




Article

Carousel Greedy Algorithms for Feature Selection in Linear Regression

Jiaqi Wang ^{1,*} , Bruce Golden ^{2,*}  and Carmine Cerrone ³ ¹ Department of Mathematics, University of Maryland, College Park, MD 20742, USA² Robert H. Smith School of Business, University of Maryland, College Park, MD 20742, USA³ Department of Economics and Business Studies, University of Genoa, 16126 Genoa, Italy; carmine.cerrone@unige.it

* Correspondence: jqwang@umd.edu (J.W.); bgolden@umd.edu (B.G.)

Abstract: The carousel greedy algorithm (CG) was proposed several years ago as a generalized greedy algorithm. In this paper, we implement CG to solve linear regression problems with a cardinality constraint on the number of features. More specifically, we introduce a default version of CG that has several novel features. We compare its performance against stepwise regression and more sophisticated approaches using integer programming, and the results are encouraging. For example, CG consistently outperforms stepwise regression (from our preliminary experiments, we see that CG improves upon stepwise regression in 10 of 12 cases), but it is still computationally inexpensive. Furthermore, we show that the approach is applicable to several more general feature selection problems.

Keywords: carousel greedy; feature selection; linear regression



Citation: Wang, J.; Golden, B.; Cerrone, C. Carousel Greedy Algorithms for Feature Selection in Linear Regression. *Algorithms* **2023**, *16*, 447. <https://doi.org/10.3390/a16090447>

Academic Editors: Sándor Szénási and Gábor Kertész

Received: 7 August 2023

Revised: 11 September 2023

Accepted: 14 September 2023

Published: 19 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The carousel greedy algorithm (CG) is a generalized greedy algorithm that seeks to overcome the traditional weaknesses of greedy approaches. A generalized greedy algorithm uses a greedy algorithm as a subroutine in order to search a more expansive set of solutions with a small and predictable increase in computational effort. To be more specific, greedy algorithms often make poor choices early on and these cannot be undone. CG, on the other hand, allows the heuristic to correct early mistakes. The difference is often significant.

In the original paper, Cerrone et al. (2017) [1] applied CG to several combinatorial optimization problems such as the minimum label spanning tree problem, the minimum vertex cover problem, the maximum independent set problem, and the minimum weight vertex cover problem. Its performance was very encouraging. More recently, it has been applied to a variety of other problems; see Table 1 for details.

CG is conceptually simple and easy to implement. Furthermore, it can be applied to many greedy algorithms. In this paper, we will focus on using CG to solve the well-known linear regression problem with a cardinality constraint. In other words, we seek to identify the k most important variables, predictors, or features out of a total of p . The motivation is quite straightforward. Stepwise regression is a widely used greedy heuristic to solve this problem. However, the results are not as near-optimal as we would like. CG can generate better solutions within a reasonable amount of computing time.

In addition to the combinatorial optimization problems studied in Cerrone et al. (2017) [1], the authors also began to study stepwise linear regression. Their experiments were modest and preliminary. A pseudo-code description of their CG stepwise linear regression approach is provided in Algorithm 1. For the problem to be interesting, we assume that the number of explanatory variables to begin with (say, p) is large and the number of these variables that we want in the model (say, k) is much smaller.

Table 1. Recent applications of CG.

Year	Problem	Authors
2023	Knapsack Problem with Forfeits	D’Ambrosio et al. [2]
2022	Knapsack Problem with Forfeits	Capobianco et al. [3]
2022	Maximum Network Lifetime Problem with Time Slots and Coverage Constraints	Cerulli et al. [4]
2021	Grocery Distribution Plans in Urban Networks with Street Crossing Penalties	Cerrone et al. [5]
2021	Finding Minimum Positive Influence Dominating Sets in Social Networks	Shan et al. [6]
2020	Knapsack Problem with Forfeits	Cerulli et al. [7]
2020	Remote Sensing with Unmanned Aerial Vehicles (UAVs)	Hammond et al. [8]
2020	Close-Enough Traveling Salesman Problem	Carrabs et al. [9]
2019	Community Detection in Complex Networks	Kong et al. [10]
2019	Strong Generalized Minimum Labeling Spanning Tree Problem	Cerrone et al. [11]
2019	Sentiment Analysis	Hadi et al. [12]
2018	A Distribution Problem	Cerrone et al. [13]
2017	Maximum Network Lifetime Problem with Interference Constraints	Carrabs et al. [14]

Algorithm 1 Pseudo-code of carousel greedy for linear regression from [1].

Input I (I is the set of explanatory variables)
Input n (n is the number of explanatory variables you want in the model)
 1: Let $S \leftarrow$ model containing all the explanatory variables in I
 2: $R \leftarrow$ partial solution produced by removing from S , $|S| - n$ elements according to the backward selection criteria
 3: **for** an iterations **do**
 4: remove from tail of R an explanatory variable
 5: according to the forward selection criteria, add an element to head of R
 $\triangleright R$ is an ordered sequence, where its head is the side for adding and the tail for removing
 6: **end for**
 7: return R

The experiments were limited, but promising. The purpose of this article is to present a more complete study of the application of CG to linear regression with a constraint on the number of explanatory variables. In all of the experiments in this paper, we assume a cardinality constraint. Furthermore, in the initial paper on CG, Cerrone et al. (2017) [1] worked with two datasets for linear regression and also assumed a target number of explanatory/predictor variables. This is an important variant of linear regression and it ensures that different subsets of variables can be conveniently compared using RSS.

2. Linear Regression and Feature Selection

The rigorous mathematical definition of the problem is:

$$\min_{\theta} \text{RSS} = \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij}\theta_j - y_i \right)^2, \tag{1}$$

$$\text{subject to } \sum_{j=1}^p I_{\theta_j \neq 0} \leq k, \tag{2}$$

where the following notation applies:

- RSS: the residual sum of squares,
- $X \in \mathbb{R}^{n \times p}$: the independent variables,
- X_{ij} : the i^{th} row and j^{th} column of X ,
- $y \in \mathbb{R}^n$: the dependent variable,
- y_i : the i^{th} element of y ,

$\theta \in \mathbb{R}^p$: the coefficient vector of explanatory variables,
 θ_j : the j^{th} element of θ ,
 n : the number of observations,
 p : the total number of explanatory variables (features),
 k : the number of explanatory variables we want in the model, and
 $I_{\theta_j \neq 0}$: equals 1 if $\theta_j \neq 0$ is true and 0 otherwise.

While we use RSS (i.e., OLS or ordinary least squares) as the objective function, other criteria, such as AIC [15], C_p [16], BIC [17], and RIC [18], are possible. We point out that our goal in this paper is quite focused. We do not consider the other criteria mentioned above. In addition, we do not separate the data into training, test, and validation sets as is commonly the case in machine learning models. Rather, we concentrate on minimizing RSS over the training set. This is fully compatible with the objective in linear regression and best subset selection.

There are three general approaches to solving the problem in (1) and (2). We summarize them below.

1. Best subset selection. This direct approach typically uses mixed integer optimization (MIO). It has been championed in articles by Bertsimas and his colleagues [19,20]. Although MIO solvers have become very powerful in the last few decades and they can solve problems much faster than in the past, running times can still become too large for many real-world datasets. Zhu et al. [21] has recently proposed a polynomial time algorithm, but it requires some mild conditions on the dataset and it works with high probability, but not always.
2. Regularization. This approach was initially used to address overfitting in unconstrained regression so that the resulting model would behave in a *regular* way. In this approach, the constraints are moved into the objective function with a penalty term. Regularization is a widely used method because of its speed and high performance in making predictions on test data. The most famous regularized model, lasso [22], uses the L^1 -penalty as shown below:

$$\min_{\theta} \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij} \theta_j - y_i \right)^2 + \lambda \sum_{j=1}^p |\theta_j|. \quad (3)$$

As λ becomes larger, it will prevent θ from having a large L^1 -norm. At the same time, the number of nonzero θ_i values will also become smaller. For any k , there exists a λ such that the number of nonzero θ_i values is roughly k , but the number can sometimes jump sharply. This continuous optimization model is very fast, but there are some disadvantages. Some follow-up papers using ideas such as adaptive lasso [23], L0Learn [24], trimmed lasso [25], elastic net [26], and MC+ [27] appear to improve the model by modifying the penalty term. Specifically, L0Learn [24] uses the L^0 -penalty as shown below:

$$\min_{\theta} \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij} \theta_j - y_i \right)^2 + \lambda \sum_{j=1}^p I_{\theta_j \neq 0}. \quad (4)$$

For sparse high-dimensional regression, some authors [28,29] use L^2 regularization without removing the cardinality constraint.

3. Heuristics. Since the subset selection problem is hard to solve optimally, a compromise would be to find a very good solution quickly using a heuristic approach. Stepwise regression is a widely used heuristic for this problem. An alternating method that can achieve a so-called partial minimum is presented in [30]. SparseNet [31] provides a coordinate-wise optimization algorithm. CG is another heuristic approach. One idea is to apply CG from a random selection of predictor variables. An alternative is to apply CG to the result of stepwise regression in order to improve the solution. We might expect the latter to outperform MIO in terms of running time, but not in terms

of solution quality. In our computational experiments, we will compare approaches with respect to RSS on training data only.

There are different variants of the problem specified in (1) and (2). We can restrict the number of variables to be at most k , as in (1) and (2). Alternatively, we can restrict the number of variables to be exactly k . Finally, we can solve an unrestricted version of the problem. We point out that when we minimize RSS over training data only, it always helps to add another variable. Therefore, when the model specifies there will be at most k variables, the solution will involve exactly k variables.

In response to the exciting recent work in [19,20] on the application of highly sophisticated MIO solvers (e.g., Gurobi) to solve the best subset regression problem posed in (1) and (2), Hastie et al. [32] have published an extensive computational comparison in which best subset selection, forward stepwise selection, and lasso are evaluated on synthetic data. The authors used both a training set and a test set in their experiments. They implemented the mixed integer quadratic program from [20] and made the resulting R code available in [32]. They found that best subset selection and lasso work well, but neither dominates the other. Furthermore, a relaxed version of lasso created by Meinshausen [33] is the overall winner.

Our goal in this paper is to propose a smart heuristic to solve the regression problem where k is fixed in advance. The heuristic should be easy to understand, code, and use. It should have a reasonably fast running time, although it will require more time than stepwise regression. We expect that for large k , it will be faster than best subset selection.

In this paper, we will test our ideas on the three real-world datasets from the UCI ML Repository (see <https://archive.ics.uci.edu/>, accessed on 11 April 2023) listed below:

1. CT (Computerized Tomography) Slice Dataset: $n = 10,001$, $p = 384$;
2. Building Dataset: $n = 372$, $p = 107$; and
3. Insurance Dataset: $n = 9822$, $p = 85$.

Since we are most interested in the linear regression problem where k is fixed, we seek to compare the results of best subset selection, CG, and stepwise regression. We will use the R code from [32], our CG code, and the stepwise regression code from (<http://www.science.smith.edu/~jcrouser/SDS293/labs/lab8-py.html>, accessed on 11 September 2022) in our experiments. For now, we can say that best subset selection takes much more time than stepwise regression and it typically obtains much better solutions. Our goal will be to demonstrate that CG represents a nice compromise approach. We expect CG solutions to be better than stepwise regression solutions and the running time to be much faster than it is for the best subset selection solutions.

We use the following hardware: CPU 11th Gen Intel(R) Core(TM) i7-11700F @ 2.50 GHz 2.50 GHz. The algorithms in this paper are implemented in Python 3.9.12, unless otherwise mentioned. CG is implemented in Python 3.9.12 without parallelization, unless otherwise mentioned. On the other hand, when Gurobi 10.0.0 is used, all of the 16 threads are utilized (in parallel).

3. Algorithm Description and Preliminary Experiments

3.1. Basic Algorithm and Default Settings

In contrast to the application of CG to linear regression in [1], shown in Algorithm 1, we present a general CG approach to linear regression with a cardinality constraint in Algorithm 2.

Algorithm 2 Pseudo-code of carousel greedy for linear regression with a cardinality constraint.

Input $I, k, S, \beta, \alpha, \gamma$

Output R

- 1: $R \leftarrow S$ with $\beta|S|$ variables dropped from head $\triangleright \beta|S|$ rounded to the nearest integer
- 2: $REC \leftarrow R$
- 3: $RECRSS \leftarrow$ RSS of R
- 4: **for** $\alpha(1 - \beta)|S|$ iterations **do**
- 5: Remove γ variables from the tail of R
- 6: Add γ variables from $I - R$ to the head of R one by one according to forward selection criterion
- 7: **if** RSS of $R < REC$ **then**
- 8: $REC \leftarrow R$
- 9: $RECRSS \leftarrow$ RSS of R
- 10: **end if**
- 11: **end for**
- 12: $R \leftarrow$ Use forward selections to add elements to REC one by one until k variables are selected
- 13: **return** R

Here, the inputs are:

I : the set of explanatory variables,

k : the number of variables we want in the model,

S : the initial set of variables with $|S| = k$,

β : the percentage of variables we remove initially,

α : the number of carousel loops where we have $(1 - \beta)|S|$ carousel steps in each loop, and

γ : the number of variables we remove/add in each carousel step.

The starting point of Algorithm 2 is a feasible variable set S with order. We drop a fraction of β from S . Then, we start our α carousel loops of removing and adding variables. In each carousel step, we remove γ variables from the tail of R and add γ variables to the head of R one by one according to forward selection. The illustration of head and tail is shown in Figure 1. Each time we finish a carousel step, the best set of variables in terms of RSS is recorded. When carousel loops are finished, we add variables to the best recorded set according to the forward selection criterion one by one until a feasible set of k variables is selected.

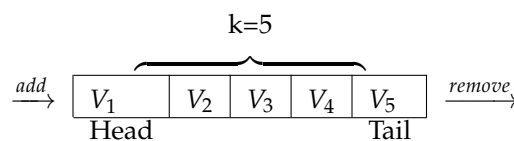


Figure 1. An illustration of head and tail for $k = 5, \beta = 0$.

For a specific linear regression problem, I and k are fixed. S, β, α, γ are the parameters which must be set by the user. In general, the best values may be difficult to find and they may vary widely from one problem/dataset to another.

As a result, we start with a default set of parameter values and run numerous experiments. These parameter values work reasonably well across many problems. We present the pseudo-code and flowchart for this simple implementation of a CG approach in Algorithm 3 and Figure 2.

Algorithm 3 Pseudo-code of the default version of carousel greedy we recommend for linear regression with a cardinality constraint.

Input I, k

Output R

```

1:  $S \leftarrow$  the solution produced by forward stepwise regression  $\triangleright$  In the order of selection
2:  $R \leftarrow S$ 
3:  $LastImpro = 0$ 
4:  $RECRSS =$  RSS of  $R$ 
5: while  $LastImpro < k$  do
6:    $LastImpro = LastImprove + 1$ 
7:   Remove 1 variable from the tail of  $R$ 
8:   Add 1 variable to the head of  $R$  according to forward selection
9:   if RSS of  $R < RECRSS$  then
10:     $LastImpro = 0$ 
11:   end if
12: end while
13: return  $R$ 

```

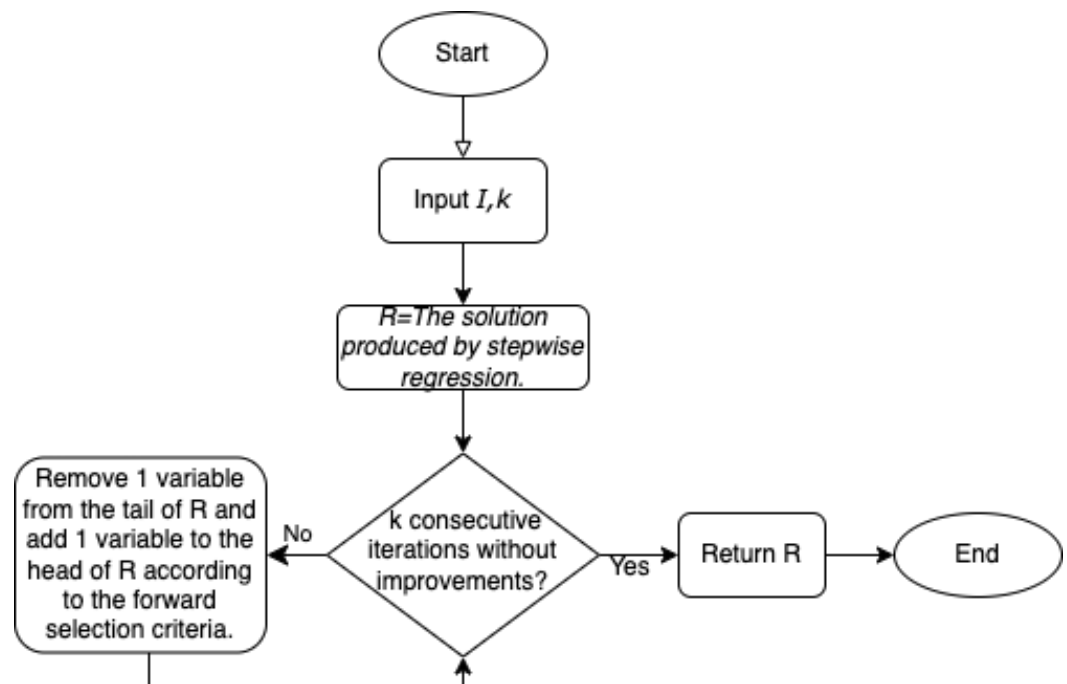


Figure 2. Flowchart of default CG.

In other words, the default parameters are:

S = the result of stepwise regression with k variables,

$\beta = 0$,

α is set in an implicit way such that we have k consecutive carousel steps without improvement of RSS, and

$\gamma = 1$.

3.2. Properties

There are a few properties for this default setting:

1. The RSS of the output will be at least as good as the RSS from stepwise regression.
2. The RSS of the incumbent solution is always monotonically decreasing.
3. When the algorithm stops, it is impossible to achieve further improvements of RSS by running additional carousel greedy steps.

- The result is a so-called full swap inescapable (FSI(1)) minimum [24], i.e., no interchange of an inside element and an outside element can improve the RSS.

3.3. Preliminary Experiments

The following experiments show how CG evolves the solution from stepwise regression. As shown in Figure 3, CG consistently improves the RSS of the stepwise regression solution gradually in the beginning and stabilizes eventually. A final horizontal line segment of length 1 indicates that no further improvements are possible.

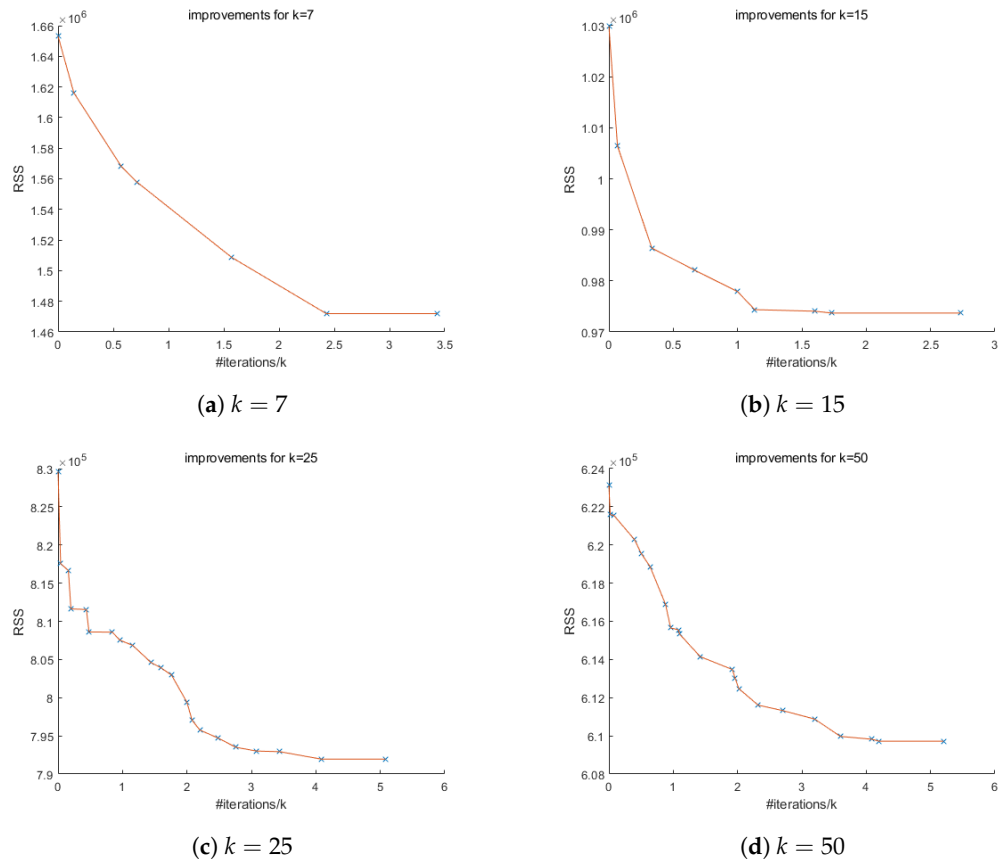


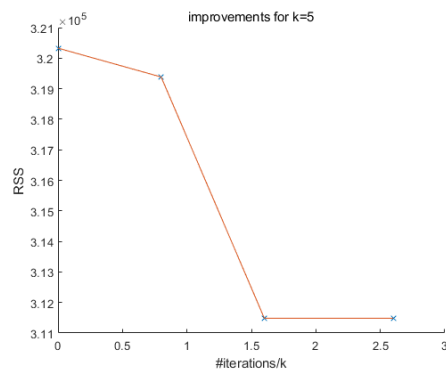
Figure 3. Improvements from carousel greedy with stepwise initialization for the CT slice dataset. The RSS of stepwise regression is at $x = 0$.

For the CT slice dataset we are using, the best subset selection cannot completely solve the problems in a reasonable time. When we limit the time of the best subset selection algorithm to a scale similar to CG, the RSS of its output is not as good as CG. Even if we give best subset selection more than twice as much time, the result is still not as good. The results are shown in Table 2 (the best results in Table 2 for each k are indicated in bold).

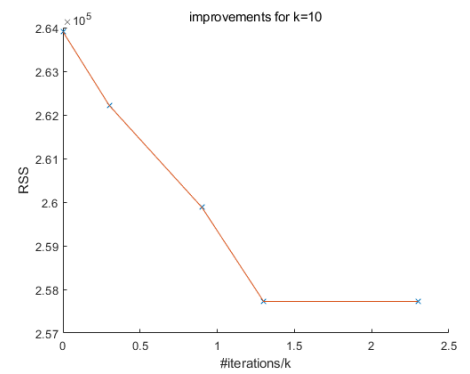
From Figures 4 and 5, some improvements from stepwise regression solutions can be observed. However, as the number of observations (n) and the number of variables (p) in the dataset become smaller, the model itself becomes simpler, in which case there will be less room for CG to excel and the number of improvements also becomes smaller. Figure 5b,c show no improvements from stepwise regression. This might mean the stepwise regression solution is already relatively good. In that case, we may want to use other initializations to see if we can find better solutions.

Table 2. Comparisons of stepwise regression, CG, and best subset with different time limits.

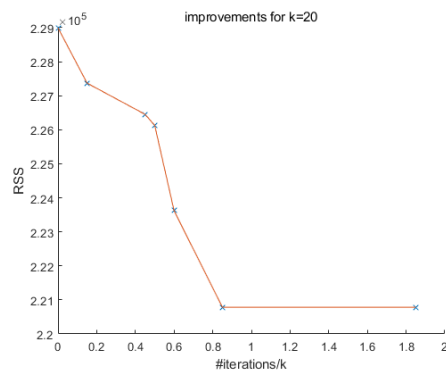
	$k = 7$		$k = 15$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	1,653,208	5.55	1,029,899	16.08
CG with stepwise initialization	1,471,932	28.45	973,695	97.82
Best subset (warm start + MIP)	1,570,721	28.35	1,015,076	97.69
Best subset (warm start + MIP)	1,570,681	100.00	999,294	250.00
	$k = 25$		$k = 50$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	829,608	38.45	623,118	132.24
CG with stepwise initialization	791,960	440.14	609,722	1409.88
Best subset (warm start + MIP)	819,911	441.75	611,207	1413.88
Best subset (warm start + MIP)	807,925	1000.00	610,542	3600.00



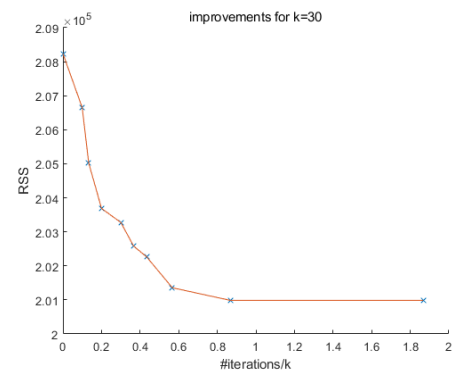
(a) $k = 5$



(b) $k = 10$



(c) $k = 20$



(d) $k = 30$

Figure 4. Improvements from carousel greedy with stepwise initialization for the Building dataset.

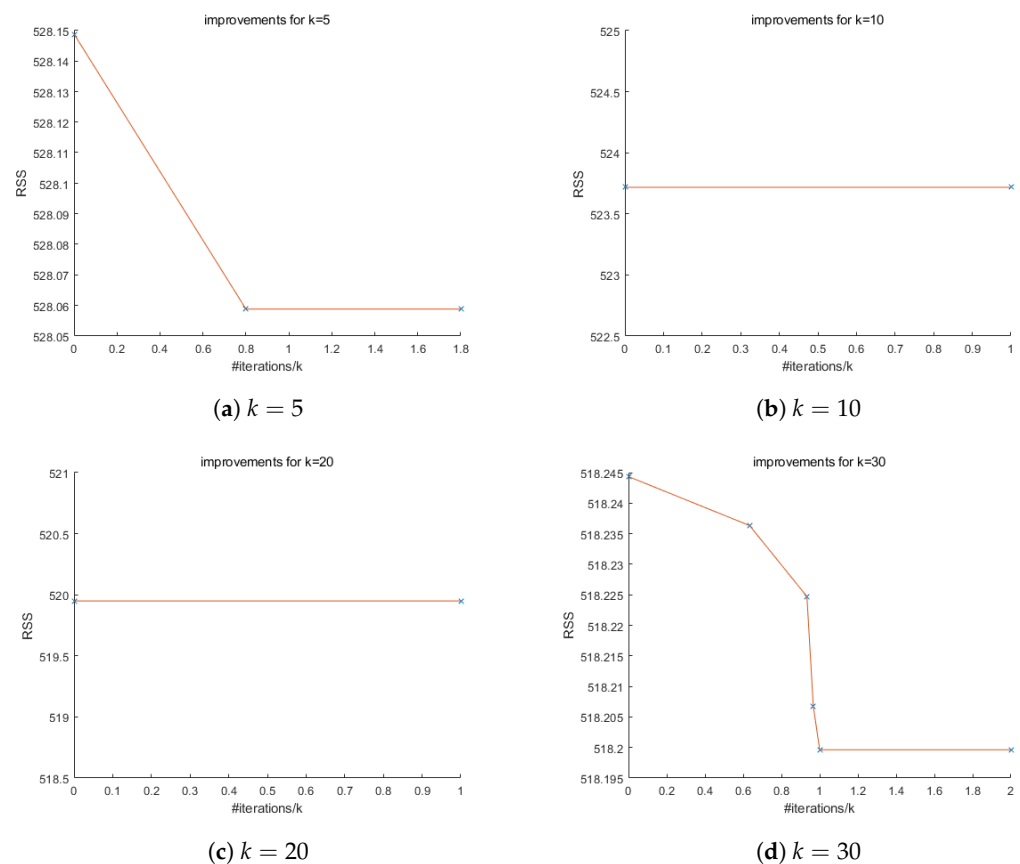


Figure 5. Improvements for carousel greedy with stepwise initialization for the Insurance dataset.

3.4. Stepwise Initialization and Random Initialization

As shown in Algorithm 2, there can be different initializations in a general CG algorithm. We might be able to find other solutions by choosing other initializations. A natural choice would be a complete random initialization.

We run the experiments for CG with stepwise initialization and random initialization for the CT slice dataset. For random initialization, we run 10 experiments and look at the average or minimum among the first 5 and among all 10 experiments.

From Table 3, we can see that the average RSS of random initialization is similar to that of stepwise initialization and usually takes slightly less time. The results are also quite close between different random initializations for most cases. However, if we run random initialization multiple times, for example, 5~10 times, the best output will very likely be better than for stepwise initialization.

We now look back at the cases of Figure 5b,c using random initializations. We run 10 experiments on the Insurance dataset for $k = 10$ and 10 for $k = 20$ and plot the best, in terms of the final RSS, out of 10 experiments.

As shown in Figure 6, although we are able to find better solutions using random initialization than stepwise initialization, the improvements are very small. In this case, we are more confident that the stepwise regression solution is already good, and CG provides a tool to verify this.

In practice, if time permits or parallelization is available, we would suggest running CG with both stepwise initialization and multiple random initializations and taking the best result. Otherwise, stepwise initialization would be a safe choice.

Table 3. CG with stepwise initialization and random initialization for the CT slice dataset.

	$k = 7$		$k = 15$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	1,653,208	5.55	1,029,899	16.08
CG with stepwise initialization	1,471,932	28.45	973,695	97.82
CG with random initialization (average over 10 experiments)	1,471,932	13.52	972,886	100.44
(min RSS over 10 experiments)	1,471,932		970,712	
CG with random initialization (average over 5 experiments)	1,471,932	16.88	972,674	96.32
(min RSS over 5 experiments)	1,471,932		970,712	
	$k = 25$		$k = 50$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	829,608	38.45	623,118	132.24
CG with stepwise initialization	791,960	440.14	609,722	1409.88
CG with random initialization (average over 10 experiments)	791,581	300.25	612,304	1057.20
(min RSS over 10 experiments)	788,836		606,865	
CG with random initialization (average over 5 experiments)	791,380	273.90	612,404	1120.17
(min RSS over 5 experiments)	788,836		606,865	

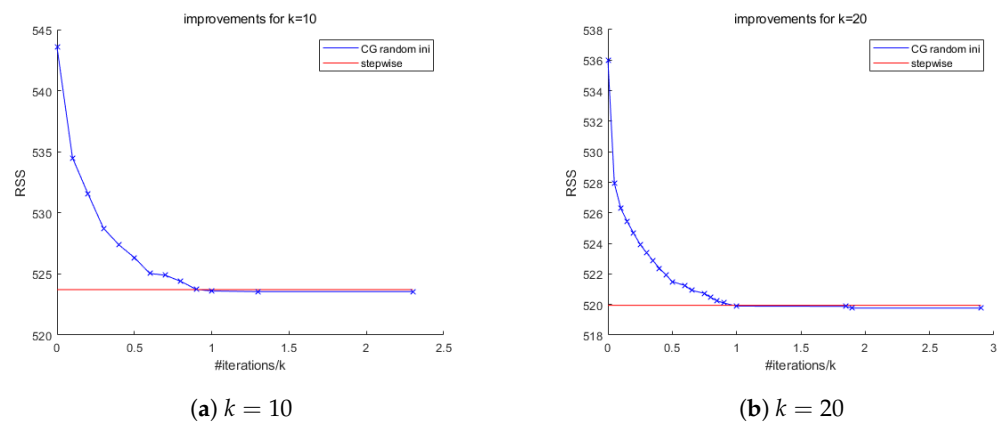


Figure 6. Improvements from carousel greedy with random initialization for the Insurance dataset.

3.5. Gurobi Implementation of Best Interchange to Find an FSI(1) Minimum

An alternative approach to Algorithm 3 is the notion of an FSI(1) minimum. This is a local minimum as described in Section 3.2. The original method for finding an FSI(1) minimum in [24] was to formulate the best interchange as the following best interchange MIP (BI-MIP) and apply it iteratively as in Algorithm 4. We will show that CG obtains results comparable to those from FSI(1), but without requiring the use of sophisticated integer programming software.

$$\text{BI-MIP: } \min \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij} \theta_j - y_i \right)^2, \tag{5}$$

$$\text{subject to } -Mz_i \leq \theta_i \leq Mz_i, \quad \forall i \in [p], \tag{6}$$

$$z_i \leq w_i, \quad \forall i \in S, \tag{7}$$

$$\sum_{i \in S^c} z_i \leq 1, \tag{8}$$

$$\sum_{i \in S} w_i \leq |S| - 1, \tag{9}$$

$$\theta_i \in \mathbb{R}, \quad \forall i \in [p], \tag{10}$$

$$z_i \in \{0, 1\}, \quad \forall i \in [p], \tag{11}$$

$$w_i \in \{0, 1\}, \quad \forall i \in S. \tag{12}$$

Here, we start from an initial set S and try to find a best interchange between a variable inside S and a variable outside S . The decision variables are θ , w , and z . z_i 's indicate the nonzeros in θ , i.e., if $z_i = 0$, then $\theta_i = 0$. w_i 's indicate whether we remove variable i from

S , i.e., if $w_i = 0$, then variable i is removed from S . $[p]$ is defined to be the set $\{1, 2, \dots, p\}$. In (5), we seek to minimize RSS. In (6), for every $i \in [p]$, $z_i = 1$ if θ_i is nonzero and M is a sufficiently large constant. From (7), we see that if variable i is removed, then θ_i must be zero. Inequality (8) means that the number of selected variables outside S is at most 1, i.e., we add at most 1 variable to S . From (9), we see that we remove at least 1 variable from S (the optimal solution removes exactly 1 variable).

In Algorithm 4 for finding an FSI(1) minimum, we solve BI-MIP iteratively until an iteration does not yield any improvement. The pseudo-code is as follows:

Algorithm 4 Pseudo-code of finding an FSI(1) local minimum by Gurobi.

- 1: Initialize $|S| = k$ by forward stepwise regression with coefficients θ
 - 2: **while** TRUE **do**
 - 3: $S' \leftarrow$ Apply BI-MIP to S
 - 4: **if** RSS of $S' \geq$ RSS of S **then**
 - 5: Break
 - 6: **end if**
 - 7: $S = S'$
 - 8: **end while**
 - 9: return S
-

Recall that our default version of CG also returns an FSI(1) local minima. The solutions of the two methods are expected to return equally good solutions on average. We begin with a comparison on the CT slice dataset.

As shown in Table 4, the final RSS by Gurobi is very close to CG (the best results in Table 4 for each k are indicated in bold), but the running time is much longer for small k and not much faster for larger k . This is the case even though Gurobi uses all of the 16 threads by default while CG uses only one thread by default in Python. Therefore, our algorithm is simpler and does not require a commercial solver like Gurobi. It is also more efficient than the BI-MIP by Gurobi in terms of finding a local optima when k is small.

The circumstance can be different for an “ill-conditioned” instance. We tried to apply Algorithm 4 to the Building dataset, but it is a very simple model where $k = 1$ cannot be solved. Meanwhile, CG can solve it without any difficulty. The source of the issue is the quadratic coefficient matrix. The objective of BI-MIP is quadratic. When Gurobi solves quadratic programming, a very large difference between the largest and smallest eigenvalues of the coefficient matrix can bring about a substantial numerical issue. For the Building dataset, the largest eigenvalue is of order 10^{15} , the smallest eigenvalue is of order 10^{-10} . That’s intractable for Gurobi. Therefore, CG is numerically more stable than Algorithm 4 using Gurobi.

Table 4. Iterated BI-MIPs by Gurobi (Algorithm 4) for the CT slice dataset (using up to 16 threads).

	$k = 5$		$k = 7$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	2,129,794	3.41	1,653,208	5.55
CG with stepwise initialization	2,104,917	10.03	1,471,932	28.45
Iterated BI-MIPs by Gurobi with stepwise initialization	2,106,992	59.89	1,471,932	99.64
	$k = 10$		$k = 15$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	1,307,711	8.22	1,029,899	16.08
CG with stepwise initialization	1,198,650	47.63	973,695	97.82
Iterated BI-MIPs by Gurobi with stepwise initialization	1,198,650	161.9	970,712	286.43

Table 4. Cont.

	$k = 20$		$k = 25$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	911,080	25.99	829,608	38.45
CG with stepwise initialization	870,346	214.43	791,960	440.14
Iterated BI-MIPs by Gurobi with stepwise initialization	870,346	303.65	792,731	566.38
	$k = 30$		$k = 35$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	767,448	52.70	723,472	68.81
CG with stepwise initialization	741,317	437.13	700,462	566.73
Iterated BI-MIPs by Gurobi with stepwise initialization	751,680	406.33	699,196	457.45
	$k = 40$		$k = 45$	
	RSS	Time (s)	RSS	Time (s)
Stepwise regression	685,136	85.31	651,250	107.74
CG with stepwise initialization	666,991	453.45	638,087	493.93
Iterated BI-MIPs by Gurobi with stepwise initialization	666,991	416.42	643,517	353.18
	$k = 50$			
	RSS	Time (s)		
Stepwise regression	623,118	132.24		
CG with stepwise initialization	609,722	1409.88		
Iterated BI-MIPs by Gurobi with stepwise initialization	609,478	768.68		

3.6. Running Time Analysis

Each time we add a variable, we need to run the least squares procedure (computing $(X_S^T X_S)^{-1} X_S^T y$, where X_S is the $n \times k$ submatrix of X whose column is indexed by S) a total of p times. For each addition of a variable, from basic numerical linear algebra, the complexity is $O(k^2(k + n))$. Therefore, for each new variable added, the complexity is $O(pk^2(k + n))$. If n is much larger than k , which is often the case, the complexity is about $O(npk^2)$. From the structure of Algorithm 2, we will add a variable $\alpha(1 - \beta)\gamma k$ times. As a result, the complexity of CG is $O(\alpha(1 - \beta)\gamma npk^3)$. We can treat α, β, γ as constants because they are independent of k, n, p . Therefore, the complexity of CG is still $O(npk^3)$.

4. Generalized Feature Selection

In this section, we will discuss two generalized versions of feature selection. Recall the feature selection problem that we discussed is

$$\min_{\theta} \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij} \theta_j - y_i \right)^2, \tag{13}$$

$$\text{subject to } \sum_{j=1}^p I_{\theta_j \neq 0} \leq k. \tag{14}$$

This can be reformulated (big-M formulation) as the following MIP (for large enough $M > 0$):

$$\min_{\theta, z} \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij} \theta_j - y_i \right)^2, \tag{15}$$

$$\text{subject to } -Mz_i \leq \theta_i \leq Mz_i, \quad \forall i \in [p], \tag{16}$$

$$\sum_{i \in [p]} z_i \leq k, \tag{17}$$

$$z_i \in \{0, 1\}, \quad \forall i \in [p]. \tag{18}$$

Here, z_i is the indicator variable for θ_i , where $z_i = 1$ if θ_i is nonzero and $z_i = 0$ otherwise.

The problem can be generalized by adding some constraints.

Case A: Suppose we have sets of variables that are highly correlated. For example, if variables $i, j,$ and k are in one of these sets, then we can add the following constraint:

$$z_i + z_j + z_k \leq 1. \tag{19}$$

If l and m are in another set, we can add

$$z_l + z_m \leq 1. \tag{20}$$

Case B: Suppose some of the coefficients must be bounded if the associated variables are selected. Then we can add the following constraints:

$$l_i z_i \leq \theta_i \leq u_i z_i, \quad \forall i \in [p]. \tag{21}$$

In (21), we allow $l_i = -\infty$ or $u_i = \infty$ for some i to include the case where the coefficients are unbounded from below or above.

CG can also be applied to Cases A and B. Let us restate CG and show how small modifications to CG can solve Cases A and B.

Recall that, in each carousel step of feature selection, we delete one variable and add one. Assume S_d is the set of variables *after* deleting one variable in a step. Then, the problem of adding one becomes: Solve unconstrained linear regression problems on the set $S_d \cup \{l\}$ for each $l \notin S_d$ and return the l with the smallest RSS. Expressed formally, this is

$$\operatorname{argmin}_{l \notin S_d} f(l, S_d). \tag{22}$$

Here $f(l, S_d)$ indicates the RSS we obtain by adding l to S_d . It can be found from the following problem:

$$f(l, S_d) = \min_{\theta} \sum_{i=1}^n \left(\sum_{j \in S_d \cup \{l\}} X_{ij} \theta_j - y_i \right)^2. \tag{23}$$

In (23), only a submatrix of X with size $n \times (|S_d| + 1)$ is involved. That is, we solve a sequence of smaller size unconstrained linear regression problems for each $l \notin S_d$ and compare the results.

Following this idea, by using CG, we solve a sequence of smaller size unconstrained linear regression problems where the cardinality constraint is no longer in any subproblem.

For generalized feature selection, CG has the same framework. The differences are that the candidate variables to be added are limited by the notion of correlated sets for Case A, and the sequence of smaller size linear regression problems become constrained, as in (21), for Case B. The term we designate for the new process of adding a variable is the “generalized forward selection criterion.” Let us start with Case A.

Case A: The addition of one variable is almost the same as for Algorithms 1–3, but the candidate variables should be those not highly correlated with any current variables, instead of any $l \notin S_d$. As an example, suppose variables $i, j,$ and k are highly correlated. (For the sake of clarity, we point out that the data for variable i is contained in $X_{.i}$.) We want no more than one of these in our solution. At the l -th carousel step before adding a variable, we check whether one of these three is in S_d or not. If i is in S_d , we cannot add i or k . If i is not in S_d , we can add i, j, k or any other variable.

Case B: We can add variables as usual, but the coefficients corresponding to some of these variables are now bounded. We will need to solve a sequence of bounded variable least squares (BVLS) problems of the form

$$\operatorname{argmin}_{l \notin S_d} f(l, S_d). \tag{24}$$

Here, $f(l, S_d)$ can be obtained from the following problem:

$$f(l, S_d) = \min_{\theta} \sum_{i=1}^n \left(\sum_{j \in S_d \cup \{l\}} X_{ij} \theta_j - y_i \right)^2, \quad (25)$$

$$\text{subject to } l_i \leq \theta_i \leq u_i, \quad \forall i \in S_d \cup \{l\}. \quad (26)$$

We extract the general subproblem of BVLS from the above:

$$\min_{\theta} \sum_{i=1}^n \left(\sum_{j \in S} X_{ij} \theta_j - y_i \right)^2, \quad (27)$$

$$\text{subject to } l_i \leq \theta_i \leq u_i, \quad \forall i \in S. \quad (28)$$

This is a quadratic (convex) optimization problem within a bounded and convex region (a p -dim box), which can be solved efficiently. There are several algorithms available without the need for any commercial solver. For example, a free and open-source Python library “Scipy” can solve it efficiently. The pseudo-codes for the application of CG and MIP to generalized feature selection can be found in Appendix A.

5. Conclusions and Future Work

In this paper, we propose an application of CG to the feature selection problem. The approach is straightforward and does not require any commercial optimization software. It is also easy to understand. It provides a compromise solution between the best subset selection and stepwise regression. The running time of CG is usually much shorter than that of the best subset selection and a few times longer than that of stepwise regression. The RSS of CG is always at least as good as for stepwise regression, but is typically not as good as for best subset. Therefore, CG is a very practical method for obtaining (typically) near-optimal solutions to the best subset selection problem.

With respect to the practical implications of our work, we point to the pervasiveness of greedy algorithms in the data science literature. Several well-known applications of the greedy algorithm include constructing a minimum spanning tree, implementing a Huffman encoding, and solving graph optimization problems (again, see [1]). In addition, numerous greedy algorithms have been proposed in the machine learning literature over the last 20 years for a wide variety of problems (e.g., see [34–38]). CG may not be directly applicable in every case, but we expect that it can be successfully applied in many of these cases.

We also show that the result of our CG can produce a so-called FSI(1) minimum. Finally, we provide some generalizations to more complicated feature selection problems in Section 4.

The implementation of CG still has some room for improvement. We leave this for future work, but here are some ideas. The computational experiments in this paper are based mainly on Algorithm 3, which includes a set of default parameter values. However, if we work from Algorithm 2, there may be a better set of parameter values, in general, or there may be better values for specific applications. Extensive computational experiments would have to focus on running time and accuracy in order to address this question. Furthermore, we think there are some more advanced results from numerical linear algebra that can be applied to improve the running time of the sequence of OLS problems that must be solved within CG. In addition, we think our approach can be applied to other problems in combinatorial optimization and data science. For example, we are beginning to look into the integration of neural networks and CG. The key idea is to replace the greedy selection function with a specifically trained neural network. CG could be employed to enhance the decisions recommended by the neural network. Again, we hope to explore this in future work.

Author Contributions: Conceptualization, B.G.; methodology, J.W., B.G. and C.C.; software, J.W.; validation, J.W., B.G. and C.C.; formal analysis, J.W.; investigation, J.W.; resources, B.G. and C.C.; data curation, J.W.; writing—original draft preparation, J.W.; writing—review and editing, B.G.; visualization, J.W.; supervision, B.G. and C.C.; project administration, B.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets presented in this study are available in UCI ML Repository at <https://archive.ics.uci.edu/> (accessed on 11 April 2023).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Pseudo-Code of Algorithms for Generalized Feature Selection

For generalized feature selection, all we need to do is to replace all of the forward selection steps in Algorithms 1–4 by generalized forward selection steps. For example, steps 1 and 8 for Algorithm A1 vs. Algorithm 3 reflect this change. Similarly, steps 1 and 3 for Algorithm A2 vs. Algorithm 4 reflect this change. Generalized forward selection includes both Case A and Case B. We provide the pseudo-codes of the generalizations of Algorithms 3 and 4 in Algorithms A1 and A2. The pseudo-codes of the generalizations of Algorithms 1 and 2 are left for the readers to deduce.

Algorithm A1 Pseudo-code of the default version of carousel greedy that we recommend for generalized feature selection.

Input I, k

Output R

```

1:  $S \leftarrow$  the solution produced by generalized forward stepwise regression  $\triangleright$  In the order
   of selection
2:  $R \leftarrow S$ 
3:  $LastImpro = 0$ 
4:  $RECRSS =$  RSS of  $R$ 
5: while  $LastImpro < k$  do
6:    $LastImpro = LastImprove + 1$ 
7:   Remove 1 variable from the tail of  $R$ 
8:   Add 1 variable to the head of  $R$  according to generalized forward selection
9:   if RSS of  $R < RECRSS$  then
10:     $LastImpro = 0$ 
11:   end if
12: end while
13: return  $R$ 

```

In Algorithm A2, to apply generalized forward selection, best interchange MIP (BI-MIP) in Algorithm 4 should be replaced by generalized best interchange MIP (GBI-MIP) in Algorithm A2. Also, forward stepwise regression is replaced by generalized forward stepwise regression in step 1. The other parts of Algorithm 4 and Algorithm A2 are the same. GBI-MIP generalizes BI-MIP by introducing two additional constraints, one for Case A and the other for Case B, which means we can also incorporate both cases. The formulation of GBI-MIP is provided below:

$$\text{GBI-MIP: } \min \sum_{i=1}^n \left(\sum_{j=1}^p X_{ij} \theta_j - y_i \right)^2, \quad (\text{A1})$$

$$\text{subject to } -Mz_i \leq \theta_i \leq Mz_i, \quad \forall i \in [p], \quad (\text{A2})$$

$$\sum_{i \in HC_j} z_i \leq 1, \quad j = 1, \dots, m, \quad (\text{A3})$$

$$l_i z_i \leq \theta_i \leq u_i z_i, \quad \forall i \in [p], \quad (\text{A4})$$

$$z_i \leq w_i, \quad \forall i \in S, \quad (\text{A5})$$

$$\sum_{i \in S^c} z_i \leq 1, \quad (\text{A6})$$

$$\sum_{i \in S} w_i \leq |S| - 1, \quad (\text{A7})$$

$$\theta_i \in \mathbb{R}, \quad \forall i \in [p], \quad (\text{A8})$$

$$z_i \in \{0, 1\}, \quad \forall i \in [p], \quad (\text{A9})$$

$$w_i \in \{0, 1\}, \quad \forall i \in S. \quad (\text{A10})$$

Here (A3) and (A4) are the two additional constraints for Case A and Case B, respectively. $HC_j, j = 1, \dots, m$ are the sets of highly correlated variables. To exclude Case A, we can let each HC_j include only one variable. That is, each variable is only highly correlated with itself, and no other variables are highly correlated with it. To exclude Case B, we can let $l_i = -M, u_i = M$ for any $i \in [p]$ in (A4).

Algorithm A2 Pseudo-code of finding an FSI(1) local minimum for generalized feature selection by Gurobi.

```

1: Initialize  $|S| = k$  by generalized forward stepwise regression with coefficients  $\theta$ 
2: while TRUE do
3:    $S' \leftarrow$  Apply GBI-MIP to  $S$ 
4:   if RSS of  $S' \geq$  RSS of  $S$  then
5:     Break
6:   end if
7:    $S = S'$ 
8: end while
9: return  $S$ 

```

References

- Cerrone, C.; Cerulli, R.; Golden, B. Carousel greedy: A generalized greedy algorithm with applications in optimization. *Comput. Oper. Res.* **2017**, *85*, 97–112. [[CrossRef](#)]
- D'Ambrosio, C.; Laureana, F.; Raiconi, A.; Vitale, G. The knapsack problem with forfeit sets. *Comput. Oper. Res.* **2023**, *151*, 106093. [[CrossRef](#)]
- Capobianco, G.; D'Ambrosio, C.; Pavone, L.; Raiconi, A.; Vitale, G.; Sebastiano, F. A hybrid metaheuristic for the knapsack problem with forfeits. *Soft Comput.* **2022**, *26*, 749–762. [[CrossRef](#)]
- Cerulli, R.; D'Ambrosio, C.; Iossa, A.; Palmieri, F. Maximum network lifetime problem with time slots and coverage constraints: Heuristic approaches. *J. Supercomput.* **2022**, *78*, 1330–1355. [[CrossRef](#)]
- Cerrone, C.; Cerulli, R.; Sciomachen, A. Grocery distribution plans in urban networks with street crossing penalties. *Networks* **2021**, *78*, 248–263. [[CrossRef](#)]
- Shan, Y.; Kang, Q.; Xiao, R.; Chen, Y.; Kang, Y. An iterated carousel greedy algorithm for finding minimum positive influence dominating sets in social networks. *IEEE Trans. Comput. Soc. Syst.* **2021**, *9*, 830–838. [[CrossRef](#)]
- Cerulli, R.; D'Ambrosio, C.; Raiconi, A.; Vitale, G. The knapsack problem with forfeits. In *Combinatorial Optimization. ISCO 2020*; Lecture Notes in Computer Science; Baiou, M., Gendron, B., Günlük, O., Mahjoub, A.R., Eds.; Springer International Publishing: Cham, Switzerland, 2020; Volume 12176, pp. 263–272.
- Hammond, J.E.; Vernon, C.A.; Okeson, T.J.; Barrett, B.J.; Arce, S.; Newell, V.; Janson, J.; Franke, K.W.; Hedengren, J.D. Survey of 8 UAV set-covering algorithms for terrain photogrammetry. *Remote Sens.* **2020**, *12*, 2285. [[CrossRef](#)]

9. Carrabs, F.; Cerrone, C.; Cerulli, R.; Golden, B. An adaptive heuristic approach to compute upper and lower bounds for the close-enough traveling salesman problem. *INFORMS J. Comput.* **2020**, *32*, 1030–1048. [[CrossRef](#)]
10. Kong, H.; Kang, Q.; Li, W.; Liu, C.; Kang, Y.; He, H. A hybrid iterated carousel greedy algorithm for community detection in complex networks. *Phys. A Stat. Mech. Its Appl.* **2019**, *536*, 122124. [[CrossRef](#)]
11. Cerrone, C.; D'Ambrosio, C.; Raiconi, A. Heuristics for the strong generalized minimum label spanning tree problem. *Networks* **2019**, *74*, 148–160. [[CrossRef](#)]
12. Hadi, K.; Lasri, R.; El Abderrahmani, A. An efficient approach for sentiment analysis in a big data environment. *Int. J. Eng. Adv. Technol. (IJEAT)* **2019**, *8*, 263–266.
13. Cerrone, C.; Gentili, M.; D'Ambrosio, C.; Cerulli, R. An efficient and simple approach to solve a distribution problem. In *New Trends in Emerging Complex Real Life Problems*; ODS: Taormina, Italy, 2018; pp. 151–159.
14. Carrabs, F.; Cerrone, C.; D'Ambrosio, C.; Raiconi, A. Column generation embedding carousel greedy for the maximum network lifetime problem with interference constraints. In *Proceedings of the Optimization and Decision Science: Methodologies and Applications*; ODS, Sorrento, Italy, 4–7 September 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 151–159.
15. Akaike, H. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 199–213.
16. Mallows, C.L. Some comments on Cp. *Technometrics* **2000**, *42*, 87–94.
17. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* **1978**, *6*, 461–464. [[CrossRef](#)]
18. Foster, D.P.; George, E.I. The risk inflation criterion for multiple regression. *Ann. Stat.* **1994**, *22*, 1947–1975. [[CrossRef](#)]
19. Bertsimas, D.; King, A. OR forum—An algorithmic approach to linear regression. *Oper. Res.* **2016**, *64*, 2–16. [[CrossRef](#)]
20. Bertsimas, D.; King, A.; Mazumder, R. Best subset selection via a modern optimization lens. *Ann. Stat.* **2016**, *44*, 813–852. [[CrossRef](#)]
21. Zhu, J.; Wen, C.; Zhu, J.; Zhang, H.; Wang, X. A polynomial algorithm for best-subset selection problem. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 33117–33123. [[CrossRef](#)] [[PubMed](#)]
22. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B (Methodol.)* **1996**, *58*, 267–288. [[CrossRef](#)]
23. Zou, H. The adaptive lasso and its oracle properties. *J. Am. Stat. Assoc.* **2006**, *101*, 1418–1429. [[CrossRef](#)]
24. Hazimeh, H.; Mazumder, R. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Oper. Res.* **2020**, *68*, 1517–1537. [[CrossRef](#)]
25. Bertsimas, D.; Copenhaver, M.S.; Mazumder, R. The trimmed lasso: Sparsity and robustness. *arXiv* **2017**, arXiv:1708.04527.
26. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2005**, *67*, 301–320. [[CrossRef](#)]
27. Zhang, C.H. Nearly unbiased variable selection under minimax concave penalty. *Ann. Stat.* **2010**, *38*, 894–942. [[CrossRef](#)] [[PubMed](#)]
28. Bertsimas, D.; Van Parys, B. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *Ann. Stat.* **2020**, *48*, 300–323. [[CrossRef](#)]
29. Atamturk, A.; Gomez, A. Safe screening rules for L0-regression from perspective relaxations. In *Proceedings of the 37th International Conference on Machine Learning, Virtual Event, 13–18 July 2020*; Volume 119, pp. 421–430.
30. Moreira Costa, C.; Kreber, D.; Schmidt, M. An alternating method for cardinality-constrained optimization: A computational study for the best subset selection and sparse portfolio problems. *INFORMS J. Comput.* **2022**, *34*, 2968–2988. [[CrossRef](#)]
31. Mazumder, R.; Friedman, J.H.; Hastie, T. SparseNet: Coordinate descent with nonconvex penalties. *J. Am. Stat. Assoc.* **2011**, *106*, 1125–1138. [[CrossRef](#)] [[PubMed](#)]
32. Hastie, T.; Tibshirani, R.; Tibshirani, R. Best subset, forward stepwise or lasso? Analysis and recommendations based on extensive comparisons. *Stat. Sci.* **2020**, *35*, 579–592. [[CrossRef](#)]
33. Meinshausen, N. Relaxed lasso. *Comput. Stat. Data Anal.* **2007**, *52*, 374–393. [[CrossRef](#)]
34. Mannor, S.; Meir, R.; Zhang, T. Greedy algorithms for classification—consistency, convergence rates, and adaptivity. *J. Mach. Learn. Res.* **2003**, *4*, 713–742.
35. Tewari, A.; Ravikumar, P.; Dhillon, I.S. Greedy algorithms for structurally constrained high dimensional problems. In *Proceedings of the the 24th International Conference on Neural Information Processing Systems, Granada, Spain, 12–15 December 2011*; Curran Associates Inc.: Red Hook, NY, USA, 2011; pp. 882–890.
36. Barron, A.R.; Cohen, A.; Dahmen, W.; DeVore, R.A. Approximation and learning by greedy algorithms. *Ann. Stat.* **2008**, *36*, 64–94. [[CrossRef](#)]
37. Painter-Wakefield, C.; Parr, R. Greedy algorithms for sparse reinforcement learning. In *Proceedings of the the 29th International Conference on International Conference on Machine Learning, Edinburgh, UK, 26 June–1 July 2012*; pp. 867–874.
38. Shafique, K.; Shah, M. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 51–65. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.