On the solution of mixed-integer nonlinear programming models for computer aided molecular design


Department of Chemical Engineering, University of Connecticut, Unit 3222, 191 Auditorium Road, Storrs, CT 06269, USA

Received 30 November 2001; accepted 14 May 2002

Abstract

This paper addresses the efficient solution of computer aided molecular design (CAMD) problems, which have been posed as mixed-integer nonlinear programming models. The models of interest are those in which the number of linear constraints far exceeds the number of nonlinear constraints, and with most variables participating in the nonlinear terms. As a result global optimization methods are needed. A branch-and-bound algorithm (BB) is proposed that is specifically tailored to solving such problems. In a conventional BB algorithm, branching is performed on all the search variables that appear in the nonlinear terms. This translates to a large number of node traversals. To overcome this problem, we have proposed a new strategy for branching on a set of linear branching functions, which depend linearly on the search variables. This leads to a significant reduction in the dimensionality of the search space. The construction of linear underestimators for a class of functions is also presented. The CAMD problem that is considered is the design of optimal solvents to be used as cleaning agents in lithographic printing.

© 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Global optimization; Linear underestimators; Computer aided molecular design; Branch-and-bound procedure; Mixed-integer nonlinear programming

1. Introduction

The synthesis of new chemical compounds by computer aided molecular design (CAMD) approaches has received considerable attention mainly due to the ever-increasing demand for application specific materials and the promise of reducing the time and effort required using empirical techniques. CAMD is a reverse engineering procedure that generates molecules, which exhibit certain physical property attributes. In this approach, the design of a compound is often expressed as a mathematical programming model. In this paper we present a branch-and-bound (BB) global optimization algorithm that is tailored to mixed-integer nonlinear programming problem (MINLP) formulations that arise in molecular design models such as ours.

1.1. Background of CAMD

In mathematical programming based CAMD models, a large number of binary (integer) and continuous variables define the search space for feasible molecules (Churi and Achenie, 1996; Joback and Stephanopoulos, 1989; Maranas, 1997; Odele and Machietto, 1993; Pistikopoulos and Stefanis, 1998). Employing structural constraints, which are expressed in terms of binary variables, ensures a valid molecular structure. Very often the expressions that model the molecular property requirements are nonlinear. The CAMD model is a MINLP with a relatively large number of linear constraints and a few nonlinear/nonconvex constraints.
Most of the search variables appear in the nonlinear terms.

A molecular design problem may be modeled as a single objective minimization/maximization subject to structural and performance constraints. Often the performance constraints posed are limits on the properties; the performance objective may be some undesirable property such as a compound’s ozone depletion potential. A typical CAMD problem for thermo-physical property matching may be written as

\[
\begin{align*}
\text{minimize/ maximize} \quad & p_{\text{obj}}(x, v, \theta) \\
\text{subject to:} \quad & p_k^l \leq p_k(x, v, \theta) \leq p_k^u \\
& \forall k = 1, 2, 3, \ldots, m_1 \\
& h_j(v, \theta) = 0 \quad \forall j = 1, 2, 3, \ldots, m_2
\end{align*}
\]

where \( v \) is a vector of binary variables that define the molecular structure, \( x \) is a vector of continuous variables such as process variables (pressure, temperature, etc.), \( \theta \) is a vector of group contribution parameters, \( p_{\text{obj}} \) is the objective function (for example some undesirable property to be minimized during design), \( p_k \) is the specified property requirements constrained between an upper (\( p_k^U \)) and a lower bound (\( p_k^L \)), and \( h_j \) is the set of equality constraints generally associated with structural feasibility requirements. The number of inequality constraints is \( 2m_1 \) and the number of equality constraints is \( m_2 \). Embedded in this formulation are several hundred candidate molecules. The solution of the above mathematical program gives the optimal molecular structure.

Most of the property constraints are of the form \( p_k = \sum n_j \theta_j^l \sum n_j \theta_j^u \) where \( n_j \) is the number of type \( j \) groups present in the molecule. In addition, ‘1’ and ‘2’ denote two different types of molecular property; \( x \) is an exponent (generally equal to 1). For example, molar density is the ratio of molar weight (‘1’) and molar volume (‘2’). Other examples of such models include glass transition temperature and specific heat capacity (Maranas, 1996). Transformation of such constraints into a linear form is straightforward. However, some property constraints are of the form \( p_k = f^1_{NL}(\sum n_j \theta_j^l)/f^2_{NL}(\sum n_j \theta_j^u) \) where \( f^1_{NL} \) and \( f^2_{NL} \) are nonlinear functions, and \( a \) and \( b \) are parameters. Property constraints of the given form include, group contribution models for solubility parameter (see Eq. (17)), and Reidel’s corresponding states method for prediction of the heat of vaporization and vapor pressure. It is not always possible to reformulate these constraints into linear or convex forms.

A more general molecular design problem than the one given earlier can be written as

\[
P_{\text{MD}} : \quad \text{minimize } f(x, v, \theta) \\
\text{Octet rule: } \quad g_0(x, v, \theta) = 0 \\
\text{Molecular structural constraints: } \quad g_1(x, v, \theta) \leq 0
\]

Physical property constraints:\n
\[
g_2(x, v, \theta) \leq 0 \quad (1)
\]

In our formulation of the CAMD problem, \( g_0 \) and \( g_1 \) are linear, while \( g_2 \) is mixed (i.e. some elements are linear, others are nonlinear). The nonlinear mathematical programming model for the CAMD problem (PMD) has the following features:

i) It is a nonconvex MINLP problem involving a large number of binary variables.

ii) The number of linear constraints is much greater than the number of nonlinear constraints.

iii) Most of the components of the optimization variables \([x, v]\) participate in the nonlinear terms.

We now present the development of a BB global optimization algorithm that exploits this problem structure. First, we outline the steps of a conventional BB algorithm, discuss its limitation for the CAMD problem, and then propose modifications.

In mathematical programming based CAMD models, a large number of binary (integer) and continuous variables define the search space for feasible molecules (Churi and Achenie, 1996; Joback and Stephanopoulos, 1989; Maranas, 1997; Odele and Machietto, 1993; Pistikopoulos and Stefanis, 1998). Employing structural constraints, which are expressed in terms of binary variables, ensures a valid molecular structure. Very often the expressions that model the molecular property requirements are nonlinear. The CAMD model then becomes a MINLP with a relatively large number of linear constraints and a few nonlinear/nonconvex constraints. Most of the search variables appear in the nonlinear terms. In this paper we present a BB global optimization algorithm that is tailored to MINLP problems that arise in molecular design models such as ours.

Local optimization techniques have been employed to solve molecular design problems. More recently, Friedler et al. (1998) used a BB strategy for solving a mixed integer formulation of the CAMD problem. Vaidyanathan et al. (1994) used GINO, which is a BB software package. Variants of the outer approximation method (such as the augmented penalty outer approximation algorithm by Viswanathan and Grossmann, 1990), which solve a sequence of NLP (nonlinear program) sub-problems and MILP (mixed integer linear program) master problems, have been used by researchers such as Odele and Machietto (1993), and Duvedi and Achenie (1996). The GAMS/DICOPT commercial software implementation of the augmented penalty outer approximation algorithm was employed by Raman and Maranas (1998) for solving the CAMD/MINLP formulation. All these approaches for solving the MINLP formulation are local, in the sense that globally optimal
solutions can be obtained only under certain convexity assumptions.

Some researchers have also applied global optimization techniques to solve molecular design problems. Vaidyanathan and El-Halwagi (1994) employed an interval analysis based global optimization approach for the CAMD/MINLP formulation. The algorithm is efficient for problems of small dimensionality. Venkatasubramaniam and Chan (1989) used a genetic algorithm for solving the CAMD problem. Since, the genetic algorithm is a soft computing approach that employs heuristics, it cannot guarantee global optimality. It can however, provide a good approximate solution in a reasonable amount of time.

In Section 2, a conventional BB algorithm is discussed. This is followed by a discussion of linear under- and overestimators for a class of functions in Section 3. In Section 4, we discuss issues involved in the choice of branching functions. Next, in Section 5, the molecular design problem is presented. Finally, in Sections 6 and 7 we present and discuss some computational results.

2. BB algorithm preliminaries

Let us pose an MINLP model in the general form

\[ P : \min_{x, v \in D} f(x, v) \]

Such that \( D = \{ x : x^L \leq x \leq x^U, \varphi_i(x, v) \leq 0, \quad i = 1, \ldots, m, \quad x \in X \subseteq \mathbb{R}^n, \quad v \in \{0, 1\}^q \} \), where \( x \) is a \( n \)-dimensional vector of continuous variables and \( v \) is a \( q \)-dimensional vector of binary variables. In the ensuing discussions, components of \( x \) and \( v \) are appropriately subscripted.

We propose to solve the problem using a BB approach (Horst and Tuy, 1990). The latter has been used for solving several problems in chemical engineering (see Ostrovsky et al., 1990; Friedler et al., 1998; Quesada and Grossmann, 1995; Maranas and Floudas, 1997; Adjiman et al., 1998). It should be noted that in a paper by Ryoo and Sahinidis (1996), an effective technique for reducing the domain of the search variables has been given.

A conventional BB approach consists of the following algorithms: (i) an algorithm for estimating a lower bound on the objective function, \( f \); (ii) an algorithm for estimating an upper bound on \( f \); and (iii) an algorithm for partitioning the feasible region \( D \). The generic BB method (Horst and Tuy, 1990) looks for a minimum of the objective function \( f(x, v) \) by partitioning the region \( D \) into subregions \( D_i \) with respect to the search variables. At each iteration, a subregion \( D_i \) is further partitioned into \( D_{ip} \) and \( D_{iq} \) (\( D_i = D_{ip} \cup D_{iq} \)) by introducing the constraints (based on a branching point \((x^*, v^*)\))

\[ x_i \leq x_i^* \text{ and } x_i \geq x_i^* \quad \text{or} \quad v_i \leq v_i^* \text{ and } v_i \geq v_i^* \tag{2} \]

The branching point \((x^*, v^*)\) is determined at each iteration at the solution of the lower bound problem discussed later. Thus in this case \( n+q \) branching variables are used. In a realistic molecular design problem, the number of branching variables can be several hundred. It is known that the number of branching nodes grows exponentially. To alleviate this problem, we avoid branching on the variables and instead use appropriate functions \( \psi_i(x, v) \) \((j = 1, \ldots, p)\) of the optimization variables for branching. Subsequently, a subregion \( D_i \) is determined by

\[ a_j^i \leq \psi_j(x, v) \leq b_j^i, \quad j = 1, \ldots, p \]

such that \( a_j^i \) and \( b_j^i \) result from the BB strategy. Thus \( D_i \) has the form

\[ D_i = \{ x, v : x, v \in D_i, \quad a_j^i \leq \psi_j(x, v) \leq b_j^i, \quad j = 1, \ldots, p, \quad v \in \{0, 1\}^q \} \]

Therefore, problem \( P \) is restricted on \( D_i \) as

\[ f_i = \min_{x, v \in D_i} f(x, v) \tag{3} \]

2.1. Lower bound algorithm

A lower bound \( f_{\inf}(\{i\})^{\{\text{all } L\}} \) for \( f_{\inf}(\{i\})^{\{\text{all } L\}} \) is obtained from the MILP

\[ \text{P}^L : \min_{x, v \in D_i} L_{\text{C}}[f(x, v)] \]

where

\[ D_i = \{ x, v : L_{\text{C}}[\varphi_k : D_i] \leq 0, \quad k = 1, \ldots, m, \quad L_{\text{C}}[\psi_j : D_i] \leq b_j^i, \quad L_{\text{C}}[-\psi_j : D_i] \leq -a_j^i, \quad v \in \{0, 1\}^q \} \]

is a convex underestimator for the generic function \( g(x, v) \) over \( D_i \). Then it is easy to verify that \( D_i \subset \hat{D}_i \). The branching point \( \psi^* = \psi(x^*, v^*) \) is determined at the solution \((x^*, v^*)\) of \( \text{P}^L \).

There are the following alternatives for estimating lower bounds (see Smith and Pantelides, 1997 for details).

1) The use of linear (or convex nonlinear) under-estimators.
2) The variables \( v \) are relaxed. That is, they are considered as continuous variables bounded by 0 and 1. In this case, the variables become integer only after the algorithm has converged.
3) Enforcing the integrality of all the binary variables \( v \) at each iteration.

We will construct linear underestimators and we will enforce integrality of all the variables \( v \) at each iteration. The resulting problem (i.e. \( \text{P}_L^{\text{box}} \)) is a MILP. If however, convex nonlinear underestimators are employed, then \( \text{P}_L^{\text{box}} \) is a convex MINLP, for which the outer approxima-
2.2. Algorithm for calculation of upper bound

We can estimate an upper bound \( f^U_i \) for \( f_i \) on \( D_i \), by setting \( f^U_i = f(\bar{x}, \bar{v}) \), where \( \bar{x} \) and \( \bar{v} \) is a feasible point for Eq. (3) and can be obtained through

\[
\bar{x} = \min_{x,v} \phi_j
\]

where

\[
\phi_j = \begin{cases}
\phi_j, & \text{if } j = 1, \ldots, m \\
\psi_j - m_{j-m}, & \text{if } j = (m + 1), \ldots, (m + p) \\
-p_j - (m + p), & \text{if } j = (m + 1 + p), \ldots, (m + 2p)
\end{cases}
\]

(4)

The nondifferentiable problem can be reformulated as

\[
\gamma^* = \min_{x,v} \gamma
\]

\[
\bar{\phi}_j(x,v) \leq \gamma, \quad j = 1, \ldots, (m + 2p)
\]

(5)

This is a nonconvex problem and therefore computationally intensive to solve at each iteration. To circumvent this problem, we obtain an upper estimate \( \tilde{\gamma} \) of \( \gamma^* \) by solving the following MILP

\[
P^U: \quad \tilde{\gamma} = \min_{x,v} \gamma
\]

\[
U[\tilde{\phi}_j(x,v); D_j] \leq \gamma, \quad j = 1, \ldots, (m + 2p)
\]

where \( U[\tilde{\phi}_j(x,v); D_j] \) is a linear overestimator of \( \tilde{\phi}_j(x,v) \) on \( D_j \), such that \( U[\tilde{\phi}_j(x,v); D_j] \geq \phi_j(x,v), \forall (x,v) \in D_j \).

Let \( \tilde{D}_j = \{ x,v : U[\tilde{\phi}_j(x,v); D_j] \leq 0 \} \), then \( \tilde{D}_j \subset D_j \) and \( \tilde{\gamma} \geq \gamma^* \). We can terminate the solution to \( P^U \) whenever the current value of \( \gamma \) \( \leq 0 \).

One of the following scenarios may occur at the \( k \)th step of the BB algorithm.

1) \( \tilde{D}_i \neq \emptyset \), \( \tilde{\gamma} < 0 \). Here we can calculate both the lower and upper bounds.

2) \( \tilde{D}_i \neq \emptyset \), \( \tilde{\gamma} > 0 \). In this case we can only calculate the lower bound since we cannot ensure that the point obtained by solving \( P^U \) will be feasible for Eq. (3) and consequently the upper bound cannot be calculated.

3) \( \tilde{D}_i = \emptyset \). In this case, \( \tilde{D}_i \) does not contain solution points and therefore it can be excluded from consideration.

Next, we discuss a method for construction of linear underestimators for a certain class of functions. This method leads to the calculation of a lower bound by solving a problem with dimensionality no greater than the dimensionality of problem \( P \). Moreover, there are instances where the number of branching variables actually decreases.

3. Construction of linear estimators

The main issue in the BB method is the construction of under- and overestimators. McCormick (1976, 1983) suggested the factorable programming technique for constructing convex underestimators for a function represented in factorable form. A more detailed treatment concentrating on implementation of these concepts is given in Ghildyal and Sahinidis (1997) and Smith and Pantelides (1997).

Quesada and Grossmann (1995) developed a technique for constructing convex underestimators for linear fractional and bilinear programming based on convex underestimators for the product of two univariate functions, \( f(x)g(y) \) (McCormick, 1976). Sherali and Alameddine (1992) suggested a general approach for constructing underestimators for arbitrary polynomial functions. The main drawback in this approach is that the dimensionality of the resulting linear program has a high order dependence on the dimensionality of \( P \).

A method for construction of underestimators for more general functions is proposed in the \( \alpha \)-BB global optimization method (Adjiman and Floudas, 1996). Maranas (1997), Maranas and Floudas (1997) suggested a method of constructing convex underestimators for multiplication of \( N \) univariate functions. The dimension of the lower bound problem, in all the above approaches, is larger than the dimension of the original problem (in some cases it can be significantly larger). Here we present an alternative approach in which the lower bound problem has a dimension no greater than the dimensionality of the original problem.

Let us consider a special class of functions \( f(x) \) (termed special tree functions, STFs) that can be represented by the following recurrent relations

\[
\phi_i^{(1)} = x_i, \quad i = 1, \ldots, n
\]

\[
\phi_i^{(N-k)} = \sum_{j \in Q_{i-k}} e_j^{(N-k)} f_j^{(N-k)} (\phi_j^{(N-k-1)}), \quad i = 1, \ldots, p_k, \quad k = 1, \ldots, (N-2)
\]

(6)

Here \( f_j^{(N-k)}(y) \) is a continuous univariate convex or concave function. Note that an STF is a multilevel representation of the function \( f(x) \), where the set \( \phi_i^{(1)} = x_i, \quad i = 1, \ldots, n \) corresponds to the first level and the set \( \phi_i^{(k)} = x_i, \quad i = 1, \ldots, p_k \) corresponds to the \( k \)th
level. Thus the STF is a superposition of univariate concave or convex functions connected by simple arithmetic operations, namely addition, subtraction, and multiplication by a scalar. It is clear that the STF is a special case of the factorable functions (a tree function) described in McCormick (1983) and McCormick (1976) such that the multiplication operation is missing. Thus when transforming a tree function into an STF we must remove all the multiplication operations. There are different ways to achieve this. For example, in order to remove the product term \( f(x)g(y) \) we will use the following identity (McCormick, 1983):

\[
f(x)g(y) = \frac{1}{4} (f(x) + g(y))^2 + \frac{1}{4} (f(x) - g(y))^2
\]

(7)

Using the transformation for each product term one can transform any factorable function to the STF. For a quadratic function

\[
f = x^T Ax + b^T x + c
\]

we will use a linear transformation \( x = Cz \) \((z = Bx, B = (C)^{-1})\) to yield the canonical form (Gantmaher, 1967)

\[
f = \sum_{i=1}^{p} d_i(b_{ii}x_i + \cdots + b_{im}x_m)^2
- \sum_{i=p+1}^{n} d_i(b_{ii}x_i + \cdots + b_{im}x_m)^2
\]

(9)

It is clear that the function is the STF.

3.1. Construction of linear underestimators for tree functions

We propose a strategy for constructing a linear underestimator for \( f(x) \) represented in the form (6). We will assume that all the coefficients \( c_i^{N-k} \) are positive without loss of generality. If a coefficient \( c_i^{N-k} \) is negative, one can employ the notation \( c_i^{N-k} = -c_i^{N-k} \) and \( f_i^{N-k} = -f_i^{N-k} \), and replace \( c_i^{N-k} \) by \( d_i^{N-k} f_i^{N-k} \)

Let \( \varphi_j^{N-k} \in S_j^{N-k} \) and

\[
S_j^{N-k} = \{ \varphi_j^{N-k} : \varphi_j^{N-k,L} \leq \varphi_j^{N-k} \leq \varphi_j^{N-k,U} \}
\]

(10)

where \( \varphi_j^{N-k,L} \) and \( \varphi_j^{N-k,U} \) are lower and upper bounds.

If we know the bounds for \( x_i \) at the first level, we can estimate bounds for \( \varphi_j^{N-k} \) at all levels through the use of interval arithmetic (Moore, 1966). A linear under-estimator of the function \( \varphi_j^{N-k} \) in the region \( S_j^{N-k} \) with respect to the functions \( \varphi_i^{N-k} \), \( j \in Q_j^{N-k} \) will be designated as \( L[\varphi_j^{N-k}; S_j^{N-k}] \). We obtain recurrent relations connecting linear underestimators of neighboring levels. Since \( c_i^{N-k} \geq 0 \), we have

\[
\varphi_j^{N-k} \equiv \sum_{j \in Q_j^{N-k}} c_i^{N-k} L[\varphi_i^{N-k}(y); S_i^{N-k}]
\]

\[
\leq \sum_{j \in Q_j^{N-k}} c_i^{N-k} f_i^{N-k}(\varphi_i^{N-k-1})
\]

(11)

where \( \varphi_j^{N-k} \) is an underestimator of \( \varphi_j^{N-k} \). Here we obtain an expression for \( \varphi_j^{N-k} \) using expressions for linear underestimators of univariate convex and concave functions \( f_i^{N-k}(\varphi_i^{N-k-1}) \) with respect to \( \varphi_i^{N-k-1} \). Let the latter satisfy Eq. (10). To simplify the notation for subsequent developments, let \( y = \varphi_i^{N-k-1} \) and consider \( f(y) \) in the region \( S_y = \{ y : \bar{y} \leq y \leq \bar{y} \} \). If \( f(y) \) is concave then in \( S_y \) the linear underestimator has the form (McCormick, 1976)

\[
L[f(y); S_y] = f(\bar{y}) + \frac{[f(\bar{y}) - f(\bar{y})]}{y - \bar{y}} (y - \bar{y})
\]

If instead \( f(y) \) is convex then as illustrated in Fig. 1, a linear underestimator is given by the tangent at the point \( C \) with abscissa \( y_m = 0.5(\bar{y} + \bar{y}) \). In this case the underestimator is given by

\[
L[f(y); S_y] = f'(y_m)(y - y_m) + f(y_m)
\]

where \( f'(y_m) \) is the derivative of the function \( f(y) \) at the point \( y_m \). Substituting the expressions for linear underestimators of the functions \( f_i^{N-k} \) in \( \varphi_i^{N-k} \) we obtain

\[
\varphi_j^{N-k} = \sum_{j \in Q_j^{N-k}} d_j \varphi_j^{N-k-1} + b_j^{N-k}
\]

Again we will assume that \( d_j > 0 \) without loss of generality; otherwise we can employ the transformation discussed earlier. Subsequently, we obtain the following expression for the linear underestimator of \( \varphi_j^{N-k} \) as

\[
L[\varphi_j^{N-k}; S_j^{N-k}] = L[\varphi_i^{N-k}]
\]
At the \((N-k-1)\)th level, we need to know the sign of \(d_j\) (determined at the adjacent upper level, \(N-k\)). Therefore, we need to work from the \(N\)th level to the 2nd level to obtain all relations in Eq. (12) for \(k = 0, 1, \ldots, N-1\).

The linear relations in Eq. (12) imply that a linear underestimator for the function \(F_1^{(N)}\) is a linear function of \(x_i\). From the above consideration, we propose the following algorithm for construction of a linear underestimator for a special tree function.

**Algorithm 1 (Construction of linear underestimator).**

1) Transform the objective function and constraints to the STF form.
2) Perform a bottom to top sweep to obtain all bounds 
\[
[\varphi_1^{(N)}, \pi_1^{(N)}], \quad \forall k = 1, \ldots, N \text{ and } i = 1, \ldots, p_k
\]
3) Perform a top to bottom sweeps to obtain the relations (12) for all levels.
4) Perform a bottom to top sweep to obtain 
\[
L[\varphi_1^{(N-k)}, \pi_1^{(N-k)}] \text{ as linear functions of } x.
\]

It is important that the underestimator is a linear function of the search variables \(x\). We note that the dimension of the lower bound problem \(P^L\) is the same as dimension of the original problem \(P\). We also note that in general one can obtain a nonlinear convex underestimator for the function \(F_0^{(N)}\) if McCormick’s technique (McCormick, 1976) is employed for obtaining convex underestimators for \(f_{\mu}^{(N-k)}(\varphi_1^{(N-k)})\).

4. Choosing branching expressions for BB

In a conventional BB method, the branching variables are the search variables \(x_i\). However, the large dimensionality of \(x\) can result in a rapid growth in the number of branches in the BB tree.

Here we propose another approach for selection of branching expressions. In some cases it allows us to exploit the problem structure and this leads to a decrease in the number of branching expressions. We will use as branching expressions *functions of the variables* instead of *the search variables* themselves. Let us consider our approach in detail. Denote the change in the function \(F_1^{(k)}\) by \(d_1^{(k)}\) such that
\[
d_1^{(k)} = F_1^{(k)} - \varphi_1^{(k)}
\]

Introduce the set \(S_1^{(l)}\) of subscripts \(j\) of the functions \(F_1^{(l)}\) at the \(l\)th level such that \(F_1^{(l)}\) is an argument of at least one of the functions \(f_{\mu}^{(l+1)}\) from the \((l+1)\)th level. In addition, introduce the set \(F_1^{(k)}\) of all \(F_1^{(l)}(l = 1, \ldots, k; j \in S_1^{(l)})\) belonging to the levels 1, \(\ldots, k\). For construction of an underestimator for the function \(F_1^{(N)}\) we must construct underestimators for all \(F_1^{(l)}(l = 1, \ldots, N; i = 1, \ldots, p_l, j \in Q_1^{(l)})\). This means that in order to obtain a tight underestimator for \(F_1^{(N)}\), all \(d_1^{(l)}(l = 1, \ldots, N; j \in S_1^{(l)})\) must tend to zero as the BB proceeds. Therefore, in general each function \(F_1^{(l)}\) in \(F_1^{(N)}\) must be a branching one. However, the number of branching variables can be decreased.

In fact, it is seen from Eq. (6) that at the \(k\)th level, each \(F_1^{(k)}\) is a complicated function of the functions \(F_1^{(l)}\) at the \(l\)th level if \(l < k\). Therefore, if \(d_1^{(l)} \rightarrow 0 (j = 1, \ldots, p_l)\) then \(d_1^{(k)} \rightarrow 0 (j = 1, \ldots, p_k)\) for \(k > l\). It follows from this that for any \(l (l = 1, \ldots, N-1)\) the set \(F_1^{(l)}\) can be used as a set of branching variables if branching with respect to all functions from \(F_1^{(l)}\) guarantees convergence to zero intervals of change of all arguments of \(f_{\mu}^{(k)}(k = l+1, \ldots, N; i = 1, \ldots, p_k, j \in Q_1^{(k)})\). In a nutshell the following condition must be met: if \(d_1^{(l)} \rightarrow 0 (k = 1, \ldots, l; j \in S_1^{(k)})\) then \(d_1^{(k)} \rightarrow 0 (q > l; j \in S_1^{(q)})\). We refer to this set as a branching set. We need to select \(l\) such that \(F_1^{(l)}\) contains the least number of \(F_1^{(l)}\). Often the set \(F_1^{(l)}\) can be divided into two subsets \(F_1^{(l)}\) and \(F_1^{(l)}\) such that any function from \(F_1^{(l)}\) can be represented as a function of the elements from \(F_1^{(l)}\). In this case one can use only the set \(F_1^{(l)}\) as a branching set.

The suggested approach to selection of branching expressions will be advantageous if the number of functions from the subset \(F_1^{(l)}\) is less than the number of variables \(x_i\) \((i = 1, \ldots, n)\). As an example, consider the determination of the set of branching functions for the quadratic function in Eq. (8), which is equivalent to the function in Eq. (9). The STF of the function has three levels. The first level corresponds to the search variables \(x_i\) \((i = 1, \ldots, n)\). The second level corresponds to the linear functions
\[
F_1^{(2)} = (b_{i_1}x_1 + \cdots + b_{i_n}x_n)
\]
where \(i_1 = 1, \ldots, n\). In constructing a convex nonlinear underestimator for Eq. (9), the nonconvex terms
\[-(b_{i_1}x_1 + \cdots + b_{i_n}x_n)^2, \quad i = p + 1, \ldots, n,
\]
must be replaced by the corresponding linear underestimators. In this case the set of \(n-p\) linear functions (Eq. (13)) with indices \(i = p+1, \ldots, n\) will form the branching set. On the other hand, to construct a linear underestimator for Eq. (9), the corresponding linear underestimators must replace all the quadratic terms. In this case all \(n\) functions in Eq. (13) must be branching ones.

Let us compare the method of constructing linear underestimator and factorizable programming from McCormick (1976, 1983).
(1) In factorable programming, an underestimator of a function is constructed based on its original tree representation (factorable form). For the product \( f(x_i)g(x_i) \) the following nondifferentiable function
\[
L(fg; S_{ij}) = \max (f^U g(x_i) + g^U f(x_i) - f^L g^L, f^U g(x_i) + g^U f(x_i) - f^U g^U)
\] (14)
is used as convex underestimator. To obtain a differentiable analog of this function, McCormick replaces each product term with an additional variable \( w_{ij} \) and two constraints as follows
\[
w_{ij} \geq f^U g(x_i) + g^U f(x_i) - f^U g^U,
\]
\[
w_{ij} \geq f^L g(x_i) + g^L f(x_i) - f^L g^L
\]
Thus the dimension of the lower bound problem can increase significantly.

In contrast in our approach, before constructing an underestimator for a function \( f(x) \), we transform the function to the STF form which does not contain product terms. This permits to construct an underestimator, which is a differentiable function.

(2) When using factorable programming, one needs to use all the original variables and the intermediate variables introduced during reformulation as branching variables (Smith and Pantelides, 1997). Therefore, the number of branching variables and the dimensionality of the lower bound problem can be greater than the dimensionality of the original problem. In contrast to the proposed Sweep method, we use a set of branching functions resulting from the construction of linear underestimators. In solving the molecular design problem, the suggested approach is advantageous (see Section 5). For illustration, consider the nonconvex quadratic programming problem
\[
\min_{x} \left( x^T A x + b^T x + c / B x + C \leq 0 \right)
\]
where \( A \) is \( n \times n \)-matrix. Here there are \( n^2 - n \) products of the form \( x_i x_j \) \((i, j = 1, \ldots, n)\). Therefore, using factorable programming in the lower bound calculation results in a linear program with \( n^2 \) optimization variables and \( 2(n^2 - n) + m \) constraints. In addition, the number of branching variables is \( n \). In our approach, when constructing linear underestimators for the lower bound calculation, we need to solve a linear program with \( n \) optimization variables and \( n + 2p \) constraints (with \( p \leq n \)). As before, the number of branching variables is \( n \). If instead we construct nonlinear convex underestimators, then for the lower bound calculation, we must solve a convex quadratic programming problem with \( n \) optimization variables and \( n + 2p \) constraints. We showed earlier that in this case the number of linear branching expressions in Eq. (13) was equal to \( n - p \), which is less than in the factorable programming case.

(3) The Sweep method obtains a linear underestimator, which is less tight than the corresponding convex nonlinear underestimator obtained through factorable programming. However, the use of linear underestimators for the lower bound problem results in an MILP, which is computationally less intensive than a convex MINLP that results from the use of convex nonlinear underestimators. Granted that one can modify factorable programming to obtain linear underestimators.

5. Molecular design framework

In computer aided solvent design, compounds are formed from certain combinations of a pre-specified set of structural groups. The set of structural groups (with cardinality \( m \)) is called the basis set. Examples of structural groups include \( \text{CH}_3^- \), \( -\text{CH}_2\text{O} \), and \( -\text{OH} \). The selection of the basis set depends on the intended application and the availability of accurate property prediction models. First, we define a set of variables based on an initial set of structural groups
\[
u_{ik} = \begin{cases} 1 & \text{if the } i\text{th group in the molecule is of the} \\
& \text{the } k\text{th structural group in the basis set} \\
0 & \text{otherwise}
\end{cases}
\] (15)

Macchietto et al. (1990) proposed a formulation that ensured that the valence of each structural group was satisfied. This formulation only accounts for the presence and absence of structural groups in the molecule. However, it does not consider the information that determines how the groups are connected to each other in the molecule. To overcome this limitation, Churi and Achenie (1996) proposed a model that results in the formation of singly connected molecules. In this molecular representation, the following new variables were introduced
\[
z_{ij} = \begin{cases} 1 & \text{if the } i\text{th group's } j\text{th site is attached to the} \\
& \text{the } p\text{th group} \\
0 & \text{otherwise}
\end{cases}
\]
\[
w_i = \begin{cases} 1 & \text{if the } i\text{th group in a molecule does not have} \\
& \text{a group attached} \\
0 & \text{otherwise}
\end{cases}
\] (16)

For single component solvents, structural constraints are imposed for: (a) limiting the number of structural groups in a molecule; (b) ensuring that the number of bonds attached to a group equals the valence of the group; and (c) ensuring that only one singly connected molecules are formed. The formulation is effective in
specifying whether the molecule is acyclic or cyclic. Moreover, the maximum number of cycles can also be controlled. This representation is also effective in distinguishing between isomers.

If the chemical process is not accounted for, then the pure component molecular design problem involves only binary variables. Let the maximum number of groups in a molecule be \( n_{\text{max}} \), the number of groups in the basis set be \( m \) and finally let each group have a maximum valence of \( s_{\text{max}} \). Corresponding to the variables in Eqs. (15) and (16) are the vectors \( u \), \( z \), and \( w \) with dimensions \( n_{\text{max}} \times m \), \( n_{\text{max}} \times s_{\text{max}} \times n_{\text{max}} \) and \( n_{\text{max}} \), respectively. The variable \( u \) contains information regarding which structural groups from the basis set are present in the molecule while the variable \( z \) contains the connectivity information. Now define \( v \) to consist of a subset of \( u \), \( z \), and \( w \). Then Table 1 summarizes the dimension of \( v \) depending on what model is employed.

It is clear from Table 1 that \( \text{dim}(v) \) is more sensitive to \( n_{\text{max}} \) than to \( m \) if the Churi and Achenie model is employed. The number of linear constraints depends on which of the two models (namely, Macchietto et al., 1990; Churi and Achenie, 1996) is employed.

We note that the nonlinear property prediction constraints \( g_2(x, v_k, \theta) \) and \( g_3(x, v_k, \theta) \) in \( \text{PMD} \) do not employ the \( z_{ijp} \) and \( w_i \) variables. Thus the problem is nonlinear with respect to only the \( u_{jk} \) variables. Due to the nature of the property prediction models, the nonlinearities are present in all the \( u_{jk} \) variables.

### 6. Solvent design

#### 6.1. Nonconvex constraint in solvent design problem

One very important property for a cleaning solvent is its ability to dissolve the solute. A solute–solvent interaction is often characterized by the Hansen solubility parameter \( \delta_T \) (Archier, 1996). This parameter is characterized by the three intermolecular interactions, namely hydrogen bonding interaction \( (\delta_H) \), polar interactions \( (\delta_P) \) and nonpolar (dispersive) interaction \( (\delta_D) \) (Hansen, 1969). The mathematical expression for the solvent selection criterion based on the Hansen solubility parameter is expressed in Eq. (17).

\[
R^s = 4(\delta_D - \delta_D^s)^2 + (\delta_H - \delta_H^s)^2 \leq (R^s)^2
\]

(17)

where

\[
\delta_D = \frac{\sum_i \sum_j u_{ij} F_{Dij}}{V_0 + \sum_i \sum_j u_{ij} V_j}
\]

\[
\delta_P = \frac{\sqrt{\sum_i \sum_j u_{ij} (F_{ij})^2}}{V_0 + \sum_i \sum_j u_{ij} V_j}
\]
Now we introduce the following functions linear underestimator for the resulting tree function. The constraint (19) is made up of four separable terms. The second term is constructed to be a strict upper bound. The branching set \( \Phi_1^{(3)} \) contains \( \psi_1^1, \psi_2^1, \psi_3^1, \psi_4^1, \psi_1^2, \psi_2^2, \psi_3^2, \psi_4^2 \). The range of the functions \( \psi_1^2, \psi_2^2, \psi_3^2, \psi_4^2 \) is used to construct the range of functions (variables) at the second level, \( (\psi_1^2, \psi_2^2, \psi_3^2, \psi_4^2) \). These are then used to construct the range of functions (variables) at the third level, \( (\psi_1^3, \psi_2^3, \psi_3^3, \psi_4^3) \). Thus the bounds are estimated in a bottom up sweep.

The linear underestimators are constructed in a reverse sweep that starts at the fifth level and goes down. First, a linear underestimator of \( g_1^4 \) is constructed in terms of \( g_1^4 \) as follows

\[
L[g_1^4, S_1^4] = \mu_1(g_1^4) + \mu_2.
\]

Here \( S_1^4 = [g_1^4, g_2^4] \), the sign of \( \mu_1 \), which depends on the interval \( (g_1^4, g_2^4) \), determines whether the function \( \psi_1^4 \) is convex or concave with respect to variables \( g_1^4 \) and \( g_2^4 \). The underestimator now has the following form

\[
L[g_1^4, S_1^4] = \mu_3(g_1^4) + \mu_4(g_2^4) + \mu_5(\psi_V) + \mu_6
\]

or

\[
L[g_1^4, S_1^4] = (\mu_3 - \mu_4)(\sqrt{\psi_V})
\]

The signs of \( \mu_3 - \mu_4 \) and \( \mu_3 + \mu_4 \) determine if the corresponding functions are concave or convex. Subsequently, the underestimator is constructed with respect to \( \psi_1^1 \) and \( \psi_1^2 \). After rearranging the terms, the linear underestimator is represented as

\[
L[g_1^4, S_1^4] = \mu_5(\psi_1) + \mu_6(\psi_1) + \mu_10
\]

We reiterate that the subregion is not in terms of the
search variables but rather in terms of functions of the search variables. Based on the region the coefficients \((\mu_3), (\mu_1-\mu_4)\) and \((\mu_3+\mu_4)\) are calculated and a decision about construction of the underestimator is made at two levels as shown in Fig. 2. This makes the algebraic structure of the underestimator adaptive.

The goal of this case study is to design optimal solvents to be used as cleaning agents in the printing industry. These solvents should: (i) have a minimal drying time; (ii) dissolve residue ink; (iii) not swell the blanket; and (iv) be environmentally benign. Drying time is correlated with the heat of vaporization of the solvent. The ink residue is assumed to consist of phenolic resins. To obtain solvents that can effectively dissolve the ink residue, we employ the following solute–solvent interaction constraint:

\[
R^d = (4(\delta_D - 23.3)^2 + (\delta_P - 6.6)^2 + (\delta_H - 8.3)^2)^{1/2} < 19.8
\]  
(21)

Also the constraint \(\delta_P \geq 6.3\) will ensure minimal blanket swelling.

The environmental impact of solvents is accounted for by requiring that the maximum value of the partition coefficient (log \(K_{ow}\)) be 4.0. To ensure that the solvent is a liquid at ambient temperature, the limits on boiling point \((T_b)\) and melting point \((T_m)\) are imposed. We choose as basis set 12 groups, namely \(\text{CH}_3\), \(\text{CH}_2\), \(\text{Ar}\), \(-\text{Ar}\), \(-\text{OH}\), \(\text{CH}_3\text{CO}\), \(-\text{CH}_2\text{CO}\), \(-\text{COOH}\), \(\text{CH}_3\text{COO}\), \(-\text{CH}_2\text{COO}\), \(-\text{CH}_3\text{O}\), and \(-\text{CH}_2\text{O}\). Constraints

\[
\sum_{j} u_j \leq 2 \quad \forall j = 5, 6, \ldots, 12
\]  
(22)

are employed in order to ensure that the last seven groups are not allowed to occur more than twice in a compound.

The heat of vaporization, boiling point and melting point solvent properties are calculated using the Constantinou and Gani method (Constantinou and Gani, 1994). The fragment-based method is used to calculate \(K_{ow}\) (Lyman et al., 1981). The group contribution parameter for solubility parameter calculation is from van Krevelen and Hoftyzer’s method (Barton, 1985). The models and their references are summarized in Table 3.

### 6.3. Case study 1

In this case study (Sinha, 1999), a simplified structural feasibility constraint is employed. Structural constraints are employed to ensure feasible molecular structures. The resulting molecular design formulation is shown below.
Table 4
Application of reduced space BB algorithm to CAMD_1

<table>
<thead>
<tr>
<th>Case</th>
<th>$n_{\text{max}}$</th>
<th>Variables</th>
<th>Constraints</th>
<th>Iterations</th>
<th>CPU time (min)</th>
<th>Maximum number of subregions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMD_1a</td>
<td>3</td>
<td>36</td>
<td>15</td>
<td>1</td>
<td>0.045</td>
<td>1</td>
</tr>
<tr>
<td>CAMD_1b</td>
<td>4</td>
<td>48</td>
<td>15</td>
<td>18</td>
<td>0.86</td>
<td>12</td>
</tr>
<tr>
<td>CAMD_1c</td>
<td>5</td>
<td>60</td>
<td>15</td>
<td>31</td>
<td>5.85</td>
<td>16</td>
</tr>
<tr>
<td>CAMD_1d</td>
<td>6</td>
<td>72</td>
<td>15</td>
<td>42</td>
<td>17.21</td>
<td>20</td>
</tr>
<tr>
<td>CAMD_1e</td>
<td>7</td>
<td>84</td>
<td>15</td>
<td>46</td>
<td>48.45</td>
<td>21</td>
</tr>
<tr>
<td>CAMD_1f</td>
<td>10</td>
<td>120</td>
<td>15</td>
<td>67</td>
<td>713.5</td>
<td>21</td>
</tr>
</tbody>
</table>

CAMD_1:

minimise \[ \sum_i \sum_j u_j(\Delta H_v) \]

subject to:

\[ \sum_i \sum_j u_j \leq n_{\text{max}} \]

\[ \sum_i \sum_j u_j(2 - \gamma_i) = 2 \]

\[ \exp \left( \frac{\sum_i \sum_j u_j(T_b)}{204.4} \right) \geq 323 \]

\[ \exp \left( \frac{\sum_i \sum_j u_j(T_w)}{102.425} \right) \leq 223 \]

\[ \sum_i \sum_j u_j(\chi^0) + \sum_i \sum_j u_j(\chi^1) \leq 4.0 \]

\[ 4 \left( \frac{\sum_i \sum_j u_j^* F_{D_i}}{V_0 + \sum_i \sum_j u_j V_j} \right)^2 \geq 23.3 \]

\[ + \left( \frac{\sum_i \sum_j u_j^* (F_{P_i})}{V_0 + \sum_i \sum_j u_j V_j} \right)^2 \geq 6.6 \]

\[ + \left( \frac{\sum_i \sum_j u_j^* (-U_{H_i})}{V_0 + \sum_i \sum_j u_j V_j} \right)^2 \leq 8.3 \leq (19.8)^2 \]

where $u_j$ are binary variables.

CAMD_1 is solved using the global optimization (i.e. BB) algorithm. There are two nonconvex constraints. The splitting functions employed are $\psi_D, \psi_P, \psi_H$ and $\psi_V$. Therefore, the subregion is further constrained by four pairs of linear constraints that confine the search between the maximum and minimum values of these splitting functions. The MILP solver used is a public domain code lp_solve by Hartmut Schwab available at ftp.es.ele.tue.nl/pub/lp_solve. This solver uses the simplex algorithm. lp_solve uses a rather simple depth first strategy; thus, it can be computationally inefficient when the problem size increases. Three different runs were investigated for case study 1. The three runs correspond to $n_{\text{max}}$ of 3, 4, 5, 6, 7, and 10 (CAMD_1a, CAMD_1b, CAMD_1c, CAMD_1d, CAMD_1e, and CAMD_1f, respectively). The corresponding problem dimensions are 36, 48, 60, 72, 84 and 120. For all cases the number of constraints are 15. The termination criterion used is an absolute tolerance of $10^{-3}$. The results are shown in Table 4.

Problem CAMD_1a has a very limited search space. A feasible solution was found in the first iteration in the BB algorithm. In CAMD_1c, the algorithm took 31 iterations and 351.4 s on a 333-MHz DELL Pentium II.
personal computer. The maximum number of subregions constructed is 16. The globally optimal solution corresponded to methyl-ethyl ketone (MEK or CH₃/CO/CH₂/CH₃) with objective function 35.471 kJ/mol. This compound was found at the 10th iteration with a valid upper bound of 35.471 and a lower bound of 33.99. Since the difference between the upper and lower bound was more than the tolerance, the algorithm continued executing. The algorithm finally converged to MEK as the global solution after 21 more iterations. The two other feasible compounds found were propanol (CH₃/CH₂/CH₂/OH) and diethyl-ketone (CH₃/CH₂/CO/CH₂/CH₃). The objective function values for propanol and diethyl-ketone were 44.77 and 40.12 kJ/mol, respectively.

We note that at any iteration, the solution of the relaxed MILP problem is a structurally feasible compound since all the structural constraints are linear. During the execution of the algorithm, 15 different compounds were found. Of these, two other compounds satisfied the specified performance constraints. The same formulation was rerun with an additional constraint to remove MEK from the design space. The optimal compound is the same compound that was designed in the 17th iteration during the first run (CH₃/CH₂/CO/CH₂/CH₃).

For case CAMD_1e, the number of iterations is 46 and three compounds are designed. The maximum number of subregions created is 21. In CAMD_1f, the number of iterations is 67. The maximum number of subregions created is 21. Even though the number of iterations does not grow very much, the CPU time increases. This is because the CPU time associated with each LP solution increases significantly when the number of variables increases. A desirable property of this algorithm is that a very small number of subregions are created. For the three cases, the number of subregions created is 16, 21 and 21. Thus the algorithm is very efficient in terms of storage requirements. It should be noted that as the dimension of the problem increases from 60 to 120, the number of iterations only increases from 31 to 67. This increase is linear in nature as shown in Fig. 3.

It is interesting to note that if we employ the standard full space BB algorithm, we will need to perform branching with respect to all the variables $u_i$. Here, the number of branching variables ranges from 60 to 120.

### 6.4. Case study 2

In this case study, CAMD_2 (Sinha, 1999), the same formulation is solved with the complete structural representation. The connectivity variables $z$ and $w$ are employed in the structural representation as described in Section 5. The second constraint in CAMD_1 is replaced by the following set of structural constraints (Churi and Achenie, 1996). This leads to a large increase in the number of linear structural constraints.

$$\sum_{p=1}^{m} \sum_{j=1}^{l_{\text{max}}} z_{ijp} = \sum_{k=1}^{l_{\text{max}}} u_{ik} v_k$$  \hspace{1cm} (24) \\
$$\sum_{j=1}^{l_{\text{max}}} \sum_{p=1}^{m} z_{ijp} > -w_i$$  \hspace{1cm} (25) \\
$$\sum_{l=1}^{n_{\text{max}}} \sum_{k=1}^{l} u_{ik} + \sum_{l=1}^{n_{\text{max}}} w_i = n_{\text{max}}$$  \hspace{1cm} (26) \\
$$w_1 = 0$$  \hspace{1cm} (27) \\
$$w_i \leq w_{i+1}, \hspace{0.5cm} i = 1, \ldots, (n_{\text{max}} - 1)$$  \hspace{1cm} (28) \\
$$\sum_{j=v_i+1}^{n_{\text{max}}} \sum_{p=1}^{m} z_{ijp} + M u_{ik} \leq M,$$  \hspace{1cm} (29) \\
\hspace{1cm} $i = 1, \ldots, n_{\text{max}}, \hspace{0.5cm} k = 1, \ldots, m$

**Table 5**

<table>
<thead>
<tr>
<th>Case</th>
<th>$n_{\text{max}}$</th>
<th>Variables</th>
<th>Constraints</th>
<th>Iterations</th>
<th>CPU time (min)</th>
<th>Maximum number of subregions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMD_2a</td>
<td>3</td>
<td>57</td>
<td>67</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>CAMD_2b</td>
<td>4</td>
<td>89</td>
<td>89</td>
<td>18</td>
<td>3.36</td>
<td>9</td>
</tr>
<tr>
<td>CAMD_2c</td>
<td>5</td>
<td>115</td>
<td>113</td>
<td>22</td>
<td>14.5</td>
<td>11</td>
</tr>
</tbody>
</table>
The dimension of $u$ compound found in an intermediate step is $22$. For the run CAMD_2c one of the feasible variables, the number of constraints is $67$, $84$ and $113$. The corresponding number of constraints is $67$, $84$ and $113$. Note that the formulation is nonlinear with respect to only the $u_{ik}$ variables. The results are summarized in Table 5.

This formulation is solved for $n_{\text{max}}$ equal to $3$, $4$ and $5$ (CAMD_2a, CAMD_2b, and CAMD_2c). The number of search variables is $57$, $84$ and $115$, respectively. The number of nodes generated in case study $2$ is much smaller compared to problems of similar dimension in CAMD_1.

For about the same number of variables, the number of iterations in case study $2$ (CAMD_2) is relatively smaller than case study $1$. In addition, the maximum number of nodes generated in case study $2$ is much smaller that in case study $1$. This can be attributed to the fact that in CAMD_2 the number of variables appearing in nonlinear term is much smaller compared to problems of similar dimension in CAMD_1.

6.5. Case study 3

In this case study (Sinha, 1999), solvents are designed with an entirely different criteria. Here the most desirable attribute of the solvent is recoverability. That is, after the blanket wash operation is performed the solvent that evaporates is recovered by a solvent recovery system (see Fig. 4). This case study attempts to find a solvent compound that will be least expensive to recover. Many competing solvent recovery techniques can be applied, namely condensation, gas adsorption and gas absorption. Here the recovery system is restricted to the condensation.

A typical condensation recovery system consists of a compressor that intakes the printing solvent-laden exhaust gases (from the ventilation system) and compresses them to a higher pressure. These high-pressure gases are passed through a condenser that cools this stream. Next, it is flashed to recover the solvent. The details of the recovery operation have been discussed elsewhere (Sinha, 1999). Here the objective is to find the solvent compound that will have minimal total annualized cost (TAC) associated with recovery. The CAMD case study with recovery considerations is:

$$\text{minimize } TAC$$

$$= 85675\times(p_{\text{comp}}^{0.284} - 1) + 99.03(298 \times p_{\text{comp}}^{0.284} - T_{\text{cond}}) + 9.69 \times 10^{-3} (\Delta H_{V} / V_{M})$$

subject to:

$$\sum_{i} \sum_{j} u_{ij} \leq 4 \quad \text{(molecular size constraint)}$$

$$\sum_{i} \sum_{j} u_{ij}(2 - \text{val}_j) = 2 \quad \text{(Odele’s octet rule implementation)}$$
log₁₀(V_m) − log₁₀(P_comp) − 2.7 \left( \frac{T_b}{T_{cond}} \right)^{1.7} \leq -11.47

(recovery constraint)

\exp \left( \sum_{i} \sum_{j} u_{ij}T_{mj} \right) / 102.425 \leq 223

(melting point constraint)

500 \leq T_b + \sum_{i} \sum_{j} u_{ij}T_{bj} \leq 700

(boiling point constraint, 1 and 2)

20 \leq H_{v0} + \sum_{i} \sum_{j} u_{ij}H_{vj} \leq 80

(heat of vaporization constraints, 1 and 2)

\sum_{i} \sum_{j} u_{ij}(\chi_i^o) + \sum_{i} \sum_{j} u_{ij}(\chi_i^1) \leq 4.0

(Octanol–water partition)

4 \left( \frac{\sum_{i} \sum_{j} u_{ij}^o F_{Dj}}{V_0 + \sum_{i} \sum_{j} u_{ij}V_j} - 23.3 \right) + \frac{\left( \sqrt{\sum_{i} \sum_{j} u_{ij}^o (F_{pj})^2} \right)^2}{V_0 + \sum_{i} \sum_{j} u_{ij}V_j} - 6.6 \right)^2 + \frac{\left( \sqrt{\sum_{i} \sum_{j} u_{ij}^o (-u_{ij})} \right)^2}{V_0 + \sum_{i} \sum_{j} u_{ij}V_j} - 8.3 \right)^2 \leq (19.8)^2

(solvent power)

\sqrt{\sum_{i} \sum_{j} u_{ij}^o (F_{pj})^2} \geq 6.3

(swelling constraint)

\tilde{\psi}_D \leq \sum_{i} \sum_{j} u_{ij}^o F_{Dj} \leq \tilde{\psi}_D

(splitting function 1)

\tilde{\psi}_P \leq \sum_{i} \sum_{j} u_{ij}^o (F_{pj})^2 \leq \tilde{\psi}_P

(splitting function 2)

\tilde{\psi}_H \leq \sum_{i} \sum_{j} u_{ij}^o (-u_{ij}) \leq \tilde{\psi}_H

(splitting function 3)

\tilde{\psi}_v \leq V_0 + \sum_{i} \sum_{j} u_{ij}V_{j} \leq \tilde{\psi}_v

(splitting function 4)

\tilde{\psi}_s \leq H_{v0} + \sum_{i} \sum_{j} u_{ij}H_{vj} \leq \tilde{\psi}_s

(splitting function 5)

\tilde{\psi}_6 \leq V_0 + \sum_{i} \sum_{j} u_{ij}H_{vj} \leq \tilde{\psi}_6

(splitting function 6)

\tilde{\psi}_7 \leq P_{comp} \leq \tilde{\psi}_7

(splitting function 7)

\tilde{\psi}_8 \leq T_{cond} \leq \tilde{\psi}_8

(splitting function 8)

Note that val_j is the valence of the structural group. The following modified basis with 15 groups is used in this study:

- \text{CH}_3, \text{CH}_2, \text{-OH}, \text{CH}_3\text{CO}-, \text{-CH}_2\text{CO}-, \text{-COOH}, \text{CH}_3\text{COO}-, \text{-CH}_2\text{COO}-,

- \text{-CH}_3\text{O}, \text{-CH}_2\text{O}-, \text{CH}_2=\text{CH}-, \text{-CH}-\text{-CH}-, \text{-CH}_2\text{NH}_2, \text{=CHNH}_2, \text{CH}_3\text{NH}-

The aromatic groups are removed and a couple or groups with nitrogen are added to include amine or other compounds with nitrogen.

There is a total of eight splitting functions. The last four splitting functions are used for construction of linear underestimators for the objective function and under- and overestimator for the recovery constraint. This case study has 60 variables and three nonlinear constraints. Moreover, the objective function is nonlinear.

All the \textit{u} variables are binary and range between 0 and 1. The condenser temperature however can range between 150 and 298 K. This results in poor scaling and causes difficulty during optimization. To overcome this we have scaled the condenser temperature between 0.1 and 0.9 such that \textit{T' = 185T_{cond} +131.5} where \textit{T'} is the scaled condenser temperature.

The globally optimal compound designed is a di-ester with the structure \text{CH}_3(\text{CH}_2\text{COO})_2\text{-CH}_2\text{NH}_2. The recovery cost associated with this compound is $25,981. The corresponding compressor pressure is 2 atm and the condenser temperature is 288.75 K. The algorithm took 56 iterations and a CPU time of 41.6 s. At termination, the number of nodes (i.e. subregions) is 20.
The above problem was solved again with DICOPT in GAMS (Brooke et al., 1996) environment. DICOPT is a local optimization software that uses a modified outer approximation algorithm with augmented penalty function and equality relaxation (OA-AP-ER, Viswanathan and Grossmann, 1990). The optimal compound found by DICOPT is HO–CH$_2$COO–CH$_3$NH. This is also an ester. The objective function associated with this compound is 106327.799, the compressor pressure is 10 atm and condenser temperature is 298 K. It is interesting to note that the extra effort associated with the global optimization is justified and results in almost four times reduction in the recovery cost.

Next, the second and third best compounds are designed by adding additional constraints to eliminate the globally optimal compound from the search space. Here an integer cut is implemented that restricts the design of compounds with the same structure as the globally optimal compound. The second best compound is also a di-ester with the structure: CH$_3$−(CH$_2$COO)$_2$−OH. The recovery cost associated with this compound is marginally higher than the first compound, at $25 982.54 per year. The algorithm took 64 iterations and CPU time of 105 s. At the termination point, there was a total of 21 nodes. The third best compound is a tri-ester with structure CH$_3$COO–CH$_2$COO–CH$_3$COO. The recovery cost for this compound is considerably higher at $28 729.57 per year. The condenser temperature here is 261 K.

7. Discussion and conclusion

The molecular design problem is reduced to solving an MINLP problem in which the number of binary variables can range from several tens to several hundreds. The use of the standard BB method for solving the problem can be computationally intensive since all the binary variables must be used as branching variables. To overcome this problem, we have proposed a new strategy. The main idea of the method consists in that we do branching using branching functions instead of all the search variables. This approach results in a decrease in the number of branching variables in our molecular design framework. For example, in case study 1, a problem with 120 nonlinear variables is solved with just four splitting variables. This is also demonstrated in the case studies. The maximum number of nodes stored in memory during the search is 21 for CAMD_1e and CAMD_1f and 20 for CAMD_3.

In other words, during BB, the bounding operation is performed in the dimension defined by the search variables, while the branching operation is performed in a reduced dimension defined by the branching (or splitting) functions.

The branching functions are determined from the special tree function representation of both the objective function and constraints. In order to construct the corresponding linear underestimators, we have developed the Sweep method. The proposed algorithm scales well. Specifically, as the problem size increases, the computational effort increases almost linearly (Fig. 3).

The public domain software LP solver lp_solve used in this work is computationally slow when the problem size increases. For every iteration in BB, two MILP problems are solved using lp_solve. Thus most of the CPU time is attributable to the LP solver. The solver will be replaced by a more computationally efficient solver such as OSL or CPLEX. This will not affect the number of iterations that the proposed reduced dimension BB algorithm will take, but it will significantly improve the CPU time.

References


