Constraints reduction programming by subset selection: a study from numerical aspect

Yuan Shen† [1]
School of Applied Mathematics, Nanjing University of Finance & Economics, Nanjing, 210023, P. R. China.†

Abstract: We consider a novel method entitled constraints reduction programming which aims to reduce the constraints in an optimization model. This method is derived from various applications of management or decision making, and has potential ability to handle a wider range of applications. Due to the high combinatorial complexity of underlying model, it is difficult to obtain a global solution. Instead, we propose three efficient greedy approaches for solving it. Preliminary experimental results show that the proposed approaches can obtain satisfactory results.

Keyword: optimization, constraints reduction, combinatoric complexity, mixed integer programming, big M method, backward selection, greedy method.

MSC(2010): 65K05, 90C30

1 Introduction

In the application of management, decision making, and etc, we may need to do decision by maximizing/minimizing some specific objective subject to some constraints. Mathematically, we aim to minimize the following optimization model:

$$\begin{aligned} &\min_x f(x) && (1.1)\\ \text{s.t.}\quad & Ax \le b \\ & x \ge 0, && (1.2)\end{aligned}$$

where $f(x) : R^n \to R$ is a closed convex function, $A \in R^{m\times n}$, $b \in R^{m\times 1}$, and $x \in R^{n\times 1}$.

In some specific applications, the objective $f(x)$ could stand for a merit function such as the total profit/revenue of a company (for maximization), the management cost of the government(for minimization), and etc. With A_i denoting the i-th row of matrix A, each constraint $A_i x \le b_i$ might represent a policy requirement we have to meet. Generally, we need to optimize the merit function $f(x)$ subject to these policy requirement constraints. However, too many policy requirement constraints can bring burden to the management, i.e., result in an increase of management cost. In such a scenario, the optimal value of model (1.1) is unsatisfactory, a decision maker might allow a number of constraints, especially for those constraints of less importance, to be violated in order to improve the optimal value, although this could result in a decrease of the service/management quality. For the above reason, we need to reduce the number of policy requirement constraints and develop novel approaches for this purpose. This includes two major tasks: (1) how many constraints should be retained; (2) which constraints should be retained. The first task can be seen as a choice of parameter, and can be tuned manually, or obtained by heuristic method. The second question is the one we are interested in and would like to solve.

Mathematically, by introducing some indicator variables, the above problem can be characterized as the following optimization:

$$\begin{aligned} \min_{x,y}\quad & f(x) \\ \text{s.t.}\quad & [y_i = 1] \Leftrightarrow [A_i x \le b_i], \qquad i = 1, ..., m && (1.3)\\ & \|y\|_0 = p, && (1.4)\\ & x \ge 0, \qquad y \in \{0,1\}^m, \end{aligned}$$

[1] Email: ocsiban@126.com. Research supported by National Natural Science Foundation of China grants 11401295 and by Provincial Natural Science Foundation of Jiangsu grants BK20141007 and by Major Program of the National Social Science Foundation of China under Grant 12&ZD114 and by National Social Science Foundation of China under Grant 15BGL58 and by Social Science Foundation of Jiangsu Province under Grant 14EUA001 and by Qinglan Project of Jiangsu Province.

where p denotes the number of constraints we would like to retain. The above model belongs to mixed-integer nonlinear programming models or mixed-integer linear programming models. Moreover, m pairs of logical constraints are given in (1.3), each one involving a binary/boolean indicator variable y_i which indicates the presence of constraint $A_ix \leq b_i$, i.e., a constraint can be “switched on or off by flipping the value of the associated indicator variable. The constraint $\|y\|_0 = p$ together with $y \in \{0,1\}^m$ can guarantee exactly p nonzero entries in y with underlying goal being essentially that of subset selection, hence exactly p constraints are retained. The model (1.3) aims to reduce the number of constraints in (1.1), hence it is referred as “constraints reduction programming” (CRP) in the present paper.

Problems with indicator constraints and logical implications as in (1.3) can be found in literatures, e.g., in sparse principal component analysis [4], in feature selection [22], and in general mixed integer programming (MIP) [5]. Devising efficient and computationally effective methods to deal with logical implications is one of the most fundamental needs to enhance the solvers capability of facing real-world optimization problems [15].

In order to formulate (1.3) as a standard mathematical programming problem one needs to remove the logical implications and write “explicit constraints. The most straightforward way of doing that is by using the so-called “big M” formulation, e.g., see [5]. In which, one can replace the constraint $A_ix \leq b_i$ with $A_ix - b_i \leq M_i(1 - y_i)$ where the constraints are activated or deactivated by multiplying the indicator variable by a very large precomputed constant M_i. If $y_i = 1$, then constraint $A_ix \leq b_i$ is imposed. Conversely, if $y_i = 0$, then constraint $A_ix \leq b_i$ is satisfied by any value of x provided the M_i is sufficiently large, i.e., the constraint is disabled. The resulting model is as follows:

$$\begin{aligned}
\min_{x,y} \quad & f(x) \\
\text{s.t.} \quad & A_ix - b_i \leq M_i(1 - y_i), \qquad i = 1, ..., m && (1.5)\\
& e^T y = p, && (1.6)\\
& x \geq 0, \qquad\qquad y \in \{0,1\}^m,
\end{aligned}$$

where e is an all-one vector. This is a standard MIP model which can be solved by any existing solvers for handling general MIP such as YALMIP, CPLEX, and etc. However, it is necessary to develop dedicated algorithm for solving it for the following reasons. First, the value of M_i might not be easily (or not even at all) computable. Next, we may also encounter numerical difficulty with large value of M_i. In addition, standard solver packages are usually lack of the ability to handle the specific features of CRP model.

Alternatively, one can use other formulations beyond (1.5) such as disjunctive programming [3] or mathematical programming with vanishing constraints (MPVC) [1] to remove the logical implications. In MPVC model, the constraint $A_ix \leq b_i$ is replaced by $y_i(A_ix - b_i) \leq 0$. It is easy to verify that the constraint $A_ix \leq b_i$ is present with $y_i = 1$, while it becomes vanishing/disabled when y_i takes 0, hence y_i plays the same role as in (1.3). We do not need to calculate M_i, and this model does not suffer from numerical difficulty with large value of M_i. However, the MPVC model (1.3) can be difficult to handle. In fact, the constraint $y_i(A_ix - b_i) \leq 0$ is not only nonconvex, but also violates standard constraint qualifications such as MFCQ, ACQ, and etc., see [1], which gives rise to serious difficulties in theoretical and numerical treatment of these problems. Furthermore, even when a convex approximation of MPVC model can be established, its solution can be far away from that of (1.3).

Let us review the CRP problem from an alternative angle. In fact, the constraints $Ax \leq b$ can be treated as a set with m elements. Each constraint in the form of $A_ix \leq b_i$ represents an element in this set. Since the row index i can uniquely represent the constraint $A_ix \leq b_i$, we may not distinguish between the constraint $A_ix \leq b_i$ and the corresponding row index i in the rest part of this paper, i.e., the aforementioned set can be either an indices set or a constraints set. In this manner, the CRP problem (1.3) is equivalent to the reformulated model as follows:

$$\begin{aligned}
\min_{x,\Omega} \quad & f(x) && (1.7)\\
\text{s.t.} \quad & A_ix \leq b_i, \qquad i \in \Omega, \\
& |\Omega| = p, && (1.8)\\
& x \geq 0,
\end{aligned}$$

where Ω is an indices set and $|\cdot|$ is the cardinality/size of the indices set (i.e., the number of indices). For simplicity, a subset with size equal to p is called a p-subset in this paper.

In order to evaluate the quality of a subset, we introduce the definition of "support optimality" [4] which is given in the following form:

Definition 1.1 *With a given index set Ω, the x is called a support optimal (SO) point of (1.7) if and only if it is an optimal solution of optimization as follows:*

$$\begin{aligned} \min_{x} \quad & f(x) \\ s.t. \quad & A_i x \leq b_i, \qquad i \in \Omega, \\ & x \geq 0, \qquad . \end{aligned} \tag{1.9}$$

The SO point is denoted by x^*_Ω.

To make the definition meaningful, the SO point of (1.7) with any p-subset Ω is always assumed to exist. The objective value of SO point (i.e., optimal value of (1.9)) can be seen as a real-valued function defined over subsets of $\{1, 2, ..., m\}$ and denoted as $f^*(\Omega)$. The function $f^*(\Omega)$ is referred as a submodular function in literatures, and it is nondecreasing w.r.t. Ω. i.e., it holds that

$$\hat{\Omega} \subset \tilde{\Omega} \Rightarrow f^*(\hat{\Omega}) \leq f^*(\tilde{\Omega}) \quad \text{for all} \quad \hat{\Omega}, \tilde{\Omega} \in \{1, 2, ..., m\}.$$

Then the subsets are evaluated by the following rule:

Proposition 1.1 *$\hat{\Omega}$ is strictly "better" than $\tilde{\Omega}$ iff $f^*(\hat{\Omega}) < f^*(\tilde{\Omega})$, $\hat{\Omega}$ is "not worse" than $\tilde{\Omega}$ iff $f^*(\hat{\Omega}) \leq f^*(\tilde{\Omega})$. A p-subset $\hat{\Omega}$ is said to be global optimal over all p-subsets when $f^*(\hat{\Omega}) \leq f^*(\Omega)$ holds for any p-subset Ω.*

With the aforementioned definitions, the CRP problem (1.7) can be characterized as seeking for a global optimizer of function $f^*(\Omega)$ over all p-subsets Ω, i.e., solving the following problem:

$$\begin{aligned} \min_{\Omega} \quad & f^*(\Omega) \\ \text{s.t.} \quad & |\Omega| = p. \end{aligned} \tag{1.10}$$

A brutal approach to achieve its global solution is solving as many as C^p_m problems in the form of (1.9) and finding out the best p-subset based on comparison of their optimal values. This procedure is impractical due to its inherently high combinatorial complexity.

Alternatively, the model (1.7) can be regarded as a special type of subset selection problem which aims to select a subset of size equal to p from a given set of m variables for optimizing some objective. Concretely, a typical subset selection problem can be formulated as the following model [17]:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X} \\ & \|x\|_0 = p, \end{aligned} \tag{1.11}$$

where vector x has at most p nonzero elements, and $\mathcal{X}$ denotes a closed convex set.

Remark: 1. With major difficulty lying in the high combinatorial complexity of selecting p positions (for nonzero entries) from all possible positions, models (1.7) and (1.11) are quite similar.

2. The CRP problem (1.7) and model (1.11) share the common point that both of them can be separated into two objectives, one optimizes the criterion, i.e., $\min_x f(x)$, meanwhile the other keeps the subset size as large/small as p. The two objectives are usually conflicting since a subset with a better criterion value could yield a smaller/larger subset size for (1.11)/(1.7).

The origin of subset selection problem might stem from some certain applications where reducing the number of variables has a significantly practical value. In the field of statistical learning [17], we may need to obtain a linear model using a small subset of variables, to minimize the mean square prediction error. The underlying question is how to choose this subset. Early works on this problem include Lasso for recovering a sparse vector from a few observations via random measuring matrix [23], learning algorithms with sparse regularization (sparse learning), column subset selection problem [14] for selecting a few columns from a matrix that captures as much of the matrix as possible. Since then, subset selection has been significantly extended and numerous applications have emerged, e.g., feature selection, sparse learning and dictionary selection in machine learning, sparse approximation and compressed sensing in signal processing, sparse portfolio selection, etc. In machine learning, the variables could be features or observable attributes of a phenomenon, and we would like to predict the phenomenon using only a small subset from the high-dimensional feature space [2]. In signal processing, the variables could correspond to a collection of dictionary vectors, and the goal is to represent the output in a compact manner. In applications of gene expression patterns, it is desirable that utilizing only a small subset of the genes to classify different tissues according to their gene expression, thus encouraging sparse solutions [18]. Some finance applications will prefer sparse portfolio selections in order to reduce transaction costs [8].

There have been abundant works on the subset selection problem (1.11). In general, the subset selection problem is NP-hard [9, 19]. It can be globally solved only for small-scale problems by performing exhaustive or a branch-and-bound search over all possible support sets, rendering these approaches as non-scalable. For solving larger-scale problems, it is important to develop polynomial-time approximation algorithms. The algorithms can be categorized into two classes: greedy algorithms and optimization methods. The greedy algorithms iteratively add or remove variables such that some performance indices can be refined [12, 25, 26]. Some most widely used greedy algorithms include orthogonal Matching Pursuit (OMP) [21], Sparse regression [17], and etc. These algorithms have been widely used in practice due to their simple algorithmic framework and satisfactory computing speed, however, their performance is limited for their greedy nature. Convex relaxation methods relax the original problem by replacing the cardinality constraint such as the L0-norm constraint with convex constraints such as the L1-norm constraint [23]. However, the optimal solutions of the relaxed problem could be distant to that of the original problem.

As mentioned above, albeit the subset selection problem (1.11) has been extensively studied in the context of variable set from the perspectives of both theoretical property and algorithm, however, to the best of present authors' knowledge, most previous works focus on reducing the variables, as a special type of subset selection problem in the context of constraints set, the CRP problem (1.7) has never been touched and studied, hence this is a novel problem. As an emerging methodology for policy reduction as well as other potential applications, hence the CRP problem (1.7) deserves deep investigations.

Although the CRP problem (1.7) is relatively novel and has never been studied, there have been related works in literatures. In chance-constrained programming (CCP) [6], due to the presence of variables of uncertainty and practical reasons, the decision maker allows some constraints to be satisfied with some pre-specified probability, that is, violation may occur for some realizations that as a whole have small probability. In [24], the authors considers solving a standard linear programming by primal-dual interior point method. Its reduces the per-iteration cost by replacing the normal equation matrix with a "reduced" version nearly active (or most violated) dual constraints at the current iterate. It reduces the constraints during iterating progress for saving computing time, but its iterating sequence converges to the solution of the original problem eventually, and does not aim to reduce the constraints.

Although the constraints reduction is a novel flexible modeling tool to incorporate subset selection into optimization problems, the resulting CRP problem (1.7) is generally hard to solve due to its inherently high combinatorial complexity, theories and algorithms have not been available until present time. In this paper, we put emphasize on developing efficient algorithms for solving it, and propose several heuristic methods. Preliminary numerical experiments are carried out to investigate their efficiency.

The rest part of this paper is organized as follows: we introduce the proposed algorithms in Section 2, in Section 3, numerical results are presented, and the concluding remarks are discussed in Section 4.

2 New methods

Before presenting our methods, we briefly summarize the greedy methods for solving sparse regression [17] because they might be relevant to our methods. Sparse regression is a special case of (1.11) where the objective function is in the form of $\|Ax-b\|^2$. There are two common greedy approaches for sparse regression: matching pursuit methods (MP) [16] and forward/backward selection methods. The aforementioned greedy approaches are able to recover a solution with a small number of variables which can "best" represent the measurements.

In MP, it iteratively adds an index i into an indices set such that A_i has the smallest angle with the present residual r^k (initial residual $r^0 = b$), i.e., find $i = \arg\min_i \arccos(A_i, r^k)$ then update residual by $r^{k+1} = r^k - \alpha^k A_i$ where α^k is chosen such that $(r^{k+1}, A_i) = 0$. Due to its simple algorithmic framework, the MP had attracted researchers' attention, and a bunch of works had been done since then. An important improvement of MP is orthogonal matching pursuit (OMP)[21]. It also maintains an indices set, says Ω, and iteratively adds an index i into Ω such that A_i has the smallest angle with the present residual r^k, but a major difference with MP lies in the definition of residual. In OMP, the residual r is updated to be the distance between measurement b and its optimal representation over the subspace spanned by A_i where $i \in \Omega$. Other variations of MP include stagewise OMP, CoSaMP, etc., see [10, 20]. Both these approaches have been studied extensively in the literatures.

Another category of greedy approaches for general subset selection is forward/backward selection methods. In forward selection [27], one adds variables to the model one at a time. At each step, each variable that is not already in the model is tested for inclusion in the model. The most significant variable is added to the model, such that some parameter like P-value (significance level) is below some pre-set level. We begin with an empty indices set including the variable that is most "significant" at the beginning, and continue adding variables until none of remaining variables are significant when added to the model. Because of the complexity that arises from the complex nature of this procedure, it is essentially impossible to control error rates and this procedure should be viewed as exploratory. For sparse regression, Das and Kempe [7] proved that the forward selection can produce a solution with currently best known approximation guarantee.

A major drawback of forward selection is that each addition of a new variable may render one or more of the already included variables non-significant. An alternate approach which avoids this phenomenon is backward selection. Under this approach, one starts with fitting a model with all the variables of interest. Then the least significant variable is dropped, so long as it is not significant at our chosen critical level. We continue by successively refitting reduced models and applying the same rule until all remaining variables are statistically significant. Compared with forward selection, this approach starts from the "other end", and we wind up with the same model. Backward selection has it's own drawbacks as well. Sometimes variables are dropped that would be significant when added to the final reduced models. This suggests that some compromise between forward and backward selection methods should be considered.

Stepwise selection is a greedy selection method that allows moves in either direction, dropping or adding variables at the various steps. Backward stepwise selection involves starting off in a backward approach and then potentially adding back variables if they later appear to be significant. The process is one of alternation between choosing the least significant variable to drop and then re-considering all dropped variables (except the most recently dropped) for re-introduction into the model. Forward stepwise selection is also a possibility, though not as common. In the forward approach, variables once entered may be dropped if they are no longer significant as other variables are added.

Another greedy selection method is the least angle regression (LARS) [11] which is also known as forward stagewise selection. It is a cautious version of forward selection, which may take smaller step size along the direction that forward selection takes. This method allows any searching direction at every step. Compared with forward/backward selection, LARS may perform more stably in some situations, and this is true when two variables are almost equally correlated with the response.

We introduce the definition of active subset:

Definition 2.1 *With respect to a given Ω, we define the active subset as the subset containing the active constraints satisfying $A_i x^*_\Omega = b_i$, and denote it by Ω^a.*

We build approximate solutions to the CRP problem (1.7) recursively based on the backward selection method. Starting with full indices set $\Omega = \{1, ..., m\}$, at each iteration, we solve a subproblem in the form of (1.9) with Ω, then seek to remove one index $\bar{j}$ from the set Ω to produce the largest decrease in the performance measure by scanning each of the remaining indices in Ω^a, i.e., solve the following model

$$\bar{j} = \arg\min_j \{ f^*_{\Omega \setminus j} \mid j \in \Omega^a \}, \tag{2.12}$$

which amounts to solving $|\Omega^a|$ problems in the form of (1.9). We then proceed to the next subproblem with a "decreased" subset $\Omega = \Omega \setminus \bar{j}$ and the recursion is repeated until $|\Omega| = p$. The size of Ω is not less than p during the iterating progress and the performance measures $f^*(\Omega)$ are always well defined according to our assumptions. The resulting algorithm is as follows:

```
Algorithm 1 (Backward selection):
Initialize with $\Omega = \{1, ..., m\}$;
While $|\Omega| > p$ and $\nabla f(x) \neq 0$
  Do Solve (1.9), find $\bar{j}$ by solving (2.12), remove $\bar{j}$ from $\Omega$ by $\Omega = \Omega \setminus \bar{j}$.
End
```

Remark:

1. The idea of Algorithm 1 is iteratively choosing an index to discard for achieving the best improvement of $f^*(\Omega)$ in a greedy manner. In fact, this idea has been applied to solve general subset selection problems with variable set in many related works, e.g., in [8, 13].

2. In general, we can not use forward selection to solve the CRP problem without imposing additional assumptions; otherwise, the algorithm may stuck at the beginning stage. We take a toy example as follows: the forward selection scheme is initialized with $\Omega = \varnothing$, at the first iteration, we would like to select an index to be included into Ω, however, the performance measure $f^*(\Omega)$ can be incomparable providing $f^*(\{i\}) = -\infty$ for some indices belonging to $\{1, ..., m\}$.

3. By assuming that x^*_Ω is non-degenerated, the sequence $\{f^*(\Omega_k)\}$ is strictly decreasing, i.e., $f^*(\Omega_{k+1}) < f^*(\Omega_k)$ for $k = 0, 1, ...$

4. In practical computation, the "activeness" of a constraint $A_i x \leq b_i$ with given x should be relaxed to $|A_i x - b_i| \leq \epsilon$ with a small tolerance $\epsilon > 0$. The numerical performance of the proposed algorithms is empirically insensitive to the setting of ϵ.

5. At each iteration of Algorithm 1, we solve (2.12) which consists of as many as $|\Omega^a|$ problems in the form of (1.9). Under non-degeneration assumption, we have $|\Omega^a| \leq \min(m, n)$, hence Algorithm 1 is a polynomial-time method when the solvers for (1.9) are polynomial-time ones such as interior point method.

6. Algorithm 1 is a one-way method, i.e., once a constraint is dropped, it will not be re-introduced into Ω.

The necessity to solve a large amount of optimizations in the form of (1.9) at each step renders Algorithm 1 computationally inefficient. To improve the time efficiency of Algorithm 1, the index to remove at each iteration can be determined by heuristic approach with much cheaper cost. For instance, we suggest to choose an index such that the normal direction of the corresponding constraint has the largest angle with the gradient direction at present iteration, i.e., calculate $\bar{j}$ by

$$\bar{j} = \arg\min_j \{ \frac{\langle A_j, \nabla f(x_\Omega) \rangle}{\|A_j\| \cdot \|\nabla f(x_\Omega)\|} \mid j \in \Omega^a \}, \tag{2.13}$$

and then remove $\bar{j}$ from Ω. The resulting algorithm is as follows:

```
Algorithm 2 (Simplified backward selection):
Initialize with $\Omega = \{1, ..., m\}$;
While $|\Omega| > p$ and $\nabla f(x) \neq 0$
  Do Solve (1.9), find $\bar{j}$ which solves (2.13), remove $\bar{j}$ from $\Omega$ by $\Omega = \Omega \setminus \bar{j}$.
End
```

Remark:

1. Both Algorithm 1 and 2 take $m - p$ loops until terminated, but Algorithm 2 costs much less at each iteration. In fact, subproblem (2.13) only involves the computation of inner products with negligible cost, the major computational cost at each iteration of Algorithm 2 lies in that of solving (1.9). In contrary, we need to solve as many as $|\Omega^a|$ problems in the form of (1.9) at each iteration of Algorithm 1.

2. Although Algorithm 2 usually takes much less computing time than Algorithm 1 according to numerical experimental results, it does not necessarily obtain the best improvement of $f^*(\Omega)$ at each iteration.

To further improve the time efficiency of Algorithm 2, a simple and straightforward idea is to discard a batch of constraints at one time, e.g., delete all active constraints. The yielded method is as follows:

```
**Algorithm 3 (Batch backward selection):**
Initialize with $\Omega = \{1, ..., m\}$;
While $|\Omega| > p$ and $\nabla f(x) \neq 0$
  Do Solve (1.9) with present $\Omega$, then determine the active subset $\Omega^a$.
  If $|\Omega \setminus \Omega^a| >= p$,
    Do Remove $\Omega^a$ from $\Omega$ by $\Omega = \Omega \setminus \Omega^a$;
    Else do Find $\bar{j} \in \Omega^a$ by solving (2.12). Remove $\bar{j}$ from $\Omega$ by $\Omega = \Omega \setminus \bar{j}$.
  End
End
```

Remark:

1. The per-iteration cost of Algorithm 3 is similar to that of Algorithm 2, but it takes much less iterations than that of Algorithm 1 and 2 in average since a batch of constraints are dropped at each iteration of Algorithm 3.

2. The active subset Ω^a removed from Ω at each step is not necessarily the best choice among all possible subsets with the same size. In fact, even the best subset were removed from Ω at each iteration, that does not imply we can achieve the global solution eventually.

3. Like Algorithm 1 and 2, Algorithm 3 is a one-way method.

4. We may encounter the situation of "overdeleting" at some iteration, which means the size of Ω may be smaller than p after the update of Ω. In this case, we adopt the subproblem (2.13) in Algorithm 2 to determine one index instead of a batch of indices to remove at this iteration.

5. At each iteration, Algorithm 1 always achieves the best improvement of objective value by solving the subproblem (2.12), i.e., Algorithm 1 is the greediest one among all proposed algorithms. Hence Algorithm 1 is expected to obtain better optimal value than that obtained by Alg 2 and Alg 3 although it can take much longer computation time.

3 Numerical experiments

To investigate the numerical performances of our proposed methods, in this section, we will do some numerical experiments and report the experimental results on the proposed algorithms under various problem settings. The testing platform was a desktop computer with an Intel Core i7 4790 at 3.6 GHz and with 8 GB memory. All proposed algorithms were coded by MATLAB and executed in a MATLAB R2015a simulation platform.

We first elaborate how the problem will be constructed, how the experiments and comparisons will be done.We need to specify the objective $f(x)$, the constraints matrix A, and the right-hand side b. The objective function $f(x)$ can be of any type, for simplicity, we only consider the linear function in the form of $f(x) = -c^T x$. We generate c, A and b by `c = rand(n,1)`, `A = rand(m,n)` and `b=10*rand(m,1)`, respectively. The reason for using `rand` instead of `randn` is to guarantee the existence of optimal solution of the original problem, otherwise the feasible region could be empty, the optimal value might be unbounded; however, we will also do some tests with `randn` as supplement. To simplify the comparison of inner products in Algorithm 2 and 3, before experiments, the vector c and all rows of A are normalized by `c = c/norm(c)` and `A = bsxfun(@rdivide,A,sqrt(diag(A*A')))`, respectively.

The test algorithms were carried out by the following procedure. The indices set Ω was always initialized with full indices set by letting `Omega=1:m`. At each iteration, we need to solve a problem in the form of (1.9) which is a standard linear programming. The optimization software package CVX [2] was used to solve it. After solving (1.9), we obtain the optimal value f^*_Ω, the primal solution x^*_Ω (i.e., the SO point), and the dual solution y^*_Ω. The active subset Ω^*_a should include the indices whose corresponding entries in y^*_Ω are positive, i.e., `Omegaa=Omega(y>0)`. For numerical reason, this is usually replaced by `Omegaa=Omega(y>eps)` where `eps` represents a relaxation parameter with default setting `eps=1e-6`. The performances of our algorithms are empirically insensitive to the setting of ϵ. At each iteration of Algorithm 2, we need to solve (2.13) which is a comparison of inner products. Since vector c and all rows of A have been normalized, this procedure can be done by `[~,del] = max(A(Omegaa,:)*c)` where `del` represents the index to remove. For simplicity, the proposed algorithms will be abbreviated as Alg 1, Alg 2 and Alg 3, respectively. As the model under consideration is relatively novel, besides our proposed algorithms, there might be no dedicated algorithm for

[2] http://www.stanford.edu/~boyd/cvx

solving CRP problem to the best of present authors' knowledge. We use the Matlab built-in MILP solver `intlinprog` to solve the "big M" model (1.5), and let it be a competing algorithm entitled "ILP". The default setting of M_i is `1e4` for all $i = 1, ..., m$.

The performances of the proposed algorithms are evaluated from the aspects of both objective value and computation time. The negative objective value $-f(x)$ serves as a performance measure for maximization. However, we may simply call it objective value, and hope this will not lead to any ambiguity. The computation time (in seconds) was used as a speed performance measure. All the experimental results are the average results over 10 runs on random instances.

To have a quick experience of the performances of the proposed algorithms, we present the test results with several different problem settings in Table 3.

Table 3.1 Numerical results on random problems

(m,n,p)	ILP		Alg 1		Alg 2		Alg 3	
	Obj	Time	Obj	Time	Obj	Time	Obj	Time
(50, 5, 10)	26.976	0.349	8.031	15.941	17.337	6.571	18.677	2.902
(50, 5, 20)	11.561	1.182	5.590	11.885	9.352	4.995	9.750	2.113
(50, 50,10)	20.048	1.530	4.331	13.447	7.620	3.669	8.094	0.964
(50, 50,20)	9.058	19.625	4.142	8.930	6.020	2.767	6.279	0.867
(100,10,20)	27.890	33.494	8.912	38.473	16.274	13.607	16.819	4.107
(100,10,30)	16.159	231.550	6.059	33.263	11.091	11.746	12.303	3.532
(100,100,20)	18.263	887.646	3.886	34.945	6.605	7.886	8.055	1.223
(100,100,30)	22.425	4870.059	5.071	45.331	12.893	14.288	14.334	1.989

We see from Table 3 that ILP can obtain the best objective value among all test algorithms, while our algorithm can only obtain a result whose objective value is 35%-75% of that obtained by ILP in average. However, the ILP utilizes the branch and bound method whose computation complexity grows exponentially with the dimension of problem, i.e., the time efficiency of ILP is not dimensionally scalable, thus it can not handle large-scale problems. In contrary, the computation times of our algorithms grow slowly as the dimension increases, hence our algorithms are more suitable for handling large-scale problems. For instances, the ILP is around 8 times faster than Alg 3 when $(m, n, p) = (50, 5, 10)$ while Alg 3 is round 2400 times faster than ILP when $(m, n, p) = (100, 100, 30)$. We also note that Alg 1 did not obtain the best objective value among our proposed algorithms although the indices were removed in a greedy manner at each iteration. Alg 2 can achieve much better result than that of Alg 1 with much less computing time, hence it is more preferable than Alg 1. Alg 3 can obtain better results than that of Alg 2, and it costs less computation time than Alg 2 as it requires less iterations. The above results ran counter to our expectation, and the strategy of dropping all active constraints at each iteration seems to be empirically good for implementing backward selection though there is no theory to guarantee that this is always true.

To investigate the performance of our proposed algorithms more precisely, we will examine the performance change of the test algorithms as the problem settings change, i.e., we let one or several problem settings vary while keeping the rest fixed, then check the performance changes.

We first examine the performance changes as the number of retained constraints (i.e., p) changes. With two fixed settings of duplet (m, n), we let p vary between 5 and 30, and the average optimal values and computation times over 10 runs on random instances are shown in Figure 1. We also show the ratios of optimal values between our algorithm and ILP (2nd subfigure in each row).

We see from Figure 1 that ILP always obtained the best objective value, but the gap between the objective value of our algorithms and ILP is becoming smaller as p increases. This trend implies that our algorithms are more preferable for larger setting of p in terms of objective value. In addition, the ILP is becoming more time-consuming as p increases while the time efficiency of our algorithms are hardly affected as p changes.

In the next test, let us reverse the roles of n and p, i.e., with fixed settings of (m, p), we let the dimension (i.e., n) vary between 5 and 30 with interval 1, and the results are shown in Figure 2.

We observe from Figure 2 that the ratios between the objective values obtained by the proposed algorithms and that obtained by ILP tends to decrease as n increases. This implies that larger setting of n is less preferable for our algorithms. Additionally, it is interesting to note that the computation times of ILP/Alg 1/Alg 2 tend to increase as n increases, while Alg 3 has the opposite tend.

Figure 1: Objective values, ratio of objective values and computation time v.s. number of retained constraints; Row 1: $(m,n) = (100, 10)$; Row 2: $(m,n) = (50, 50)$.

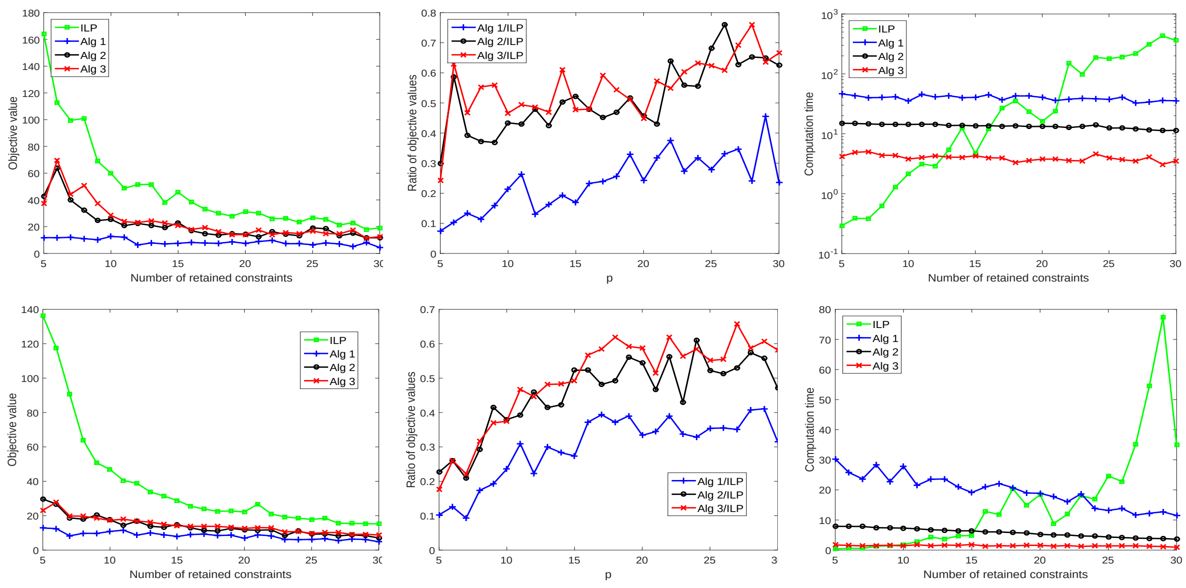
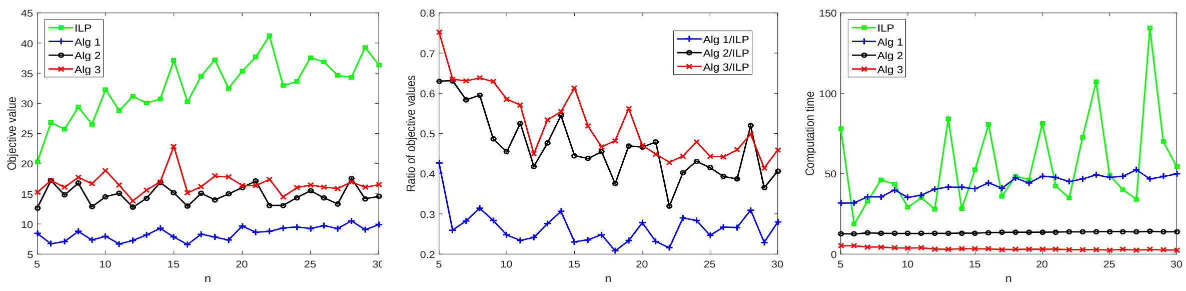


Figure 2: Objective values/computation time v.s. n, $(m,p) = (100, 20)$.

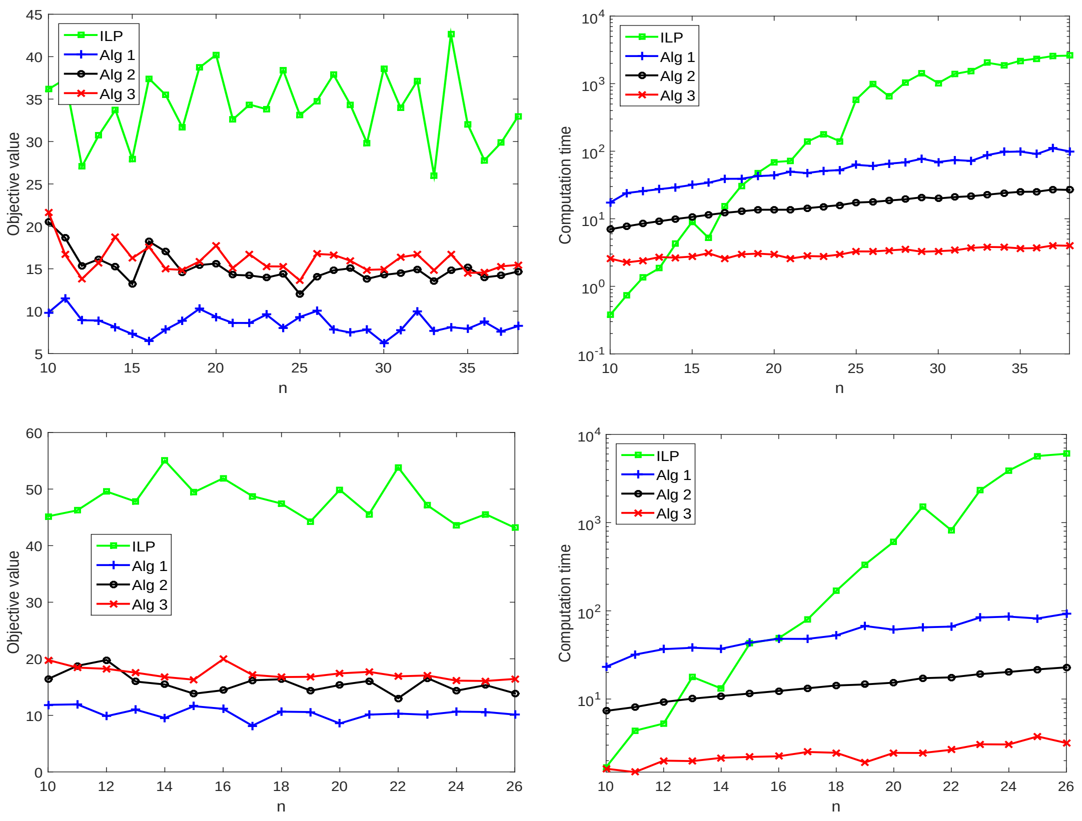


In the above two tests, we only let one parameter change. To investigate the performance change as the problem settings (m,n,p) increase simultaneously, we let n vary in an interval with two fixed ratios of $m:n:p$, and examine the performance change of test algorithms. The results are shown in Figure 3.

We observe from Figure 3 that the ILP always obtained the best objective value which coincides with the results of previous tests. We also note that the ratios between the objective value obtained by our algorithms and ILP are almost unchanged as (m,n,p) increase: precisely speaking, the ratio is around 0.25 for Alg 1 vs ILP, and around 0.5 for Alg 2/Alg 3 vs ILP. The computation time of ILP becomes increasingly expensive as the triplet (m,n,p) increase. In contrary, the computation time of the proposed algorithms increases mildly as (m,n,p) increase. For instance, the computation times of ILP and Alg 3 are around 0.4 and 2.5 seconds respectively when $(m,n,p) = (50, 10, 10)$ while that are around 2500 and 4 seconds respectively when $(m,n,p) = (200, 40, 40)$, i.e., the increments of computation time of ILP and Alg 3 are around 6000 times and 0.6 times respectively as the settings of (m,n,p) have been quadrupled. The above observation indicates that the advantage of our algorithms are two-fold. On one hand, the gap between the objective values of our algorithms and ILP are nonincreasing as (m,n,p) increase; on the other hand, the time efficiency of our algorithms is dimensionally scalable. Hence our algorithms are suitable for handling large-scale problems.

As an extension, an additional test was done by changing the generation rule of objective function, i.e., replacing `c=rand(n,1)` with `c=randn(n,1)`, while other procedures to construct the problem are left unchanged. It is clear that this change allows some entries in c to be negative, and we can observe that whether

Figure 3: Objective values/computation time v.s. n; Row 1: $m:n:p = 5:1:1$; Row 2: $m:n:p = 5:5:1$.



our algorithms can still give satisfactory results in this case. The results are shown in Table 3.2.

Table 3.2 Numerical results on random problems ($c = randn(n, 1)$)

(m,n,p)	ILP		Alg 1		Alg 2		Alg 3	
	Obj	Time	Obj	Time	Obj	Time	Obj	Time
(50, 10,10)	11.418	0.191	4.138	7.604	8.983	3.385	10.772	1.904
(50, 10,20)	4.392	0.431	2.069	5.589	4.137	2.587	4.102	1.507
(50, 50,10)	22.005	0.630	5.204	10.232	13.201	3.496	14.550	1.095
(50, 50,20)	8.004	1.639	3.461	8.411	4.830	2.741	6.351	0.854
(100,10,20)	13.747	4.200	3.690	13.931	10.592	6.605	12.419	3.840
(100,10,30)	6.106	49.656	2.498	13.794	4.290	5.968	5.525	2.809
(100,100,10)	49.454	5.245	6.331	30.409	13.584	8.577	22.654	2.003
(100,100,20)	21.453	114.296	5.295	26.356	8.483	7.752	12.842	1.730
(200,200,20)	N/A	>7200	11.026	204.174	17.200	49.211	28.823	5.802

The results in Table 3 coincide with that shown in Table 3. The ILP still achieves the best objective value under all settings, while our algorithms can obtain a result whose objective value is 45%-95% of that obtained by ILP. Recalling that the ratio of objective values is 35%-75% when c is generated by `c=rand(n,1)`, hence our algorithms perform better when c is generated by `c=randn(n,1)` w.r.t. objective value. Similar to the results of previous tests, the time efficiency of ILP is not dimensionally scalable, and it is not suitable for handling large-scale problems. In contrary, our algorithms have much better dimensional scalability of time efficiency, e.g., when the triplet (m, n, p) increase from $(50, 50, 10)$ to $(100, 100, 20)$, the computation time of ILP grows by around 190 times while that of Alg 3 only grows by around 0.6 time. Among our proposed algorithms, Alg 3 gives the best objective values with much less computation time under most problem settings. Therefore, the speed advantage and performance robustness of the proposed algorithms are invulnerable to the change of generation rule of c.

From the preliminary experimental results shown in this section, we can draw conclusion that, compared with standard MIP solvers which can obtain a global solution, our algorithms can obtain a moderately good result with much less computing time. Further experimental results reveal that the speed performances of our algorithms are dimensionally scalable while the computational cost of MIP solvers grows exponentially with the dimension of problem. According to additional experimental results, their performances are also stable to the change of problem settings and problem generation rules. The efficiency of the third algorithm is especially satisfactory in terms of both objective value and computation time. This may indicate that the strategy of removing a batch of active constraints at each step is a preferable strategy to implement backward selection scheme for solving CRP model though the reason is still unclear and deserve further studies.

4 Conclusions and discussion

In this paper, we propose a novel optimization model entitled constraints reduction programming (CRP) which stems from some important applications. Due to its nonconvexity and inherent high combinatorial complexity, it is difficult to obtain its global solution. Traditional MIP solvers can be used to obtain a global solution of small-scale CRP problems, but its time efficiency is not dimensionally scalable, hence it is not suitable for handling large-scale problems. As a remedy, we propose three greedy algorithms based on the classical backward selection scheme. Due to their greedy nature, it is difficult to analyze the theoretical property of the proposed algorithms. However, the proposed algorithms exhibit the following three advantages over MIP solvers: first, they are almost parameter free; next, they are easy to implement; finally, they show stunningly satisfactory numerical performance.

Due to the importance of the underlying optimization model and the inspiring numerical performance of the proposed algorithms, there is a pressing need to improve the optimization model as well as the proposed algorithms. For instance, we can develop two-way selection scheme with potentially better ability to escape from local optimum. However, this is left to the future works.

Acknowledgement

The authors would like to give thanks to Liping Wang from Nanjing University of Aeronautics and Astronautics for her valuable suggestions on this paper.

Remark

Although the numerical performance of the proposed greedy approaches are satisfactory, we are unable to do theoretical analysis. Therefore, we call for coauthors who are interested in the content of our manuscript or familiar with the analysis method for the proposed approaches.

References

[1] W. Achtziger and C. Kanzow. Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications. *Math. Program.*, 114(1):69–99, 2008.

[2] Ahmed Al-ani. Feature subset selection using ant colony optimization. *Int. J. Comput. Intell.*, 2:53–58, 2005.

[3] E. Balas. Disjunctive programming. *Ann. Discrete Math.*, 5:3–51, 1979.

[4] A. Beck and Y. Vaisbourd. The sparse principal component analysis problem: Optimality conditions and algorithms. *J. Optim. Theory Appl.*, 170:119–143, 2016.

[5] P. Belotti and et.al. On handling indicator constraints in mixed integer programming. *Comput. Optim. Appl.*, 2016.

[6] A. Charnes, W. W. Cooper, and G. Symmonds. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Manage. Sci.*, 4:235–263, 1958.

[7] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *In ICML, Bellevue, WA*, pages 1057–1064, 2011.

[8] A. d'Aspremont. Identifying small mean-reverting portfolios. *Quant. Finance*, 11(3):351–364, 2011.

[9] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constr. Approx.*, 13(1):57–98, 1997.

[10] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Technical report, Department of Statistics, Stanford University, 2006.

[11] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 2004.

[12] A. C. Gilbert, S. Muthukrishnan, and M. J. Strauss. Approximation of functions over redundant dictionaries using coherence. In *SODA, Baltimore, MD*, pages 243–252, 2003.

[13] P. R. Goundan and A. S. Schulz. Revisiting the greedy approach to submodular set function maximization. *Submitted*, 2009. http://www.optimization-online.org/DB_FILE/2007/08/1740.pdf.

[14] M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

[15] T. Koch, B. Hiller, M.E. Pfetsch, and L. Schewe. *Evaluating Gas Network Capacities.* SIAM-MOS series on Optimization. SIAM, Philadelphia, 2015.

[16] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Proces.*, 41(12):3397–3415, December 1993.

[17] A. Miller. *Subset Selection in Regression (2nd edition).* Chapman and Hall/CRC, 2002.

[18] J. Misra, W. Schmitt, D. Hwang, L. L. Hsiao, S. Gullans, G. Stephanopoulos, and G. Stephanopoulos. Interactive exploration of microarray gene expression patterns in a reduced dimensional space. *Genome Res.*, 12(7):1112–1120, 2002.

[19] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995.

[20] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. A.*, 26(3):301–321, Apr 2008.

[21] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proc. off the 27th Annual Asilomar Conference on Signals, Systems, and Computers*, pages 40–44, 1993.

[22] T. Sato, Y. Takano, R. Miyashiro, and A. Yoshise. Feature subset selection for logistic regression via mixed integer optimization. *Comput. Optim. Appl.*, 64:865–880, 2016.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[24] A. L. Tits, P. A. Absil, and W. P. Woessner. Constraint reduction for linear programs with many inequality constraints. *SIAM J. Optim.*, 17(1):119–146, 2006.

[25] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50(10):2231–2242, 2004.

[26] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inform. Theory*, 53(12):4655–4666, 2007.

[27] S. Weisberg. *Applied Linear Regression.* Wiley, New York, 1980.