IEEE Transactions On Industrial Informatics, Vol.6(1), 2010, p.54-61

Bidirectional Branch and Bound for Controlled Variable Selection Part III: Local Average Loss Minimization

Vinay Kariwala[†] and Yi Cao[‡*]

[†] Division of Chemical & Biomolecular Engineering,
Nanyang Technological University, Singapore 637459

[‡]School of Engineering, Cranfield University, Cranfield, Bedford MK43 0AL, UK

Abstract

The selection of controlled variables (CV) from available measurements through exhaustive search is computationally forbidding for large-scale problems. We have recently proposed novel bidirectional branch and bound (B^3) approaches for CV selection using the minimum singular value (MSV) rule and the local worst-case loss criterion in the framework of self-optimizing control. However, the MSV rule is approximate and worst-case scenario may not occur frequently in practice. Thus, CV selection through minimization of local average loss can be deemed as most reliable. In this work, the B^3 approach is extended to CV selection based on the recently developed local average loss metric. Lower bounds on local average loss and, fast pruning and branching algorithms are derived for the efficient B^3 algorithm. Random matrices and binary distillation column case study are used to demonstrate the computational efficiency of the proposed method.

Keywords: Branch and bound, Control structure design, Controlled variables, Combinatorial optimization, Self-optimizing control.

1 Introduction

Control structure design deals with the selection of controlled and manipulated variables, and the pairings interconnecting these variables. Among these tasks, the selection of controlled variables (CVs) from available measurements can be deemed to be most important. Traditionally, CVs have been selected

*Corresponding Author: Tel: +44-1234-750111; Fax: +44-1234-754685; E-mail:y.cao@cranfield.ac.uk

based on intuition and process knowledge. To systematically select CVs, Skogestad [16] introduced the concept of self-optimizing control. In this approach, CVs are selected such that in presence of disturbances, the loss incurred in implementing the operational policy by holding the selected CVs at constant setpoints is minimal, as compared to the use of an online optimizer.

The choice of CVs based on the general non-linear formulation of self-optimizing control requires solving large-dimensional non-convex optimization problems [16]. To quickly pre-screen alternatives, local methods have been proposed including the minimum singular value (MSV) rule [17] and exact local methods with worst-case [7] and average loss minimization [13]. Though the local methods simplify loss evaluation for a single alternative, every feasible alternative still needs to be evaluated to find the optimal solution. As the number of alternatives grows rapidly with process dimensions, such an exhaustive search is computationally intractable for large-scale processes. Thus, an efficient method is needed to find a subset of available measurements, which can be used as CVs (Problem 1).

Instead of selecting CVs as a subset of available measurements, it is possible to obtain lower losses using linear combinations of available measurements as CVs [7]. Recently, explicit solutions to the problem of finding locally optimal measurement combinations have been proposed [1, 8, 11, 13]. It is, however, noted in [1, 11, 13] that the use of combinations of a few measurements as CVs often provide similar loss as the case where combinations of all available measurements are used. Though the former approach results in control structures with lower complexity, it gives rise to another combinatorial optimization problem involving the identification of the set of measurements, whose combinations can be used as CVs (Problem 2).

Both Problems 1 and 2 can be seen as subset selection problems, for which only exhaustive search and branch and bound (BAB) method guarantee globally optimal solution [4]. A BAB approach divides the problem into several sub-problems (nodes). For minimization problems, lower bounds are computed on the selection criteria for all solutions with target subset size, which can be reached from the node under consideration. Subsequently, the current node is pruned (eliminated without further evaluation), if the computed lower bound is greater than an upper bound on the optimal solution (usually taken as the best known solution). The pruning of nodes allows the BAB method to gain efficiency in comparison with exhaustive search. The traditional BAB methods for subset selection use downwards approach, where pruning is performed on nodes with gradually decreasing subset size [3, 4, 14, 18, 19]. Recently, a novel bidirectional BAB (B^3) approach [2] has been proposed for CV selection, where non-optimal nodes are pruned in downwards as well as upwards (gradually increasing subset size) directions simultaneously, which significantly reduces the solution time.

The bidirectional BAB (B^3) approach has been applied to solve Problem 1 with MSV rule [2] and local worst-case loss [12] as selection criteria. A partially bidirectional BAB (PB^3) method has also been proposed to solve Problem 2 through minimization of local worst-case loss [12]. The MSV rule, however, is approximate and can lead to non-optimal set of CVs [9]. Selection of CVs based on local worst-case loss minimization can also be conservative, as the worst-case may not occur frequently in practice [13]. Thus, CV selection through minimization of local average loss, which represents the expected loss incurred over the long-term operation of the plant, can be deemed as most reliable.

In this paper, a B^3 method for solving Problem 1 through minimization of local average loss is proposed. Upwards and downwards lower bounds on the local average loss and fast pruning algorithms are derived to obtain an efficient B^3 algorithm. The downwards lower bound derived for Problem 1 can also be used for pruning non-optimal nodes for Problem 2 with local average loss minimization. The upwards lower bound for Problem 2, however, only holds when the number of elements of the node under consideration exceeds a certain number. To realize the advantages of bidirectional BAB method to some extent, we develop a PB^3 method for selection of measurements, whose combination can be used as CVs to minimize local average loss. Random matrices and binary distillation column case study are used to demonstrate the computational efficiency of the proposed method.

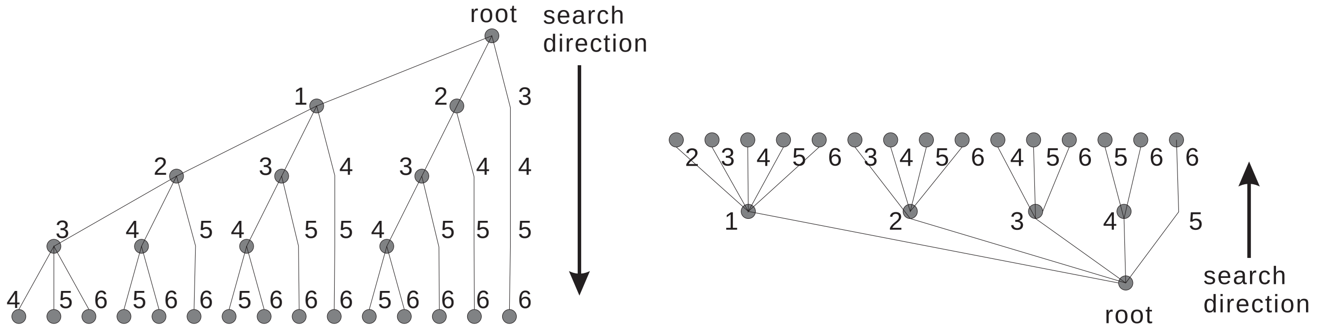
The rest of this paper is organized as follows: a tutorial overview of the unidirectional and bidirectional BAB methods for subset selection problems is given in Section 2. The local method for self-optimizing control is described and the CV selection problems through local average loss minimization are posed in Section 3. The B^3 methods for solving these CV selection problems are presented in Section 4 and their numerical efficiency is demonstrated in Section 5. The work is concluded in Section 6.

2 BAB Methods for Subset selection

Let $X_m = \{x_i\}$, $i = 1, 2, \cdots, m$, be an m-element set. The subset selection problem with selection criterion T involves finding an n-element subset $X_n \subset X_m$ such that

$$T(X_n^*) = \min_{X_n \subset X_m} T(X_n) \tag{1}$$

For a subset selection problem, the total number of candidates is C_m^n, which can be extremely large for large m and n rendering exhaustive search unviable. Nevertheless, BAB approach can find the globally optimal subset without exhaustive search.



(a) downward search (b) upward search

Figure 1: Solution trees for selecting 2 out of 6 elements.

2.1 Unidirectional BAB approaches

Downwards BAB method. BAB search is traditionally conducted downwards (gradually decreasing subset size) [3, 4, 14, 18, 19]. A downwards solution tree for selecting 2 out of 6 elements is shown in Figure 1(a), where the root node is same as X_m. Other nodes represent subsets obtained by eliminating one element from their parent sets. Labels at nodes denote the elements discarded there.

To describe the pruning principle, let B be an upper bound of the globally optimal criterion, *i.e.* $B \geq T(X_n^*)$ and $\underline{T}_n(X_s)$ be a downwards lower bound over all n-element subsets of X_s, *i.e.*

$$\underline{T}_n(X_s) \leq T(X_n) \quad \forall X_n \subseteq X_s \tag{2}$$

If $\underline{T}_n(X_s) > B$, it follows that

$$T(X_n) > T(X_n^*) \quad \forall X_n \subseteq X_s \tag{3}$$

Hence, all n-element subsets of X_s cannot be optimal and can be pruned without further evaluation.

Upwards BAB method. Subset selection can also be performed upwards (gradually increasing subset size) [12]. An upwards solution tree for selecting 2 out of 6 elements is shown in Figure 1(b), where the root node is an empty set. Other nodes represent supersets obtained by adding one element to their parent sets. Labels at nodes denote the elements added there.

To introduce the pruning principle, let the upwards lower bound of the selection criterion be defined as

$$\underline{T}_n(X_t) \leq T(X_n) \quad \forall X_n \supseteq X_t \tag{4}$$

Similar to downwards BAB, if $\underline{T}_n(X_t) > B$,

$$T(X_n) > T(X_n^*) \quad \forall X_n \supseteq X_t \tag{5}$$

Hence, all n-element supersets of X_t cannot be optimal and can be pruned without further evaluation.

Remark 1 (Monotonicity) *When T is upwards (downwards) monotonic, $T(X_s)$ with $s < n$ ($s > n$), can be used as the upwards (downwards) lower bound on T for pruning [2]. Although, monotonicity simplifies lower bound estimation, it is not a prerequisite for the use of BAB methods and the availability of any lower bound suffices.*

2.2 Bidirectional BAB approach

The upwards and downwards BAB approaches can be combined to form a more efficient bidirectional BAB (B^3) approach. This approach is applicable to any subset selection problem, for which both upwards and downwards lower bounds on the selection criterion are available [2].

Bidirectional pruning. In a B^3 approach, the whole subset selection problem is divided into several subproblems. A sub-problem is represented as the 2-tuple $\mathcal{S} = (F_f, C_c)$, where F_f is an f-element fixed set and C_c is a c-element candidate set. Here, $f \leq n$ and $n \leq f + c \leq m$. The elements of F_f are included in all n-element subsets that can be obtained by solving $\mathcal{S}$, while elements of C_c can be freely chosen to append F_f. In terms of fixed and candidate sets, downwards and upwards pruning can be performed if $\underline{T}_n(F_f \cup C_c) > B$ and $\underline{T}_n(F_f) > B$, respectively. In B^3 approach, these pruning conditions are used together (bidirectional pruning), where the subproblem $\mathcal{S}$ is pruned, if either downwards or upwards pruning condition is met.

The use of bidirectional pruning significantly improves the efficiency as non-optimal subproblems can be pruned at an early stage of the search. Further gain in efficiency is achieved by carrying out pruning on the sub-problems of $\mathcal{S}$, instead of on $\mathcal{S}$ directly. For $x_i \in C_c$, upward pruning is conducted by discarding x_i from C_c, if $\underline{T}_n(F_f \cup x_i) > B$. Similarly, if $\underline{T}_n(F_f \cup (C_c \backslash x_i)) > B$, then downward pruning is performed by moving x_i from C_c to F_f. Here, an advantage of performing pruning on sub-problems is that the bounds $\underline{T}_n(F_f \cup x_i)$ and $\underline{T}_n(F_f \cup (C_c \backslash x_i))$ can be computed from $\underline{T}_n(F_f)$ and $\underline{T}_n(F_f \cup C_c)$, respectively, for all $x_i \in C_c$ together, resulting in computational efficiency.

Bidirectional branching. In downwards and upwards BAB methods, branching is performed by removing elements from C_c and moving elements from C_c to F_f, respectively. These two branching approaches can be combined into an efficient bidirectional approach by selecting a decision element and deciding upon whether the decision element be eliminated from C_c or moved to F_f.

In B^3 algorithm, the decision element is selected as the one with the smallest upwards or downwards lower bound for upward or downward (best-first) search, respectively. To select the branching direction, we note that downwards and upwards branching result in subproblems with $\mathcal{C}_{c-1}^{n-f}$ and $\mathcal{C}_{c-1}^{n-f-1}$ terminal nodes, respectively. In B^3 algorithm, the simpler branch is evaluated first, *i.e.* downwards branching is performed, if $\mathcal{C}_{c-1}^{n-f} > \mathcal{C}_{c-1}^{n-f-1}$ or $2(n-f) > c$, otherwise upwards branching is selected.

3 Local method for Self-optimizing control

To present the local method for self-optimizing control, consider that the economics of the plant is characterized by the scalar objective function $J(\mathbf{u}, \mathbf{d})$, where $\mathbf{u} \in \mathbb{R}^{n_u}$ and $\mathbf{d} \in \mathbb{R}^{n_d}$ denote the degrees of freedom or inputs and disturbances, respectively. Let the linearized model of the process around the nominally optimal operating point be given as

$$\mathbf{y} = \mathbf{G}^y\,\mathbf{u} + \mathbf{G}_d^y\,\mathbf{W}_d\,\mathbf{d} + \mathbf{W}_e\,\mathbf{e} \tag{6}$$

where $\mathbf{y} \in \mathbb{R}^{n_y}$ denotes the process measurements and $\mathbf{e} \in \mathbb{R}^{n_y}$ denotes the implementation error, which results due to measurement and control error. Here, the diagonal matrices $\mathbf{W}_d$ and $\mathbf{W}_e$ contain the magnitudes of expected disturbances and implementation errors associated with the individual measurements, respectively. The CVs $\mathbf{c} \in \mathbb{R}^{n_u}$ are given as

$$\mathbf{c} = \mathbf{H}\,\mathbf{y} = \mathbf{G}\,\mathbf{u} + \mathbf{G}_d\,\mathbf{W}_d\,\mathbf{d} + \mathbf{H}\,\mathbf{W}_e\,\mathbf{e} \tag{7}$$

where

$$\mathbf{G} = \mathbf{H}\,\mathbf{G}^y \quad \text{and} \quad \mathbf{G}_d = \mathbf{H}\,\mathbf{G}_d^y \tag{8}$$

It is assumed that $\mathbf{G} = \mathbf{H}\,\mathbf{G}^y \in \mathbb{R}^{n_u \times n_u}$ is invertible. This assumption is necessary for integral control.

When $\mathbf{d}$ and $\mathbf{e}$ are uniformly distributed over the set

$$\left\| \begin{bmatrix} \mathbf{d}^T & \mathbf{e}^T \end{bmatrix} \right\|_2^T \leq 1 \tag{9}$$

the average loss is given as [13]

$$L_{\text{average}}(\mathbf{H}) = \frac{1}{6(n_y + n_d)} \left\| (\mathbf{H}\tilde{\mathbf{G}})^{-1}\mathbf{H}\mathbf{Y} \right\|_F^2 \tag{10}$$

where $\|\cdot\|_F$ denotes Frobenius norm and

$$\begin{aligned}\tilde{\mathbf{G}} &= \mathbf{G}^y \mathbf{J}_{uu}^{-1/2} &(11)\\ \mathbf{Y} &= \left[\,(\mathbf{G}^y\,\mathbf{J}_{uu}^{-1}\,\mathbf{J}_{ud} - \mathbf{G}_d^y)\,\mathbf{W}_d \quad \mathbf{W}_e\,\right] &(12)\end{aligned}$$

Here, $\mathbf{J}_{uu}$ and $\mathbf{J}_{ud}$ represent $\frac{\partial^2 J}{\partial u^2}$ and $\frac{\partial^2 J}{\partial u\,\partial d}$, evaluated at the nominally optimal operating point, respectively.

When individual measurements are selected as CVs, the elements of $\mathbf{H}$ are restricted to be 0 or 1 and

$$\mathbf{H}\mathbf{H}^T = \mathbf{I} \tag{13}$$

In words, selection of a subset of available measurements as CVs involves selecting n_u among n_y measurements, where the number of available alternatives is $\mathcal{C}_{n_y}^{n_u}$. Using index notation, this problem can be stated as

$$\min_{X_{n_u} \subset X_{n_y}} L_1(X_{n_u}) = \left\| \tilde{\mathbf{G}}_{X_{n_u}}^{-1}\, \mathbf{Y}_{X_{n_u}} \right\|_F^2 \tag{14}$$

In this work, we assume both n_y and n_d are given. Hence, the scalar $1/(6(n_y+n_d))$ is constant and is neglected in (14), as it does not depend on the selected CVs. Instead of 2-norm, as used in (9), if a different norm is used to define the allowable set of $\mathbf{d}$ and $\mathbf{e}$, the resulting expressions for average losses only differ by scalar constants [13]. Thus, the formulation of optimization problem in (14) is independent of the norm used to define the allowable set of $\mathbf{d}$ and $\mathbf{e}$.

Instead of using individual measurements, it is possible to use linear combinations of measurements as CVs. In this case, the integer constraint on $\mathbf{H} \in \mathbb{R}^{n_u \times n_y}$ is relaxed, but the condition $\mathrm{rank}(\mathbf{H}) = n_u$ is still imposed to ensure invertibility of $\mathbf{H}\,\mathbf{G}^y$. The minimal average loss over the set (9) using measurements combinations as CVs is given as [13]

$$\min_{\mathbf{H}} L_{\text{average}} = \frac{1}{6\,(n_y+n_d)} \sum_{i=1}^{n_u} \lambda_i^{-1}\left(\tilde{\mathbf{G}}^T\,(\mathbf{Y}\,\mathbf{Y}^T)^{-1}\,\tilde{\mathbf{G}}\right) \tag{15}$$

Equation (15) can be used to calculate the minimum loss provided by the optimal combination of a given set of measurements. However, the use of all measurements is often unnecessary and similar losses may be obtained by combining only a few of the available measurements [1, 11, 13]. Then, the combinatorial optimization problem involves finding the set of n among n_y measurements ($n_u \le n \le n_y$) that can provide the minimal loss for specified n. In index notation, the n measurements are selected by minimizing

$$\min_{X_n \subset X_{n_y}} L_2(X_n) = \sum_{i=1}^{n_u} \lambda_i^{-1}\left(\tilde{\mathbf{G}}_{X_n}^T(\mathbf{Y}_{X_n}\mathbf{Y}_{X_n}^T)^{-1}\tilde{\mathbf{G}}_{X_n}\right) \tag{16}$$

where the scalar constant has been omitted as (14).

4 Bidirectional controlled variable selection

As shown in Section 3, the selection of CVs using exact local method can be seen as subset selection problems. In this section, the BAB methods for solving these problems is presented. For simplicity of notation, we define $\mathbf{M}(X_p) \in \mathbb{R}^{p\times p}$ and $\mathbf{N}(X_p) \in \mathbb{R}^{n_u\times n_u}$ as

$$\begin{aligned} \mathbf{M}(X_p) &= \mathbf{R}^{-T}\tilde{\mathbf{G}}_{X_p}\tilde{\mathbf{G}}_{X_p}^T\mathbf{R}^{-1} && (17)\\ \mathbf{N}(X_p) &= \tilde{\mathbf{G}}_{X_p}^T(\mathbf{Y}_{X_p}\mathbf{Y}_{X_p}^T)^{-1}\tilde{\mathbf{G}}_{X_p} && (18) \end{aligned}$$

where $\mathbf{R}$ is the Cholesky factor of $\mathbf{Y}_{X_p}\mathbf{Y}_{X_p}^T$, *i.e.* $\mathbf{R}^T\mathbf{R} = \mathbf{Y}_{X_p}\mathbf{Y}_{X_p}^T$.

4.1 Lower bounds

Individual measurements. L_1 in (14) requires inversion of $\mathbf{G}_{X_{n_u}}$ and thus $L_1(X_p)$ is well-defined only when $\mathbf{G}_{X_p}$ is a square matrix, *i.e.* $p = n_u$. On the other hand, BAB methods require evaluation of loss, when the number of selected measurements differs from n_u. Motivated by this drawback, an alternate representation of L_1 is derived in the following discussion. Since n_u measurements are selected as CVs, L_1 can be represented as

$$\begin{aligned} L_1(X_p) &= \sum_{i=1}^{n_u}\sigma_i^2\left(\tilde{\mathbf{G}}_{X_{n_u}}^{-1}\mathbf{Y}_{X_{n_u}}\right) && (19)\\ &= \sum_{i=1}^{n_u}\lambda_i\left(\tilde{\mathbf{G}}_{X_p}^{-1}\mathbf{Y}_{X_p}\mathbf{Y}_{X_p}^T\tilde{\mathbf{G}}_{X_p}^{-T}\right) && (20)\\ &= \sum_{i=1}^{r}\lambda_i^{-1}(\mathbf{N}(X_p)) = \sum_{i=1}^{r}\lambda_i^{-1}(\mathbf{M}(X_p)) && (21) \end{aligned}$$

where $r = \operatorname{rank}(\tilde{\mathbf{G}}_{X_p})$.

In practice, every measurement has a non-zero implementation error associated with it. Thus, based on (12), $[\mathbf{W}_e]_{ii} \neq 0$ and $\mathbf{Y}$ has full row rank. This implies that the inverse of $\mathbf{Y}_{X_p}\mathbf{Y}_{X_p}^T$ is well defined for all practical problems and the expression for L_1 in (21) holds for any number of measurements. Using the generalized expression for L_1, the downwards and upwards lower bounds required for the application of B^3 algorithm are derived next.

Lemma 1 Let the matrix $\hat{\mathbf{A}}$ be defined as

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & a \end{bmatrix} \qquad (22)$$

where $\mathbf{A} \in \mathbb{R}^{p\times p}$ is a Hermitian matrix, $\mathbf{b} \in \mathbb{R}^{p\times 1}$ and $a \in \mathbb{R}$. Let the eigenvalues of $\mathbf{A}$ and $\hat{\mathbf{A}}$ be arranged in descending order. Then [10, Th. 4.3.8]

$$\lambda_{p+1}(\hat{\mathbf{A}}) \leq \lambda_p(\mathbf{A}) \leq \lambda_p(\hat{\mathbf{A}}) \leq \lambda_{p-1}(\mathbf{A}) \leq \cdots \leq \lambda_1(\mathbf{A}) \leq \lambda_1(\hat{\mathbf{A}}) \tag{23}$$

Proposition 1 (Lower bounds for L_1) Consider a node $\mathcal{S} = (F_f, C_c)$. For L_1 defined in (21),

$$\begin{aligned} L_1(F_f) &\leq \min_{X_{n_u}\supset F_f} L_1(X_{n_u});\ f < n_u \qquad (24)\\ L_1(F_f \cup C_c) &\leq \min_{X_{n_u}\subset (F_f\cup C_c)} L_1(X_{n_u});\ f + c > n_u \qquad (25)\end{aligned}$$

Proof: To prove (24), let $\tilde{\mathbf{G}}_{F_f\cup i} = \begin{bmatrix}\tilde{\mathbf{G}}_{F_f}^T & \tilde{\mathbf{G}}_i^T\end{bmatrix}^T$ and $\mathbf{Y}_{F_f\cup i} = \begin{bmatrix}\mathbf{Y}_{F_f}^T & \mathbf{Y}_i^T\end{bmatrix}^T$. Further, let $\mathbf{R}^T\mathbf{R} = \mathbf{Y}_{F_f}\mathbf{Y}_{F_f}^T$ and $\tilde{\mathbf{R}}^T\tilde{\mathbf{R}} = \mathbf{Y}_{F_f\cup i}\mathbf{Y}_{F_f\cup i}^T$ (Cholesky factorization). Then, it follows that $\mathbf{R}$ and $\mathbf{M}(F_f)$ are principal submatrices of $\tilde{\mathbf{R}}$ and $\mathbf{M}(F_f \cup i)$, respectively, obtained by deleting the last row and column of the corresponding matrices. We note that $r = \text{rank}(\tilde{\mathbf{G}}_{F_f}) = f$. Now, using (23), we have

$$\lambda_i^{-1}(\mathbf{M}(F_f)) \quad\leq\quad \lambda_{i+1}^{-1}(\mathbf{M}(F_f \cup i));\quad i = 1, 2, \cdots, f; f < n_u \tag{26}$$

Then,

$$\begin{aligned} L_1(F_f \cup i) &= \sum_{i=1}^{f+1} \lambda_i^{-1}(\mathbf{M}(F_f \cup i)) \qquad (27)\\ &= \left(\lambda_1^{-1}(\mathbf{M}(F_f \cup i)) + \sum_{i=2}^{f+1} \lambda_i^{-1}(\mathbf{M}(F_f \cup i))\right) \qquad (28)\\ &\geq \left(\lambda_1^{-1}(\mathbf{M}(F_f \cup i)) + \sum_{i=1}^{f} \lambda_i^{-1}(\mathbf{M}(F_f))\right) \qquad (29)\\ &\geq \sum_{i=1}^{f} \lambda_i^{-1}(\mathbf{M}(F_f)) = L_1(F_f) \qquad (30)\end{aligned}$$

Now, (24) follows by recursive use of the above expression.

For (25), it can be similarly shown that $\mathbf{M}((F_f \cup C_c) \setminus i)$ is a principal submatrix of $\mathbf{M}(F_f \cup C_c)$. Based on (23),

$$\lambda_i^{-1}(\mathbf{M}(F_f \cup C_c)) \quad\leq\quad \lambda_i^{-1}(\mathbf{M}((F_f \cup C_c) \setminus i));\quad i = 1, 2, \cdots, n_u \tag{31}$$

We have $\text{rank}(\tilde{\mathbf{G}}_{F_f\cup C_c}) = \text{rank}(\tilde{\mathbf{G}}_{(F_f\cup C_c)\setminus i}) = n_u$. Now,

$$\begin{aligned} L_1((F_f \cup C_c) \setminus i) &= \sum_{i=1}^{n_u} \lambda_i^{-1}(\mathbf{M}((F_f \cup C_c) \setminus i)) \qquad (32)\\ &\geq \sum_{i=1}^{n_u} \lambda_i^{-1}(\mathbf{M}(F_f \cup C_c)) = L_1(F_f \cup C_c) \qquad (33)\end{aligned}$$

Now, (25) can be shown to be true through recursive use of the above expression. ■

To illustrate the implications of Proposition 1, let B represent the best available upper bound on $L_1(X^*_{n_u})$. Then (24) implies that, if $L_1(F_f) > B$, the optimal solution cannot be a superset of F_f and hence all supersets of F_f need not be evaluated. Similarly, if $L_1(F_f \cup C_c) > B$, (25) implies that the optimal solution cannot be a subset of $F_f \cup C_c$ and hence all subsets of $F_f \cup C_c$ need not be evaluated. Thus, upwards and downwards pruning can be conduced using (24) and (25) and the optimal solution can be found without complete enumeration.

Measurements combinations. The expression for L_2 in (16) is the same as the expression for L_1 in (21). Thus, similar to Proposition 1, it can be shown that

$$L_2(F_f \cup C_c) \quad \leq \quad \min_{X_n \subset (F_f \cup C_c)} L_2(X_n); \quad f + c > n \tag{34}$$

For selecting measurements, whose combinations can be used as CVs, the result in (34) is useful for downwards pruning. Equation (25), however, also implies that when $n_u \leq f < n$, $L_2(F_f)$ decreases as the subset size increases. Thus, unlike L_1, the expression for L_2 cannot be directly used for upwards pruning. In the following proposition, a lower bound on L_2 is derived, which can instead be used for upwards pruning, whenever $n - n_u < f < n$.

Proposition 2 (Upwards lower bound for L_2) For the node $\mathcal{S} = (F_f, C_c)$, let

$$\underline{L}_2(F_f) = \sum_{i=1}^{f+n_u-n} \lambda_i^{-1}\left(\mathbf{N}(F_f)\right); \quad f > n - n_u \tag{35}$$

Then,

$$\underline{L}_2(F_f) \quad \leq \quad \min_{\substack{X_n \supset F_f \\ X_n \subset (F_f \cup C_c)}} L_2(X_n) \tag{36}$$

Proof: Consider the index set $X_n \subset (F_f \cup C_c)$. For $j \in X_n$ with $j \notin F_f$, similar to the proof of Proposition 1, $\mathbf{M}(X_n \setminus j)$ is a principal submatrix of $\mathbf{M}(X_n)$. Based on Lemma 1, we have

$$\lambda_i^{-1}(\mathbf{M}(X_n \setminus j)) \quad \leq \quad \lambda_{i+1}^{-1}(\mathbf{M}(X_n)); \quad i = 1, 2 \cdots n_u - 1 \tag{37}$$

Then,

$$\begin{aligned} L_2(X_n) &= \sum_{i=1}^{n_u} \lambda_i^{-1}(\mathbf{M}(X_n)) && (38)\\ &= \left(\lambda_1^{-1}(\mathbf{M}(X_n)) + \sum_{i=2}^{n_u} \lambda_i^{-1}(\mathbf{M}(X_n))\right) && (39)\\ &\geq \left(\lambda_1^{-1}(\mathbf{M}(X_n)) + \sum_{i=1}^{n_u-1} \lambda_i^{-1}(\mathbf{M}(X_n \setminus j))\right) && (40)\\ &\geq \sum_{i=1}^{n_u-1} \lambda_i^{-1}(\mathbf{M}(X_n \setminus j)) && (41) \end{aligned}$$

Through repeated application of (23), for $X_p \in X_n$, $X_p \notin F_f$

$$L_2(X_n) \geq \sum_{i=1}^{n_u-p} \lambda_i^{-1}(\mathbf{M}(X_n \setminus X_p)) \tag{42}$$

Without loss of generality, we can select X_p such that $F_f = X_n \setminus X_p$. Then, $p = n - f$ and

$$L_2(X_n) \geq \sum_{i=1}^{n_u-n+f} \lambda_i^{-1}(\mathbf{M}(F_f)) \tag{43}$$

which implies (36). ■

Proposition 2 implies that $\underline{L}_2(F_f)$ in (35) is a lower bound on the loss corresponding to combinations of any n measurements obtained by appending indices to F_f and hence can be used for upwards pruning. In this case, upwards pruning can only be applied to a node with $f > n - n_u$. Thus, the BAB algorithm based on $\underline{L}_2$ in (35) is referred to as partial bidirectional BAB (PB^3) algorithm. Development of fully bidirectional BAB algorithm for selection of measurement combination as CVs is an open problem.

4.2 Fast pruning and branching

Propositions 1 and 2 can be used to prune the non-optimal nodes quickly. Thus, the optimal solution can be found with evaluation of fewer nodes, but the solution time can still be large, as direct evaluation of L_1 in (21) and $\underline{L}_2$ in (35) is computationally expensive.

Individual measurements. We note that when $f < n_u$, $\mathbf{M}(F_f)$ in (17) is invertible. Similarly for $s = f + c > n_u$, $\mathbf{N}(S_s)$ in (18) is invertible. Thus, based on (21),

$$\begin{aligned} L_1(F_f) &= \sum_{i=1}^{f} \lambda_i(\mathbf{M}^{-1}(F_f)) = \mathrm{trace}(\mathbf{M}^{-1}(F_f)) && (44)\\ L_1(S_s) &= \sum_{i=1}^{n_u} \lambda_i(\mathbf{N}^{-1}(S_s)) = \mathrm{trace}(\mathbf{N}^{-1}(S_s)) && (45) \end{aligned}$$

The use of (44) and (45) for evaluation of lower bounds on L_1 avoids computation of eigenvalues. The next two propositions relate the bounds of a node with the bounds of sub-nodes allowing pruning on sub-nodes directly and thus improving efficiency of the B^3 algorithm further.

Proposition 3 (Upwards pruning for L_1) Consider a node $\mathcal{S} = (F_f, C_c)$ and index $i \in C_c$. Then

$$L_1(F_f \cup i) = L_1(F_f) + \frac{\|\mathbf{z}_i^T \mathbf{Y}_{F_f} - \mathbf{Y}_i\|_2^2}{\eta_i} \tag{46}$$

where $\mathbf{z}_i = (\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_{F_f}^T)^{-1}\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_i^T$ and $\eta_i = \tilde{\mathbf{G}}_i(\mathbf{I} - \mathbf{G}_{F_f}^T(\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_{F_f}^T)^{-1}\tilde{\mathbf{G}}_{F_f})\tilde{\mathbf{G}}_i^T$.

Proof: Based on (44),

$$L_1(F_f \cup i) \;=\; \text{trace}(\mathbf{Q}(\mathbf{G}_{F_f\cup i}\mathbf{G}_{F_f\cup i}^T)^{-1}\mathbf{Q}^T) \tag{47}$$

where $\mathbf{Q}$ is the Cholesky factor of $\mathbf{Y}_{F_f\cup i}\mathbf{Y}_{F_f\cup i}^T$, *i.e.* $\mathbf{Q}^T\mathbf{Q} = \mathbf{Y}_{F_f\cup i}\mathbf{Y}_{F_f\cup i}^T$. Since $\mathbf{G}_{F_f\cup i} = \begin{bmatrix}\tilde{\mathbf{G}}_{F_f}^T & \tilde{\mathbf{G}}_i^T\end{bmatrix}^T$, using the matrix inversion formula for partitioned matrices [10]

$$(\mathbf{G}_{F_f\cup i}\mathbf{G}_{F_f\cup i}^T)^{-1} \;=\; \begin{bmatrix}\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_{F_f}^T & \tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_i^T \\ \tilde{\mathbf{G}}_i\tilde{\mathbf{G}}_{F_f}^T & \tilde{\mathbf{G}}_i\tilde{\mathbf{G}}_i^T\end{bmatrix}^{-1} \tag{48}$$

$$= \begin{bmatrix}(\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_{F_f}^T)^{-1} + \mathbf{z}_i\mathbf{z}_i^T/\eta_i & -\mathbf{z}_i/\eta_i \\ \mathbf{z}_i^T\eta_i & 1/\eta_i\end{bmatrix} \tag{49}$$

$$= \begin{bmatrix}(\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_{F_f}^T)^{-1} & \mathbf{0} \\ \mathbf{0} & 0\end{bmatrix} + 1/\eta_i \begin{bmatrix}\mathbf{z}_i \\ -1\end{bmatrix}\begin{bmatrix}\mathbf{z}_i^T & -1\end{bmatrix} \tag{50}$$

Through simple algebraic manipulations, it can be shown that

$$\mathbf{Q} = \begin{bmatrix}\mathbf{R} & \mathbf{p}_i \\ 0 & \delta_i\end{bmatrix} \tag{51}$$

where $\mathbf{R}$ is the Cholesky factor of $\mathbf{Y}_{F_f}\mathbf{Y}_{F_f}^T$, $\mathbf{p}_i = \mathbf{R}^{-T}\mathbf{Y}_{F_f}\mathbf{Y}_i^T$ and $\delta_i = \sqrt{\mathbf{Y}_i\mathbf{Y}_i^T - \mathbf{p}_i^T\mathbf{p}_i}$. Thus,

$$\mathbf{Q}(\mathbf{G}_{F_f\cup i}\mathbf{G}_{F_f\cup i}^T)^{-1}\mathbf{Q}^T = \begin{bmatrix}\mathbf{R}(\mathbf{G}_{F_f}\mathbf{G}_{F_f}^T)^{-1}\mathbf{R}^T & \mathbf{0} \\ \mathbf{0} & 0\end{bmatrix} + 1/\eta_i \begin{bmatrix}\mathbf{R}\mathbf{z}_i - \mathbf{p}_i \\ -\delta_i\end{bmatrix}\begin{bmatrix}\mathbf{z}_i^T\mathbf{R}^T - \mathbf{p}_i^T & -\delta_i\end{bmatrix} \tag{52}$$

and

$$\text{trace}(\mathbf{Q}(\mathbf{G}_{F_f\cup i}\mathbf{G}_{F_f\cup i}^T)^{-1}\mathbf{Q}^T) = \text{trace}(\mathbf{R}(\mathbf{G}_{F_f}\mathbf{G}_{F_f}^T)^{-1}\mathbf{R}^T) + \frac{\text{trace}((\mathbf{R}\mathbf{z}_i - \mathbf{p}_i)(\mathbf{R}\mathbf{z}_i - \mathbf{p}_i)^T) + \delta_i^2}{\eta_i} \tag{53}$$

Since $\text{trace}((\mathbf{Rz}_i - \mathbf{p}_i)(\mathbf{Rz}_i - \mathbf{p}_i)^T) = (\mathbf{Rz}_i - \mathbf{p}_i)^T(\mathbf{Rz}_i - \mathbf{p}_i)$,

$$\begin{aligned}
\text{trace}((\mathbf{Rz}_i - \mathbf{p}_i)(\mathbf{Rz}_i - \mathbf{p}_i)^T) + \delta_i^2 &= \mathbf{z}_i^T\mathbf{R}^T\mathbf{Rz}_i - \mathbf{z}_i^T\mathbf{R}^T\mathbf{p}_i - \mathbf{p}_i^T\mathbf{Rz}_i + \mathbf{Y}_i\mathbf{Y}_i^T &(54)\\
&= \mathbf{z}_i^T\mathbf{Y}_{F_f}\mathbf{Y}_{F_f}^T\mathbf{z}_i - \mathbf{z}_i^T\mathbf{Y}_{F_f}\mathbf{Y}_i^T - \mathbf{Y}_i\mathbf{Y}_{F_f}\mathbf{z}_i + \mathbf{Y}_i\mathbf{Y}_i^T &(55)\\
&= (\mathbf{z}_i^T\mathbf{Y}_{F_f} - \mathbf{Y}_i)(\mathbf{z}_i^T\mathbf{Y}_{F_f} - \mathbf{Y}_i)^T &(56)
\end{aligned}$$

and the result follows. ■

The main computation load in using (46) is in Cholesky factorization and the inversion of the matrix $\tilde{\mathbf{G}}_{F_f}\tilde{\mathbf{G}}_{F_f}^T$, which need to be calculated only once for all $i \in C_c$. Hence, the calculation is more efficient than direct calculation of L_1 using (44).

Proposition 4 (Downward pruning for L_1) For a node $\mathcal{S} = (F_f, C_c)$, let $S_s = F_f \cup C_c$, where $s = f+c$. For $i \in C_c$,

$$L_1(S_s \setminus i) = L_1(S_s) + \frac{\|\mathbf{x}_i\mathbf{N}^{-1}(S_s)\|_2^2}{\zeta_i - \mathbf{x}_i\mathbf{N}^{-1}(S_s)\mathbf{x}_i^T} \tag{57}$$

where $\mathbf{x}_i = \mathbf{Y}_i\mathbf{Y}_{S_s\setminus i}^T(\mathbf{Y}_{S_s\setminus i}\mathbf{Y}_{S_s\setminus i}^T)^{-1}\mathbf{G}_{S_s\setminus i} - \mathbf{G}_i^T$ and $\zeta_i = \mathbf{Y}_i(\mathbf{I} - \mathbf{Y}_{S_s\setminus i}^T(\mathbf{Y}_{S_s\setminus i}\mathbf{Y}_{S_s\setminus i}^T)^{-1}\mathbf{Y}_{S_s\setminus i})\mathbf{Y}_i^T$.

Proof: For simplicity of notation, define $\mathbf{Q} = \mathbf{Y}_{S_s\setminus i}\mathbf{Y}_{S_s\setminus i}^T$. Then,

$$\begin{aligned}
(\mathbf{Y}_{S_s}\mathbf{Y}_{S_s}^T)^{-1} &= \begin{bmatrix} \mathbf{Q} & \mathbf{Y}_{S_s\setminus i}\mathbf{Y}_i^T \\ \mathbf{Y}_i\mathbf{Y}_{S_s\setminus i}^T & \mathbf{Y}_i\mathbf{Y}_i^T \end{bmatrix}^{-1} &(58)\\
&= \begin{bmatrix} \mathbf{Q}^{-1} + \mathbf{Q}^{-1}\mathbf{Y}_{S_s\setminus i}\mathbf{Y}_i^T\mathbf{Y}_i\mathbf{Y}_{S_s\setminus i}^T\mathbf{Q}^{-1}/\zeta_i & -\mathbf{Q}^{-1}\mathbf{Y}_{S_s\setminus i}\mathbf{Y}_i^T/\zeta_i \\ -\mathbf{Y}_i\mathbf{Y}_{S_s\setminus i}^T\mathbf{Q}^{-1}/\zeta_i & 1/\zeta_i \end{bmatrix} &(59)\\
&= \begin{bmatrix} \mathbf{Q}^{-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + 1/\zeta_i \begin{bmatrix} \mathbf{Q}^{-1}\mathbf{Y}_{S_s\setminus i}\mathbf{Y}_i^T \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{Y}_i\mathbf{Y}_{S_s\setminus i}^T\mathbf{Q}^{-1} & -1 \end{bmatrix} &(60)
\end{aligned}$$

where (59) is obtained using the matrix inversion formula for partitioned matrices [10]. Since $\tilde{\mathbf{G}}_{S_s}^T = \begin{bmatrix} \tilde{\mathbf{G}}_{S_s\setminus i}^T & \tilde{\mathbf{G}}_i^T \end{bmatrix}$, we have

$$\begin{aligned}
\mathbf{N}(S_s) &= \tilde{\mathbf{G}}_{S_s}^T(\mathbf{Y}_{S_s}\mathbf{Y}_{S_s}^T)^{-1}\tilde{\mathbf{G}}_{S_s} = \tilde{\mathbf{G}}_{S_s\setminus i}^T\mathbf{Q}^{-1}\tilde{\mathbf{G}}_{S_s\setminus i} + \mathbf{x}_i\mathbf{x}_i^T/\zeta_i &(61)\\
&= \mathbf{N}(S_s \setminus i) + \mathbf{x}_i\mathbf{x}_i^T/\zeta_i &(62)
\end{aligned}$$

Using matrix inversion lemma [10], we have

$$\mathbf{N}^{-1}(S_s \setminus i) = \mathbf{N}^{-1}(S_s) + \frac{1}{\zeta_i - \mathbf{x}_i^T\mathbf{N}^{-1}(S_s)\mathbf{x}_i}\mathbf{N}^{-1}(S_s)\mathbf{x}_i\mathbf{x}_i^T\mathbf{N}^{-1}(S_s) \tag{63}$$

which implies that,

$$\mathrm{trace}(\mathbf{N}^{-1}(S_s \setminus i)) = \mathrm{trace}(\mathbf{N}^{-1}(S_s)) + \frac{\mathrm{trace}(\mathbf{N}^{-1}(S_s)\mathbf{x}_i\mathbf{x}_i^T\mathbf{N}^{-1}(S_s))}{\zeta_i - \mathbf{x}_i^T\mathbf{N}^{-1}(S_s)\mathbf{x}_i} \tag{64}$$

The result follows by using (45) and noting that $\mathrm{trace}(\mathbf{N}^{-1}(S_s)\mathbf{x}_i\mathbf{x}_i^T\mathbf{N}^{-1}(S_s)) = \mathbf{x}_i^T\mathbf{N}^{-1}(S_s)\mathbf{N}^{-1}(S_s)\mathbf{x}_i = \|\mathbf{x}_i^T\mathbf{N}^{-1}(S_s)\|_2^2$. ■

Using (59), it can be shown that $1/\zeta_i$ is the ith diagonal element of $(\mathbf{Y}_{S_s}\mathbf{Y}_{S_s}^T)^{-1}$ and $\mathbf{x}_i^T/\zeta_i$ is the ith row of the matrix $(\mathbf{Y}_{S_s}\mathbf{Y}_{S_s}^T)^{-1}\tilde{\mathbf{G}}_{S_s}$. Therefore, the use of condition in (57) requires inversion of two matrices, $(\mathbf{Y}_{S_s}\mathbf{Y}_{S_s}^T)$ and $\mathbf{N}(S_s)$, which need to be calculated only once for all $i \in C_c$. Hence, the calculation is more efficient than direct calculation of L_1 using (45).

The bidirectional branching approach mentioned in Section 2.2 requires selecting a decision element, which can be done directly based on the loss calculated for the super-nodes, $L_1(F_f \cup i)$ and for the sub-nodes, $L_1(S_s \setminus i)$. More specifically, according to the "best-first" rule, for upwards-first branching, element i is selected as the decision element if $L_1(F_f \cup i) = \min_{j \in C_c} L_1(F_f \cup j)$ or if $L_1(S_c \setminus i) = \max_{j \in C_c} L_1(S_s \setminus j)$. Similarly, for downwards-first branching, element i is selected as the decision element if $L_1(F_f \cup i) = \max_{j \in C_c} L_1(F_f \cup j)$ or if $L_1(S_c \setminus i) = \min_{j \in C_c} L_1(S_s \setminus j)$. Between these two criteria for upwards and downwards branching, the one with the larger value is less conservative and hence is adopted for the selection of decision element. Overall, no extra calculation is required for fast branching. The flowchart for recursive implementation of the proposed B^3 algorithm is available in [12].

Measurements combinations. As the downwards pruning criteria for minimization of L_1 and L_2 are the same, Proposition 4 can be used for fast downwards pruning for selection of a subset of measurements, whose combinations can be used as CVs. The fast upwards pruning criteria for minimization of L_2 is presented in the next proposition.

Proposition 5 (Upwards pruning for L_2) Consider a node $\mathcal{S} = (F_f, C_c)$ and index $i \in C_c$. Let $q = f + n_u - n + 1$. Then

$$\underline{L}_2(F_f \cup i) \geq \frac{q^2}{\sum_{j=1}^{q} \lambda_j(\mathbf{N}(F_f)) + \|\mathbf{s}_i\|_2^2/\beta_i} \tag{65}$$

where $\mathbf{s}_i = \mathbf{Y}_i\mathbf{Y}_{F_f}^T(\mathbf{Y}_{F_f}\mathbf{Y}_{F_f}^T)^{-1}\mathbf{G}_{F_f} - \mathbf{G}_i^T$ and $\beta_i = \mathbf{Y}_i(\mathbf{I} - \mathbf{Y}_{F_f}^T(\mathbf{Y}_{F_f}\mathbf{Y}_{F_f}^T)^{-1}\mathbf{Y}_{F_f})\mathbf{Y}_i^T$.

Proof: Similar to the proof of Proposition 4, it can be shown that $\mathbf{N}(F_f \cup i) = \mathbf{N}(F_f) + \mathbf{s}_i\mathbf{s}_i^T/\beta_i$. According to [6, Th. 8.1.8], $\lambda_j(\mathbf{N}(F_f \cup i)) = \lambda_j(\mathbf{N}(F_f)) + t_j$; $j = 1, 2, \cdots, n_u$ and $\sum_{j=1}^{n_u} t_j = \|\mathbf{s}_i\|_2^2/\beta_i$; $t_j \geq 0$.

Therefore, a lower bound on $\mathbf{N}(F_f \cup i)$ can be obtained by solving the following minimization problem:

$$\min_{t_1,\ldots,t_{n_u}} \sum_{j=1}^{q} \frac{1}{\lambda_j(\mathbf{N}(F_f)) + t_j} \tag{66}$$

$$\text{s.t.} \sum_{j=1}^{n_u} t_j = \|\mathbf{s}_i\|_2^2/\beta_i;\ t_j \geq 0,\ j = 1,\ldots,n_u \tag{67}$$

Let the Lagrangian function be defined as

$$\mathcal{L} = \sum_{j=1}^{q} \frac{1}{\lambda_j(\mathbf{N}(F_f)) + t_j} + \nu \left(\sum_{j=1}^{n_u} t_j - \|\mathbf{s}_i\|_2^2/\beta_i \right) + \sum_{j=1}^{n_u} \mu_j t_j$$

The optimality conditions for minimizing $\mathcal{L}$ are (see *e.g.* [5]):

$$\frac{\partial \mathcal{L}}{\partial t_j} = \nu - \frac{1}{(\lambda_j(\mathbf{N}(F_f)) + t_j)^2} + \mu_j = 0, \qquad j = 1,\ldots,q \tag{68}$$

$$\frac{\partial \mathcal{L}}{\partial t_k} = \nu + \mu_k = 0, \qquad k = q+1,\ldots,n_u \tag{69}$$

$$\mu_j t_j = 0, \qquad j = 1,\ldots,n_u \tag{70}$$

$$\sum_{j=1}^{n_u} t_j = \|\mathbf{s}_i\|_2^2/\beta_i \tag{71}$$

Since $\nu \neq 0$, we have $\mu_k \neq 0$ and thus $t_k = 0$ for $k = q+1,\ldots,n_u$. Also, since $t_j \neq 0$, we have $\mu_j = 0$ for $j = 1,\ldots,q$. Therefore,

$$\lambda_j(\mathbf{N}(F_f)) + t_j = \frac{1}{\sqrt{\nu}}; \quad j = 1,\ldots,q \tag{72}$$

which leads to the following dual problem

$$\mathcal{D} : \max_{\nu} 2q\sqrt{\nu} - \nu \left(\sum_{j=1}^{q} \lambda_j(\mathbf{N}(F_f)) + \|\mathbf{s}_i\|_2^2/\beta_i \right) \tag{73}$$

The solution of the dual problem is obtained as follows

$$\sqrt{\nu} = \frac{q}{\sum_{j=1}^{q} \lambda_j(\mathbf{N}(F_f)) + \|\mathbf{s}_i\|_2^2/\beta_i} \tag{74}$$

Now, (65) follows by substituting for ν in (73). ∎

The direct computation of $\underline{L}_2(F_f \cup i)$ requires finding the eigenvalues of $\mathbf{N}(F_f \cup i)$ for all $i \in C_c$. In comparison, Proposition 5 only requires computing eigenvalues of $\mathbf{N}(F_f)$ and is thus much faster than direct computation of $\underline{L}_2(F_f \cup i)$. Note that the relationship in (65) is an inequality, which can be conservative. As a BAB method spends most of its time in evaluating nodes that cannot lead to the optimal solution, we use the computationally cheaper albeit weaker pruning criteria in this paper. For the PB^3 algorithm for minimization of L_2, the decision element for fast branching is chosen using a similar approach as taken for minimization of L_1.

5 Numerical Examples

To examine the efficiency of the proposed BAB algorithms, numerical tests are conducted using randomly generated matrices and binary distillation column case study. Programs used for loss minimization are listed in Table 1. All tests are conducted on a Windows XP SP2 notebook with an Intel® Core™ Duo Processor T2500 (2.0 GHz, 2MB L2 Cache, 667 MHz FSB) using MATLAB® R2008a.

Table 1: BAB programs for comparison

program	description
UP	upwards pruning using (46)
DOWN	downwards pruning using (57)
B^3	bidirectional BAB by combining (46) and (57)
PB^3	partially B^3 by combining (57) and (65)

5.1 Random tests

Four sets of random tests are conducted to evaluate the efficiency of different BAB algorithms mentioned in Table 1 for selection of a subset of available measurements as CVs through minimization of the local average loss. For each test, six random matrices are generated: three full matrices, $\mathbf{G}^y \in \mathbb{R}^{n_y \times n_u}$, $\mathbf{G}_d^y \in \mathbb{R}^{n_y \times n_d}$ and $\mathbf{J}_{ud} \in \mathbb{R}^{n_u \times n_d}$, and three diagonal matrices, $\mathbf{W}_e \in \mathbb{R}^{n_y \times n_y}$, $\mathbf{W}_d \in \mathbb{R}^{n_d \times n_d}$ and $\mathbf{J}_{uu} \in \mathbb{R}^{n_u \times n_u}$. All the elements of $\mathbf{G}^y$, $\mathbf{G}_d^y$ and $\mathbf{J}_{ud}$, and the diagonal elements of $\mathbf{W}_e$ and $\mathbf{W}_d$ are uniformly distributed between $0 - 1$. To avoid ill-conditioning, the diagonal elements of $\mathbf{J}_{uu}$ are uniformly distributed between $1 - 10$. For all tests, we use $n_d = 5$, while n_u and n_y are varied. For each selection problem, 100 random cases are tested and the average computation time and number of nodes evaluated over the 100 random cases are summarized in Figure 2 for Tests 1 and 2 and Figure 3 for Tests 3 and 4, respectively.

The first and second tests are designed to select $n_u = 5$ and $n_u = n_y - 5$ out of n_y measurements, respectively. From Figure 2, it can be seen that algorithm UP is more suitable for problems involving selection of a few variables from a large candidate set, whilst algorithm DOWN is more efficient for problems, where a few among many candidate variables need to be discarded to find the optimal solution. The solution times for UP and DOWN algorithms increase only modestly with problem size, when $n_u << n_y$ and $n_u \approx n_y$, respectively. The solution times for the B^3 algorithm is similar to the better of UP and DOWN algorithms, however, its efficiency is insensitive to the kind of selection problem.

The third test consists of selecting n_u out of $n_y = 2n_u$ measurements with n_u increasing from 5 to 18,

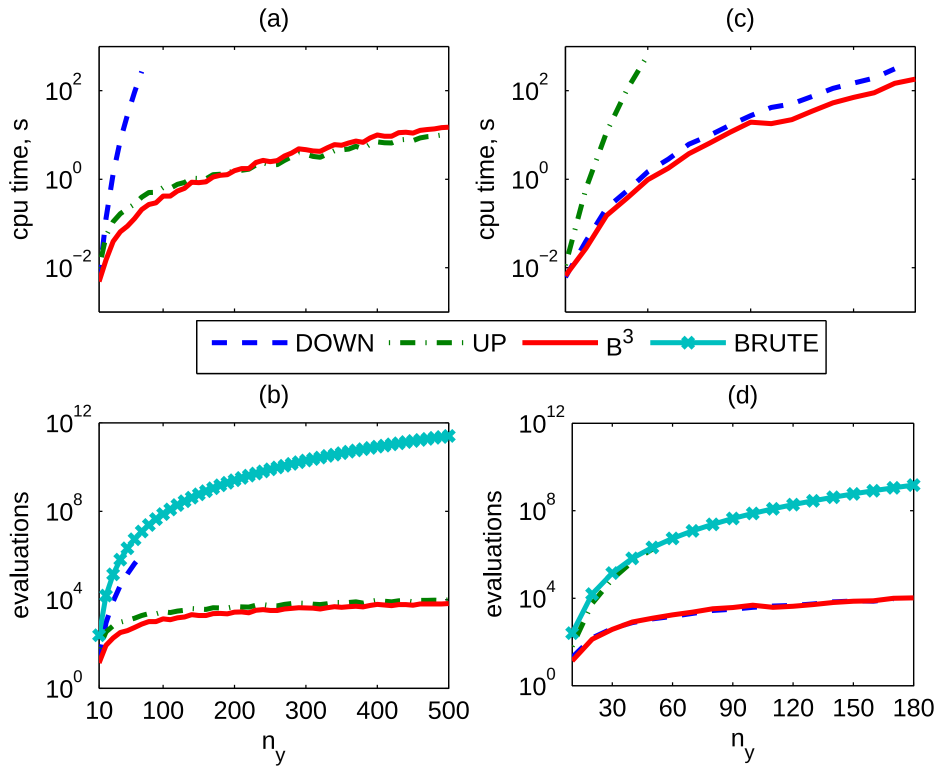


Figure 2: Random test 1: (a) computation time and (b) number of nodes evaluated against n_y; Random test 2: (c) computation time and (d) number of nodes evaluated against n_y.

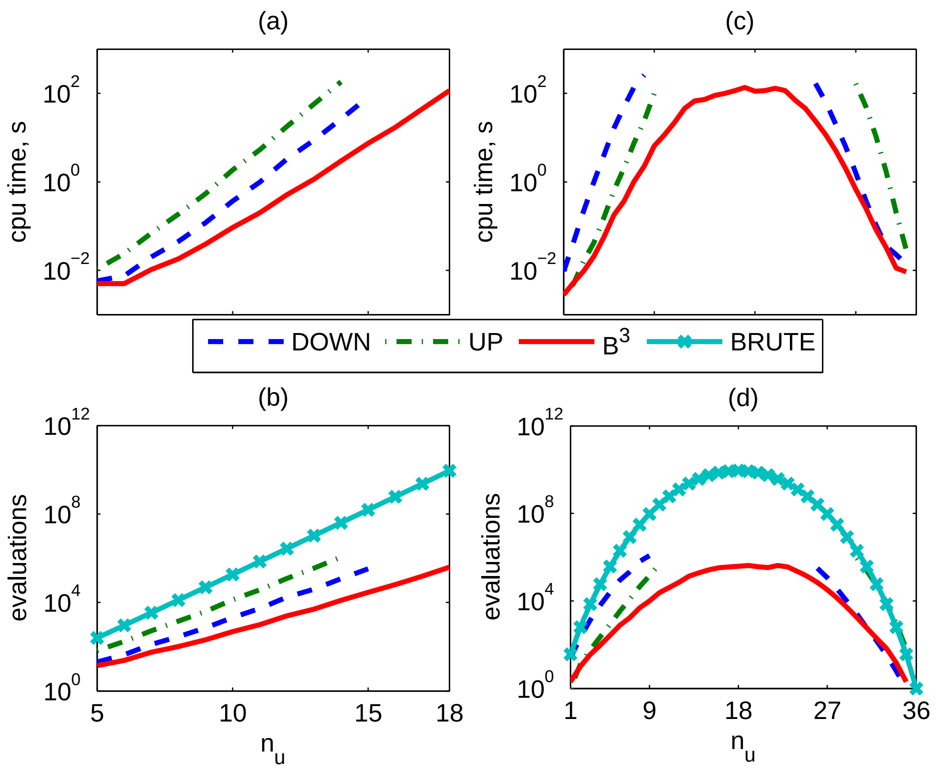


Figure 3: Random test 3: (a) computation time and (b) number of nodes evaluated against n_u; Random test 4: (c) computation time and (d) number of nodes evaluated against n_u.

while the fourth test involves selecting n_u out of $n_y = 36$ variables with n_u ranging from 1 to 35. Figure 3 indicates that while the UP and DOWN algorithms show reasonable performance for small n_u, their performance degrades, when $n_u \approx n_y/2$. Within 1000 seconds, both UP and DOWN algorithms can only handle problems with $n_u < 9$ or $n_y - n_u < 9$. For all cases, however, the B^3 algorithm exhibits superior efficiency and is able to solve problems up to $n_u = 18$ within 200 seconds.

In summary, for selection of individual measurements as CVs by minimizing the average loss, all the developed algorithms (UP, DOWN and B^3) show much superior performance than the currently used brute force method. In comparison with the UP and DOWN algorithms, the B^3 algorithm shows superior performance and similar efficiency for different problem dimensions including problems with $n_u << n_y$, $n_u \approx n_y$ and $n_u \approx n_y/2$.

5.2 Distillation column case study

To demonstrate the efficiency of the developed PB^3 algorithm, we consider self-optimizing control of a binary distillation column [15]. The objective is to minimize the relative steady-state deviations of the distillate (z^L_{top}) and bottoms (z^H_{btm}) compositions from their nominal values, *i.e.*

$$J = \left(\frac{z^L_{\text{top}} - z^L_{\text{top,s}}}{z^L_{\text{top,s}}}\right)^2 + \left(\frac{z^H_{\text{btm}} - z^H_{\text{btm,s}}}{z^L_{\text{btm,s}}}\right)^2 \tag{75}$$

where the superscripts L and H refer to the light and heavy components and the nominal steady-state values are $z^L_{\text{top,s}} = z^H_{\text{btm,s}} = 0.01$ (99% purity). Two manipulated variables, namely reflux (u_1) and vapor boilup rates (u_2), are available for minimizing J in (75). The main disturbances are in feed flow rate (d_1), feed composition (d_2) and vapor fraction of feed (d_3), which can vary between 1 ± 0.2, 0.5 ± 0.1 and 1 ± 0.1, respectively. The top and bottom compositions are not measured online and thus two CVs needs to be identified for indirect control of the compositions. It is considered that the temperatures on 41 trays ($y_1, \cdots, y_{41}$) are measured with an accuracy of $\pm 0.5^o$ C, whose combinations can be used as CVs for implementation of self-optimizing control strategy.

For local analysis, the following linear model is derived:

$$\begin{bmatrix} z^L_{\text{top}} \\ z^H_{\text{btm}} \end{bmatrix} = \begin{bmatrix} 1.083 & 1.097 \\ 0.877 & -0.863 \end{bmatrix} \mathbf{u} + \begin{bmatrix} 0.586 & 1.117 & 1.091 \\ 0.394 & 0.883 & 0.869 \end{bmatrix} \mathbf{d}$$

Using the linear model and (75), the Hessian matrices required for local analysis are calculated to be

$$\begin{aligned} \mathbf{J}_{uu} &= \begin{bmatrix} 38832.119 & -38888.107 \\ -38888.107 & 38951.438 \end{bmatrix} \\ \mathbf{J}_{ud} &= \begin{bmatrix} 19600.452 & 39679.404 & 38863.870 \\ -19652.356 & -39742.991 & -38924.023 \end{bmatrix} \end{aligned}$$

while $\mathbf{W}_d = \text{diag}(0.2, 0.1, 0.1)$ and $\mathbf{W}_e = 0.5\mathbf{I}_{41}$. The reader is referred to [9] for further details of this case study.

The PB^3 algorithm is used to select the 10 best measurement combinations for $2 \leq n \leq 41$. The trade-off between the losses of the 10 best selections and n is shown in Figure 4(a). It can be seen that when $n \geq 14$, the loss is less than 0.075, which is close to the minimum loss (0.052) obtained by using a combination of all 41 measurements. Furthermore, the reduction in loss is negligible, when combinations of more than 20 measurements are used. Figure 4(a) also shows that the 10 best selections have similar self-optimizing

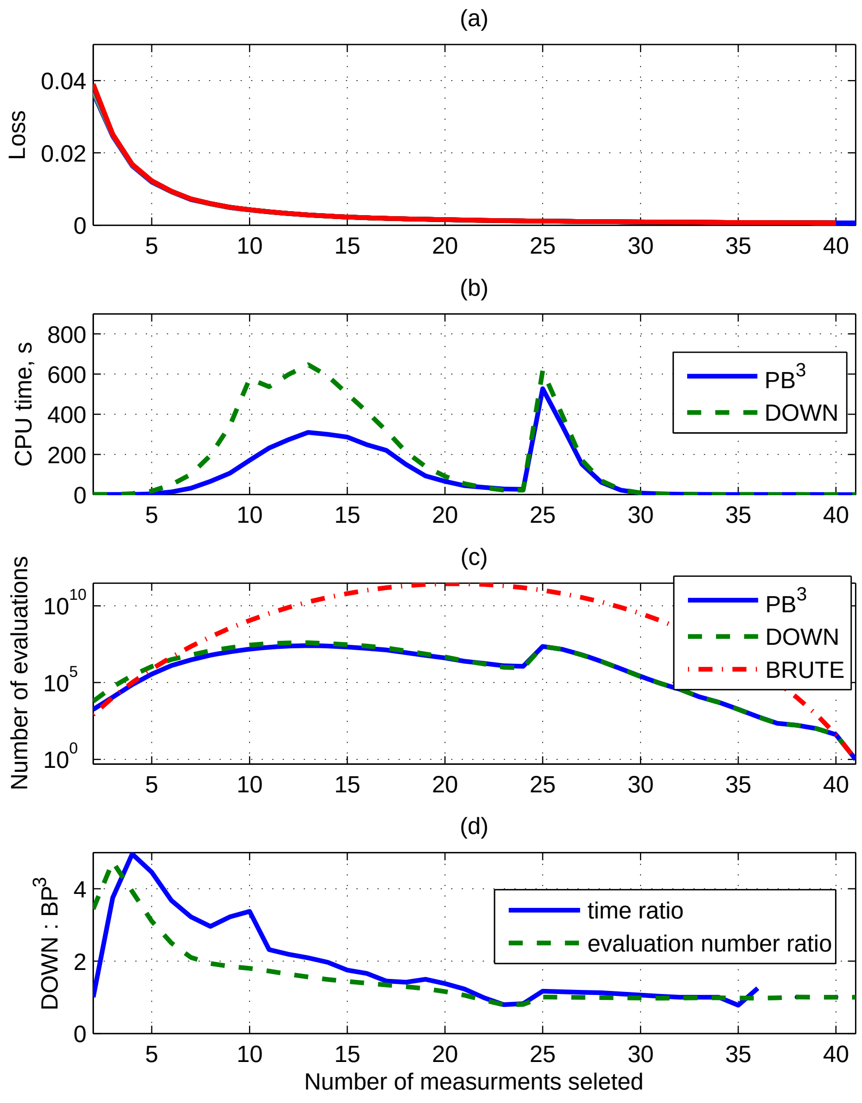


Figure 4: (a) Average losses of 10-best measurement combinations against the number of measurements, (b) Comparison of computation time, (c) Comparison of number of node evaluations, and (d) Ratios of computation time and number of node evaluations required by PB^3 over DOWN algorithms

capabilities. Thus, the designer can choose the subset of measurements among these 10 best alternatives based on some other important criteria, such as dynamic controllability [17].

Figure 4(b) and (c) show the computation time and number of node evaluations for PB^3 and DOWN algorithms. To facilitate the comparison further, the ratios of number of node evaluations and computation times are also shown in Figure 4(d). Due to the conservativeness of the pruning condition (65), the PB^3 algorithm is only able to reduce the number of node evaluations and hence computation time up to a factor of 4 for selection problems involving selection of a few measurements from a large candidate set. It is expected that a less conservative or fully upwards pruning rule would improve the efficiency, but the derivation of such a rule is currently an open problem.

Overall, both algorithms are very efficient and are able to reduce the number of node evaluations by 5 to 6 orders of magnitude, as compared to the brute force search method. For example, to select 20 measurements from 41 candidates, evaluation of a single alternative requires about 0.15 ms on the specified notebook computer. Thus, a brute force search methods would take more than one year to evaluate all possible alternatives. However, both PB^3 and DOWN algorithms are able to solve this problem within 100 seconds. Hence, without the algorithms developed here, it would be practically impossible to generate of the trade-off curve shown in Figure 4(a).

6 Conclusions

Self-optimizing control is a promising method for systematic selection of controlled variables (CVs) from available measurements. In this paper, efficient bidirectional branch and bound (BAB) algorithms have been developed for selection of controlled variables (CVs) using the local average loss minimization criterion for self-optimizing control. The numerical tests using randomly generated matrices and binary distillation column case study show that the number of evaluations required by the proposed algorithms are 4 to 5 orders of magnitude lower than the current practice of CV selection using brute force search. This algorithm would allow the practicing engineer to select CVs for large-dimensional problems in a computationally tractable manner.

The proposed algorithm for selection of subset of measurements, whose combinations can be used as CVs, is only partially bidirectional. For this problem, the development of a fully bidirectional branch and bound algorithm is currently open and is an issue for future research. It is noted in [13] that for CV selection, the problems involving minimization of local worst-case and average losses can be conflicting in nature.

To overcome this difficulty, an extension of the bidirectional BAB algorithm to select CVs based on the bi-objective minimization of local worst-case and average losses for self-optimizing control is currently under consideration.

Acknowledgements

The first author gratefully acknowledges the financial support from Nanyang Technological University, Singapore through grant no. RG42/06.

References

[1] V. Alstad, S. Skogestad, and E. S. Hori. Optimal measurement combinations as controlled variables. *J. Proc. Control*, 19(1):138–148, 2009.

[2] Y. Cao and V. Kariwala. Bidirectional branch and bound for controlled variable selection: Part I. Principles and minimum singular value criterion. *Comput. Chem. Engng.*, 32(10):2306–2319, 2008.

[3] Y. Cao and P. Saha. Improved branch and bound method for control structure screening. *Chem. Engg. Sci.*, 60(6):1555–1564, 2005.

[4] X.-W. Chen. An improved branch and bound algorithm for feature selection. *Pattern Recognition Letters*, 24:1925–1933, 2003.

[5] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization.* Oxford University Press, Newyork, NY, USA, 1995.

[6] G. H. Golub and C. F. van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1993.

[7] I. J. Halvorsen, S. Skogestad, J. C. Morud, and V. Alstad. Optimal selection of controlled variables. *Ind. Eng. Chem. Res.*, 42(14):3273–3284, 2003.

[8] S. Heldt. On a new approach for self-optimizing control structure design. In *Proc. 7th Intl. Symposium on ADCHEM*, Istanbul, Turkey, 2009.

[9] E. S. Hori and S. Skogestad. Selection of controlled variables: Maximum gain rule and combination of measurements. *Ind. Eng. Chem. Res.*, 47(23):9465–9471, 2008.

[10] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1985.

[11] V. Kariwala. Optimal measurement combination for local self-optimizing control. *Ind. Eng. Chem. Res.*, 46(11):3629–3634, 2007.

[12] V. Kariwala and Y. Cao. Bidirectional branch and bound for controlled variable selection: Part II. Exact local method for self-optimizing control. *Comput. Chem. Eng.*, 33(8):1402–1412, 2009.

[13] V. Kariwala, Y. Cao, and S. Janardhanan. Local self-optimizing control with average loss minimization. *Ind. Eng. Chem. Res.*, 47(4):1150–1158, 2008.

[14] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. Comput.*, C-26:917–922, 1977.

[15] S. Skogestad. Dynamics and control of distillation columns - A tutorial introduction. *Trans. IChemE Part A*, 75:539–562, 1997.

[16] S. Skogestad. Plantwide control: The search for the self-optimizing control structure. *J. Proc. Control*, 10(5):487–507, 2000.

[17] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design.* John Wiley & Sons, Chichester, UK, 2nd edition, 2005.

[18] P. Somol, P. Pudil, and J. Kittler. Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 900–912, 2004.

[19] B. Yu and B. Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, 26:883–889, 1993.

List of Figures

1 Solution trees for selecting 2 out of 6 elements. 4

2 Random test 1: (a) computation time and (b) number of nodes evaluated against n_y; Random test 2: (c) computation time and (d) number of nodes evaluated against n_y. 17

3 Random test 3: (a) computation time and (b) number of nodes evaluated against n_u; Random test 4: (c) computation time and (d) number of nodes evaluated against n_u. 18

4 (a) Average losses of 10-best measurement combinations against the number of measurements, (b) Comparison of computation time, (c) Comparison of number of node evaluations, and (d) Ratios of computation time and number of node evaluations required by PB^3 over DOWN algorithms 20

List of Tables

1 BAB programs for comparison 16