I.E., Westerberg, A.W.: Systematic Methods of Chemical Process Design. Prentice Hall, New Jersey (1997)
5. Bromberg, M., Chang, T.C.: Linear lower bound approach. Recent Adv. Glob. Optim. **50**, 200–220 (1992)
6. Byrne, R.P., Bogle, I.D.L.: Global optimisation of constrained nonconvex programs using reformulation and interval analysis. Comput. Chem. Eng. **23**(9), 1341–1350 (1999)
7. Byrne, R.P., Bogle, I.D.L.: Global optimisation of modular flowsheets. Ind. Eng. Chem. Res. **39**(11), 4296–4301 (2000)
8. Edgar, T.F., Himmelblau, D.M., Lasdon, L.S.: Optimization of chemical processes, 2nd edn. McGraw Hill 2001
9. Floudas, C.A.: Nonlinear and Mixed-Integer Optimization. Fundamentals and Applications. Oxford University Press, New York (1995)
10. Floudas, C.A.: Recent advances in global optimization for process synthesis, design and control: enclosure of all solutions. Comput. Chem. Eng. **23S**, S963–S973 (1999)
11. Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained Global Optimization Algorithms Lecture Notes in Computer Science. Springer, Berlin (1990)
12. Floudas, C.A., Visweswaran, V.: A global optimization algorithm (gop) for certain classes of nonconvex NLPs-I. Theory Comput. Chem. Eng **13**(10), 1117–1132 (1990)
13. Garrard, A., Fraga, E.S.: Mass exchange network synthesis using genetic algorithms. Comput. Chem. Eng **22**(12), 1837–1850 (1998)
14. Griewank, A., Juedes, D., Utke, J.: Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. ACM TOMS **22**(2), 131 (1996)
15. Grossmann, I.E.: Global Optimisation for Engineering Design. Kluwer Academic, Dordrecht (1996)
16. Hansen, E.: A globally convergent interval method for computing and bounding real roots. BIT Numer. Math. **18**, 415–424 (1978). doi:10.1007/BF01932020
17. Hansen, E.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (1992)
18. Hansen, E., Walster, G.W.: Global Optimization Using Interval Analysis, 2nd edn. Marcel Dekker, New York (2003)
19. Haverly, C.A.: Studies of the behavior of recursion for the pooling problem. SIGMAP Bull. **25**(Dec), 19–28 (1978). doi:10.1145/1111237.1111238
20. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lecture notes on Economics and Mathematical Systems Springer-Verlag, New York (1981)
21. Horst, R., Tuy, H.: Global Optimization. Deterministic Approaches. Springer-Verlag, Berlin (1992)
22. Houizot, P.: Acsysteme interval toolbox user guide. www.acsysteme.com. (2003)
23. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer-Verlag, London (2001)
24. Kolev, L.V., Nenov, I.P.: Cheap and tight bounds on solution set of perturbed systems of nonlinear equations. Reliable Comput. **7**, 399–408 (2001). doi:10.1023/A:1011475926711

25. Lin, Y., Stadtherr, M.A.: Advances in interval methods for deterministic global optimization in Chemical Engineering. J. Glob. Optim. **29**, 281–296 (2004). doi:10.1023/B:JOGO.0000044770.73245.14
26. McAllester, D., Hentenryck, P.V., Kapur, D.: Three cuts for accelerated interval propagation. MIT Institute for Artificial Intelligence Lab memo 1542 (1995)
27. Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs, NJ (1966)
28. Perkins, J.D.: Equation Oriented Flowsheeting. In Proceedings of the Second International Conference on Foundations of Computer-Aided Process Design; CACHE, 1983.
29. Quesada, I., Grossmann, I.E.: Global optimization algorithm for heat-exchanger networks. Ind. Eng. Chem. Res. **32**(3), 487–499 (1993). doi:10.1021/ie00015a012
30. Quesada, I., Grossmann, I.E.: Global optimization of bilinear process networks with multicomponent flows. Comput. Chem. Eng. **19**(12), 1219–1242 (1995)
31. Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput. Chem. Eng. **19**(5), 551 (1995)
32. Schmidt, C., Biegler, L.T.: A simultaneous approach for flowsheet optimization with modeling procedures. Trans. I.Chem. E. **72**, 382–388 (1994)
33. Schnepper, C.A., Stadtherr, M.A.: Robust process simulation using interval methods. Comput. Chem. Eng. **20**(2), 187–199 (1996)
34. Torn, A., Zilinskas, A.: Global Optimization Lecture Notes in Computer Science. Springer-Verlag, Berlin (1989)
35. Vaidyanathan, R., El-Halwagi, M.: Global optimisation of nonconvex programs via interval analysis. Comput. Chem. Eng. **18**(10), 889–897 (1994)
36. Yannou, B., Harmel, G.: Use of Constraint Programming for Design. Springer-Verlag, London (2006)