BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 05/06 pp208–224
Volume 3, Number 3, September 2020
DOI: 10.26599/BDMA.2020.9020005

Novel and Efficient Randomized Algorithms for Feature Selection

Zigeng Wang, Xia Xiao, and Sanguthevar Rajasekaran*

Abstract: Feature selection is a crucial problem in efficient machine learning, and it also greatly contributes to the explainability of machine-driven decisions. Methods, like decision trees and Least Absolute Shrinkage and Selection Operator (LASSO), can select features during training. However, these embedded approaches can only be applied to a small subset of machine learning models. Wrapper based methods can select features independently from machine learning models but they often suffer from a high computational cost. To enhance their efficiency, many randomized algorithms have been designed. In this paper, we propose automatic breadth searching and attention searching adjustment approaches to further speedup randomized wrapper based feature selection. We conduct theoretical computational complexity analysis and further explain our algorithms' generic parallelizability. We conduct experiments on both synthetic and real datasets with different machine learning base models. Results show that, compared with existing approaches, our proposed techniques can locate a more meaningful set of features with a high efficiency.

Key words: feature selection; randomized algorithms; efficient selection

1 Introduction

In the big data world, we see an explosion in data volume, data diversity, and data dimensions. High-dimensional data bring a lot of difficulties to machine learning. They require not only a large amount of computational resources and storage space, but also more training data for the machine learning models to learn well. Feature selection can reduce data dimensions by selecting a subset of important features. With a concise subset of features, machine learning models' learning and prediction efficiency can be greatly improved, and the models' explainability can be greatly enhanced.

High-dimensional feature space is a common complication in different scientific applications. For example, in modern bioinformatics and genetics, a conventional DNA sequence representation is through counting the occurrences of length-K sub-strings called K-mers[2, 3]. Typically, K is chosen to be larger than 20, so that the number of possible K-mers can be greater than 10^{12}. As another example, in smart city travel demand applications, a large number of different vehicle records, weather conditions, and geographic metrics are taken as features in the prediction models with feature dimensions greater than 200 million[4, 5]. Moreover, the data generated in the smart city context are usually gathered in a heterogeneous and streaming fashion[6–8]. A majority of these applications are time-sensitive (e.g., smart and connected vehicles) which need real-time or near-real-time data analysis[9]. As a result, an effective and efficient feature selection algorithm design, nowadays, becomes even more important and necessary.

Feature selection is the process of selecting a subset of the most relevant features from a given set of features

• Zigeng Wang, Xia Xiao, and Sanguthevar Rajasekaran are with the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, USA. E-mail: {zigeng.wang, xia.xiao, sanguthevar. rajasekaran}@uconn.edu.

† A preliminary version of this paper was published at 2019 IEEE 21th International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)[1].

* To whom correspondence should be addressed.
Manuscript received: 2020-04-15; revised: 2020-05-06; accepted: 2020-05-07

in a group of data examples by discarding irrelevant, redundant, and noisy features, either based on or not based on the corresponding labels of the examples. Feature selection is different from feature extraction. Feature extraction constructs a new feature space by projecting the original high-dimensional feature space to a low-dimensional space. For feature selection, the new feature set is a directly selected subset of the original feature set, while the new feature space of feature extraction is transformed from the original space through linear or nonlinear combinations. Examples of feature extraction techniques include PCA[10] and LDA[11]. Although feature selection and feature extraction reduce the data dimension from two different perspectives, both methods share the same potential advantages of increasing model computational efficiency, reducing storage space, improving the learning accuracy, etc. Due to the advantage of high explainability, feature selection has been employed in many different applications, ranging from bioinformatics[12], text mining[13] to image processing[14], ecological modeling[15], and industrial fault diagnosis[16]. Li et al.[17] and Jović et al.[18] gave comprehensive summaries of feature selection applications.

Feature selection methods can be classified into three main categories: filter methods, wrapper methods, and embedded methods. Filter methods select features without actually training a machine learning model, in which features are ranked with respect to certain statistical criteria, such as Pearson's correlation and mutual information with dependent variables. Some novel graph based approaches[19,20] are proposed. These approaches construct a feature interaction graph with each node representing each feature and the weights on the edges representing the relations between each pair of features. Then, clustering algorithms are designed and applied to group important feature sets. Filter methods can usually select features efficiently[21], but the selected features can be with low quality for a given prediction problem[22], which is mainly due to the fact that the features are selected without actually training a prediction model. Different from filter methods, wrapper methods select features in a supervised way, in which the important feature subsets are pinpointed based on the predicting performance given the true labels and learning models. In wrapper methods, different optimization algorithms have been used to locate the optimal feature set that minimizes the predicting error. But wrapper methods can be time consuming since the wrapper methods treat the learner as a black box and evaluate the predicting performance of the well-trained learner iteratively with different potentially meaningful feature subsets. The randomized feature selection algorithms proposed in this paper fall under the wrapper category, which aim to accelerate the selection process. Details on different wrapper methods are provided in Section 2. Besides filter and wrapper methods, embedded methods perform feature selection in a hybrid way in which the important features are being located at the same time the learner is being trained. Decision tree and Least Absolute Shrinkage and Selection Operator (LASSO) based models are typical examples. But this feature selection ability can only be embedded in a small number of machine learning models, while this may be difficult for popular non-parametric models such as KNNs[23]. Moreover, conventional training of linear models with l_1 normalization should be with the whole feature set, which can be memory intensive when the data dimension is very high[24]. A general summary of the pros and cons of the three classes of feature selection methods is shown in Table 1. Besides selecting concise features to improve prediction performance, there are also many novel discussions on the fairness issues[25,26] and the privacy issues[27,28] involving in the whole feature selection process.

In different research domains, many kinds of non-parametric and linear models, which are not embedded with feature selection capability, have been shown to have great prediction advantages. For example, in the material science domain, Gaussian process regressor has been used in polymer discovery and takes the lead in dielectric and bandgap predictions given the multi-scale material descriptors as features[29,30]. For another example, in bioinformatics, KNNs classifier has been extensively applied to the metagenomic classification problem, in which short DNA/RNA subsequences are used as features[2,3]. The number of different combinations of these short subsequences can be exponential in the length of the sequence(s). More specifically, in the smart city context, a unified linear regression model has been applied to predict unit original taxi demands on temporal, spatial, event, and multiple combinational features. Results show that this linear model outperforms popular non-linear

Table 1 Pros and cons of different feature selection methods.

Category	Pros	Cons
Filter	Computationally efficient	Low accuracy
Wrapper	High accuracy	Computationally expensive
Embedded	Low extra cost	Learner dependent

models on large scale datasets from an industrial online taxicab platform[4]. For all the applications above, the data dimensions can be from 100+ to 1 000 000+, and selecting important feature subsets becomes an urgent requirement. In view of the above discussion, we observe that wrapper feature selection methods are a suitable/ideal approach for many applications.

Traditional wrapper based feature selection methods rely on conventional searching algorithms such as branch and bound[31,32] and sequential search[33,34]. But those schemes usually cannot overcome local optima or features' dependency. In order to select a better subset of features, randomized optimization approaches have been extensively used[35,36]. These algorithms can locate a better feature set with a higher computational cost. As a result, lightweight randomized selection algorithms have recently received a growing attention. Randomized feature elimination and selection algorithms[37–39] have been designed to locate important feature subsets effectively.

In this paper, we propose efficient randomized feature selection algorithms incorporating three novel techniques to further accelerate and enhance the feature selection process. The first *semi-randomized selection* technique speeds up feature discovery by dynamically controlling the feature candidate generation breadth through evaluating the quality of the already discovered feature subset in each iteration. The second *warm start* technique initializes the feature selection with a comparatively meaningful warm feature subset and quickly refines the feature subset with a predefined learner based on a fine-grained feature selection stage estimation. The third *cool down* technique introduces cool down factors to describe the degree of potential of each feature based on feature candidate evaluation history. Searching attention is optimized and concentrated on feature candidates with higher accuracy improvement potentials. We conduct extensive experiments to show that, compared to existing models, the proposed techniques can select important feature sets more effectively and efficiently.

Our paper has the following contributions: (1) The three techniques we propose are highly generic and can be used in different randomized selection algorithms with arbitrary machine learning models; (2) our algorithms are naturally parallelizable and work-optimal; (3) our algorithms are very memory efficient where the memory usage is independent from the total number of features; and (4) we offer a theoretical time complexity analysis and a convergence proof for our feature selection algorithms.

The rest of this paper is organized as follows: Related works are summarized in Section 2. In Section 3, we present our randomized feature selection speedup techniques. Section 4 provides the theoretical analysis of the algorithm. The experimental evaluation and comparison are in Section 5. Section 6 concludes this paper.

2 Related Work

In this section, we first summarize the feature selection strategy of wrapper methods. Then, we discuss two classes of wrapper methods, deterministic wrapper algorithms, and randomized wrapper algorithms, respectively.

2.1 Wrapper feature selection strategy

Wrapper methods select features by iteratively generating and evaluating the predictive performance of feature set candidates. Figure 1 illustrates a generic wrapper feature selection method, which includes two main components: (1) Search and generate a candidate feature subset; (2) evaluate the candidate set's prediction performance. These two components are repeated iteratively until a predefined stopping condition is satisfied.

As shown above, the wrapper methods first generate candidate subsets by using different searching strategies based on the given data. A candidate subset then goes to a predefined learner for performance evaluation and a certain quantitative metric (e.g., accuracy or Root Mean Square Error (RMSE)) is usually computed based on an objective function*. The evaluation metric is then

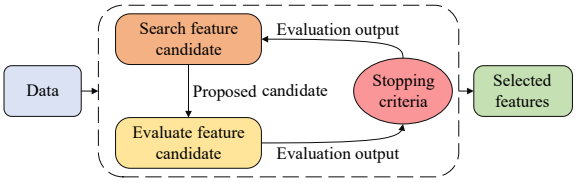


Fig. 1 A generic wrapper based feature selection framework.

*For a given learning algorithm, there are usually two main steps to finalize the model. The first step is model training, where the learner is trained based on the training data. And the second step is model testing, where the trained learner is tested based on a separate set of testing data. People always use the test result to evaluate the *true* prediction performance of the constructed model. Similarly, in wrapper methods, the testing results are usually used as the evaluation metrics for feature candidate performance comparison.

compared with the previous results, if the candidate subset yields a better value, the previously stored best feature subset and its solution are usually replaced with the current ones. Conventionally, based on the searching and evaluation history, new feature candidates are then generated and tested. This process iterates over the feature space until the stopping criteria is satisfied. The stopping criteria can be no performance improvement over a given number of iterations, the maximum number of iterations has been reached, etc. The important feature subset is finalized and output by incorporating prior knowledge.

2.2 Selection of candidate subsets

Selecting candidate feature subsets is a main part in wrapper methods. For a dataset with n different features, the number of different potential feature combinations can be exponential ($\Omega(2^n)$). Algorithms like exhaustive search may not be practical for large values of n. Due to this fact, different heuristic algorithms have been designed to locate a probabilistic optimal solution. With respect to searching strategies, wrapper methods can be categorized into two classes, deterministic feature selection and randomized feature selection.

2.2.1 Deterministic feature selection

Any deterministic feature selection method will employ deterministic algorithms to search feature subsets and generate deterministic outputs. Due to its ability to successfully solve discrete and combinatorial optimization problems, Branch and Bound (B&B) has also been used in feature selection[31,32]. The search starts with the whole feature set and it iteratively removes features until the error threshold is met. Then, the residual feature subset is pruned and refined through backtracking. In order to further improve the searching speed, Approximate Monotonicity with Branch and Bound (AMB&B)[40] was proposed, in which non-monotonic evaluation functions can be used with a relaxed error threshold. Although B&B based methods empirically run much faster than exhaustive search, these approaches can still take exponential times in the worst case. They also assume monotonicity of the features with respect to the evaluation functions.

Sequential feature selection is another main deterministic feature selection category. Traditional Sequential Forward Search (SFS) and Sequential Backward Search (SBS) follow greedy hill climbing strategies which slightly change feature subsets by adding/deleting features in each iteration[34]. Based on SFS and SBS, a sequential floating search algorithm[41] and its variations have been proposed. In sequential floating search, the algorithm adjusts the trade-off between forward and backward steps dynamically and allows a "self-controlled backtracking" to moderate the features' nonmonotonicity effect. *Replacing-the-weak-feature* strategy[33] was designed to broaden the sequential forward floating search's scope by checking whether removing any feature in the current feature subset and adding a new one can improve the accuracy. Sequential selection based approaches can generate solutions in asymptotically linear time, but these approaches usually suffer from local optima in the optimization process and also assume feature monotonicity.

2.2.2 Randomized feature selection

To overcome getting stuck in local optima and accelerate feature selection, different randomized approaches have been proposed.

One group of these randomized algorithms employ or combine existing conventional randomized optimization algorithms[42], such as evolutionary algorithms[35], genetic algorithms[36], and simulated annealing[43], in which feature selection is directly treated as a 0-1 integer programming problem. Random feature subsets are generated and modified iteratively based on the embedded randomized optimization algorithms to maximize the prediction performance of a selected feature subset. Compared to the deterministic feature selection, these randomized approaches, relying on powerful well-developed optimization algorithms, can locate a better feature subset without getting trapped in local optima. But this type of randomized approaches is heuristic in nature. There are no theoretical guidelines on their effectiveness and efficiency[35], which makes the selection of the most suitable algorithm difficult. Also for some light-weight applications, because of the high computational cost of the complicated optimization schemes being used, the above approaches may not be perfectly suitable. As a result, some concise randomized feature selection approaches have attracted a lot of research interest.

In 2004, Stracuzzi and Utgoff[37] proposed a Randomized Variable (feature) Elimination (RVE) algorithm. The RVE algorithm conducts feature searching backwards by iteratively eliminating one or more random features. The search was conducted along a narrow trajectory through a sparsely sampled feature

space, by which a smaller number of subsets need to be evaluated and a significant amount of computation power can be saved. RVE comes with a convergence proof and it works effectively in practice, especially when the features are redundant or irrelevant, but when the features are with high dimension, RVE can use a lot of memory.

In 2015, Saha et al.[38] proposed a Randomized Feature Selection (RFS) algorithm. As shown in Fig. 2, the RFS starts with a random feature subset and generates a feature candidate by looking at its neighboring feature subsets, either through adding and/or deleting one random feature. A decision on whether to update the starting feature set or not will be made based on the neighbor's prediction performance. The RFS algorithm selects features following a finer granularity with single feature operations and provides a solid convergence proof by modeling the feature selection process as a homogeneous Markov chain. Different from RVE, RFS has a high memory efficiency since its feature candidate selection is conducted by evaluating only a subset of the features and it is suitable for high

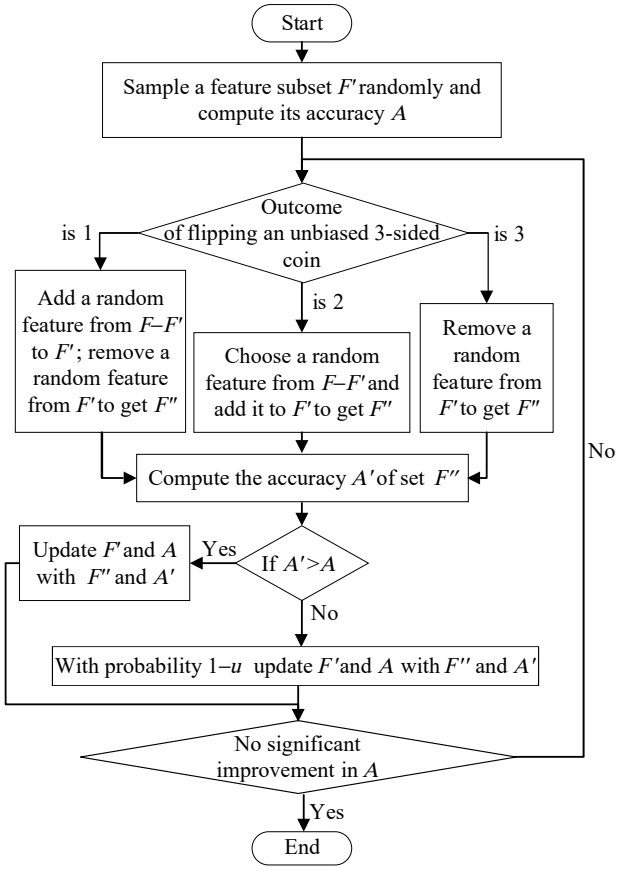


Fig. 2 Flowchart of randomized feature selection algorithm[38].

dimensional data. But there are still many opportunities to further improve the models' performance. In each iteration, the RFS model refines the selected feature set by only comparing the accuracy with one candidate feature set, which can slow down the search process by escalating on adding and deleting less important features. The escalation can be very severe when the selected feature set is premature since the initial feature set is randomly selected at the beginning of the search in RFS. Another opportunity to accelerate the feature selection process is by leveraging all the previous evaluation history and properly adjusting the searching attention. Motivated by these, we propose our semi-randomized feature selection algorithms as the following and we are mainly comparing our approaches with RFS.

3 Our Algorithm

Inspired by the state-of-the-art randomized approaches in wrapper based feature selection algorithms[37, 38], we design novel techniques to accelerate existing randomized feature selection algorithms and to locate a more meaningful subset of features. The techniques are summarized as follows:

• **Semi-randomized selection** which increases the exploration breadth in every search iteration automatically according to the search stage estimation. By evaluating more than one candidate feature set, semi-randomized selection selects candidates based on their grouped potential contributions to the accuracy. In order to construct a better subset of features and to speedup the convergence, neighboring feature combinations with the current feature set are ranked based on the stage estimation.

• **Warm start** which initializes the feature selection with a comparatively more meaningful warm feature subset instead of directly conducting search from a pure random feature subset (e.g., as in RFS[38]) to avoid the escalations caused by frequently adding and deleting less important features into/from the selected feature set. The warm start technique is also jointly developed with feature elimination, which reduces the initializing cost especially for the high dimensional data sets with high feature dependency.

• **Cool down** which assigns a cool-down factor to each feature to describe their temporary potentials to improve the accuracy based on the previous performance evaluations. With the cool-down factor, our algorithm can adjust the searching attention effectively to the

features with high probability to improve accuracy, so that computational cost is optimized in such a way that candidate feature evaluations which consume the most running time are reduced.

In the following subsections, we detail our new feature selection enhancement techniques. We further discuss their potential for feature selection speed and prediction accuracy improvement.

3.1 Semi-randomized selection

In order to search for a meaningful feature set, a nice balance between the greediness and randomness in the searching process always has to be carefully considered. As shown in Fig. 2, the RFS algorithm replaces a temporally selected feature set F' with a feature candidate subset F'' if F'' improves the prediction performance, no matter how tiny the improvement is, so that features which are not very meaningful can also be selected. In this way, the *pure-randomized* selection mechanism in RFS, which adds/deletes one single feature at a time can potentially increase the time complexity of the selection algorithm. Such irrelevant features can dramatically slow down the selection process, especially when there is randomness in the learner or there is noise in the training set. At the initialization stage of feature selection, the *pure-randomized* approach can be even more harmful as we discuss in detail in Section 3.2.

To pinpoint a feature subset which is more informational, we propose a new Semi-Randomized Feature Selection (Semi-RFS) approach. The main idea of our algorithm is that, in each step, Semi-RFS samples more than one feature and picks the local-optimal to be the candidate set F''. Comparing to the *pure-randomized* feature, our feature set candidate has a higher probability of contributing to a larger increase in the prediction accuracy. In this way, a feature subset can be constructed in a more concise and efficient way.

With respect to application requirements, we customize two different variations of the Semi-RFS approach, namely, static Semi-RFS and adaptive Semi-RFS, respectively. The main difference between the two algorithms is on the choice of the feature group size. A pseudo-code of Semi-RFS is shown in Algorithm 1. For static Semi-RFS, features are selected from random feature samples (F_g and $F_{g'}$) of a static size. g and g' are the predefined sizes for feature groups sampled from the residual feature set $F - F'$ and currently selected feature set F', respectively. In contrast the adaptive Semi-RFS

Algorithm 1 Semi-RFS

Input: F' (an initial feature subset), F (the entire feature set), and L (a predefined learner)

```
Compute accuracy A evaluated with learner L on feature set F';
repeat
    *For adaptive Semi-RFS: compute feature group sizes g and g';
    Evaluate feature candidates by adding and/or removing each single feature from the feature group(s) F_g and/or F_{g'};
    Identify the accuracy A' of the locally best feature candidate F'';
    if A' > A then
        Update A and F' with A' and F'', then start a new search from F';
    else
        With probability 1 − u, update A and F' and start a new search from F';
    end
until A predefined stopping condition is satisfied;
Return F';
```

selects features from feature groups of adaptive sizes based on the feature selection status. The group size is computed based on a specially designed adaptive function F_{semi}. This function works as follows. When the feature selection begins, the selected feature set F' is far from optimal, so that weak residual features are more likely to be selected and added to F'. In order to avoid the frequent interactions with the weak features, the feature group size should be comparatively large and the searching should be conducted in a broader and greedier manner. When the feature selection is nearing an end, a concise feature group can be more helpful, because the searching randomness introduced by a small feature group can result in a potentially further descend. A theoretical analysis of the Semi-RFS is given in Section 4.1.

In adaptive Semi-RFS, we design a function F_{semi} for computing the feature group size adaptively based on estimating the searching progress. When the feature selection initially begins, a generated feature candidate set may more likely have a better accuracy since the initial feature set could be far from optimal. There can be many consecutive iterations with no better sets discovered near the final stage of the selection. This could happen if the feature set F' is close to optimal and it is difficult to locate a candidate set with a better accuracy. Thus, the group size can be computed based

on estimating of the searching stage. We discovered that the number of consecutive iterations with no new better feature sets being found would be a good metric for the searching stage estimation. We use N_{ni} to denote this metric and the subscript ni stands for no accuracy improvement. Here, N_{ni} with a small value indicates the feature selection is starting, while a larger N_{ni} indicates the search is approaching an end. We formulate feature group size computation functions as Eq. (1). In Eq. (1), we use generalized sigmoid functions to calculate the percentage of the features in the residual feature set and the current feature set to be sampled. There are two hyper-parameters in the equations, α and β, where β is for controlling the initial group size and α is for balancing the influence of the stage estimation of metric N_{ni}. In Section 5.2, we conduct an experimental study that reveals that the hyper-parameters β and α are not sensitive.

$$\begin{aligned} g &= |F - F'| \cdot \frac{1}{\beta + \mathrm{e}^{\alpha \cdot N_{\mathrm{ni}}}}; \\ g' &= |F'| \cdot \frac{1}{\beta + \mathrm{e}^{\alpha \cdot N_{\mathrm{ni}}}} \end{aligned} \tag{1}$$

In Eq. (1), the searching stage metric N_{ni} can drop dramatically to 0 once a better feature set is located. In order to maintain the stability of N_{ni} and the group sizes, we leverage ideas from adaptive filters and calculate N_{ni} with adaptive averaging.

$$N_{\mathrm{ni}}(k) = \begin{cases} 0, & k = 0; \\ w \cdot n_{\mathrm{ni}}(k) + (1-w) \cdot N_{\mathrm{ni}}(k-1), & k > 0 \end{cases} \tag{2}$$

In Eq. (2), the lowercase $n_{\mathrm{ni}}(k)$ denotes the current exact number of consecutive iterations with no accuracy improvement at the k-th iteration. The capitalized $N_{\mathrm{ni}}(k)$ is the metric we used in Eq. (1) and it is calculated with weighted averaging of the current exact $n_{\mathrm{ni}}(k)$ and its previous value $N_{\mathrm{ni}}(k-1)$. The adaptive averaging will enhance the robustness of the feature group size calculation. For simplicity, we omit the iteration index k in N_{ni}.

3.2 Warm start

In feature selection, the initial feature set largely influences the search efficiency. In RFS[38], the selection starts from a random feature subset, in which there can be many irrelevant features, so that an arbitrary neighbor of the initial feature set can be selected with a high probability based on a tiny contribution of prediction accuracy. In the initialization stage, with a tiny accuracy improvement, feature set F' will be rapidly updated. If we formulate it as an pure optimization problem, because of the plateau, the initial point can be far from optimal, which makes the whole optimization process hard to converge. As a result, locating a warmly initialized preliminary feature subset F_{pre} can speedup the whole optimization process.

A warmly initialized preliminary feature subset carries general information to describe the data distributions for discrimination. The initial feature set F_{pre} can be generated from other low weight feature selection algorithms, or from domain experts and numerical simulations. But, a good starting feature set may not perform perfectly for the predefined learner due to its specific learning strategy and learnability. Thus, a good transition is of great importance, in which the preliminary initial feature would be quickly tuned with the predefined learner for a better accuracy. So here, we design a refined adaptive Semi-RFS to speedup the transition from the preliminary features in which we refine the feature group size computation in Eq. (3) with a finer granularity.

$$\begin{aligned} g &= |F - F'| \cdot \frac{I_{\mathrm{warm}}}{\beta \cdot I_{\mathrm{warm}} + \mathrm{e}^{\alpha \cdot N_{\mathrm{ni}}}}; \\ g' &= |F'| \cdot \frac{I_{\mathrm{warm}}}{\beta \cdot I_{\mathrm{warm}} + \mathrm{e}^{\alpha \cdot N_{\mathrm{ni}}}} \end{aligned} \tag{3}$$

In contrast to Eq. (1), a warm start index I_{warm} is introduced for a more accurate searching stage prediction. During the initialization, N_{ni} can be close to zero for multiple steps, so that using N_{ni} solely cannot accurately describe the search status at the transition time. Thus, we involve a parameter $\mathrm{PCT}_{\mathrm{imp}}$ for stage estimation. $\mathrm{PCT}_{\mathrm{imp}}$ is the percentage of potential feature set candidates that can contribute to the prediction accuracy with the features drawn from the sampled feature group. A larger $\mathrm{PCT}_{\mathrm{imp}}$ indicates the selection process is far from completion.

$$I_{\mathrm{warm}}(k) = \begin{cases} 1, & k = 0; \\ \dfrac{1}{1 - \mathrm{PCT}_{\mathrm{imp}}(k)}, & k > 0 \end{cases} \tag{4}$$

We formulate the positive correlation between I_{warm} and $\mathrm{PCT}_{\mathrm{imp}}$ in Eq. (4). For $\mathrm{PCT}_{\mathrm{imp}}$, we also employ adaptive averaging shown as the following, where lowercase $\mathrm{pct}_{\mathrm{imp}}(k)$ denotes the percentage of better candidates at the k-th iteration,

$$\mathrm{PCT}_{\mathrm{imp}}(k) = \begin{cases} 0, & k = 0; \\ \mathrm{pct}_{\mathrm{imp}}(k-1), & k = 1; \\ (1-w') \cdot \mathrm{PCT}_{\mathrm{imp}}(k-1) + & \\ \quad w' \cdot \mathrm{pct}_{\mathrm{imp}}(k-1), & k > 1 \end{cases} \tag{5}$$

We describe the two-step warm start feature selection process in Algorithm 2. The algorithm starts from a preliminary feature set and the feature set will be fine-tuned with the warm start adaptive process.

Warm start can work with a preliminary feature set and also it can enhance feature elimination algorithms such as RVE[37]. In feature elimination, the algorithm begins with the entire feature set. If there are many irrelevant or redundant features, the feature elimination process will face the same plateau problem in the initialization stage of feature selection. As a result, feature elimination algorithms can also leverage the warm start and follow the Semi-RFS mechanism as shown in Algorithm 3. In order to select a concise set of features, we penalize the feature addition operation by introducing biased coin tosses, so that the features will have a higher probability of being removed instead of being added. Please note that, due to the nature of evaluating from the whole feature set, feature elimination algorithms can suffer from high memory usage when the total feature dimension is very high.

3.3 Cool down

Throughout the feature selection process, the efforts spent on searching and evaluating irrelevant/redundant features cannot contribute to the prediction accuracy. Thus, locating the temporarily weak features and focusing search attention on high potential features can largely contribute to the selection efficiency. In previous works[37,38], features from the residual set $F - F'$ are randomly selected with equal probability, and then added into the current feature set F', while the features in F' are also to be removed with the same probability. This memory-less searching strategy does not leverage any previous feature candidate evaluation results, which brings redundancies to duplicate candidate evaluations.

Algorithm 2 Warm start initialization

Input: F_{pre} (a preliminary feature subset), F (the entire feature set), and L (learner)
Output: A subset F' of features ($F' \subset F$)
begin
  Compute the accuracy A evaluated by learner L with feature subset F_{pre};
  Set F_{warm} to override the existing group size function F_g and $F_{g'}$ in Semi-RFS;
  Call Semi-RFS with overrided F_{warm}, and with variables F_{pre}, F, and L;
  Return the return value F' of Semi-RFS;
end

Algorithm 3 Warm start elimination

Input: F (the entire feature set) and L (learner)
Output: A subset F' of features ($F' \subset F$)
begin
  Set Preliminary feature set F_{pre} with F;
  Compute the accuracy A evaluated by learner L with feature subset F_{pre};
  Set F_{warm} to override the existing group size function F_g and $F_{g'}$ in Semi-RFS;
  Call Semi-RFS with overridden F_{warm} and feature addition penalty, and with variables F_{pre}, F, and L;
  Return the return value F' of Semi-RFS;
end

In feature addition operation, new feature candidates are constructed by picking feature f from $F - F'$ purely randomly and letting the merge of f and F' to be F''. If F'' does not achieve a better accuracy than the previous feature set F', in the next iteration, however, feature f can still be selected with an equal probability with all the other features in the residual feature set $F - F'$. But in the previous round of evaluation, feature f has been proved to be a temporarily weak feature and it is redundant to be selected in the exact adjacent round and to be re-evaluated. As a result, a lot of computational power is wasted. We design a cool down technique which generates feature candidates with higher potentials by fully utilizing history evaluations.

In each addition and deletion operation, instead of sampling features equally, we assign a large cool-down factor to the weak features based on historic evaluations to decrease their possibility of being sampled for evaluation. For an addition operation, if a picked feature f from the residual feature set has been proved to be temporarily irrelevant when the feature candidate brings a big accuracy drop, we will assign a large value to the cool-down factor f. The same holds for removing features from the current feature set F'. If a feature f' in F' has been evaluated to be temporarily relevant (the removal of f' brings a large performance drop), a large cool down value will be assigned to f', so that the removal probability of feature f' can be reduced. We formulate the relation between the probability of being chosen and the cool-down factors in Eq. (6).

$$P(i) = \begin{cases} \dfrac{1/\text{fac}(i)}{\sum_{i=1, f_i \in F'}^{|F|} 1/\text{fac}(i)}, & \text{if } f_i \in F'; \\ \dfrac{1/\text{fac}(i)}{\sum_{i=1, f_i \in F-F'}^{|F|} 1/\text{fac}(i)}, & \text{if } f_i \in F - F' \end{cases} \tag{6}$$

In Eq. (6), f_i denotes the i-th feature in the entire

feature set, $P(i)$ is f_i's probability of being chosen, and $\text{fac}(i)$ is the cool-down factor of feature f_i. The larger the cool-down factor, the lower the probability will be that a feature will be chosen in local selection.

During the entire feature selection process, F' can change dramatically from its being initialized to being finalized. And the temporary relevance of a feature in the residual feature set $F-F'$ and F' may not be permanent. So, the cool-down factor assigned to each feature should be gradually warmed up and the features will have gently increasing probabilities of being sampled. To summarize, we also enclose a detailed illustration of Semi-RFS with cool down in Algorithm 4. For a better illustration of the cool down approach, we explain this idea together with Semi-RFS.

4 Analysis

4.1 Convergence proof for Semi-RFS

We model Semi-RFS in Section 3.1 as a walk in a complete directed graph $G(V, E)$. We use V to denote the set of nodes and E to denote the set of edges. Each node v in V represents a distinct combination of features. There can be 2^n distinct nodes when there are in total n features. Given a node v, we define its arbitrary neighbor as v' which has an edge directly connecting to it. Semi-RFS algorithm begins with a node v and moves to its neighbors iteratively. In Semi-RFS, the

Algorithm 4 Semi-randomized feature selection with cool down (detailed)

Input: An initial feature subset F' ($F' \subset F$), a complete feature set F, a predefined learner L, a stopping condition C_{stop}, and a group size computation function A_{semi}
Output: A subset of selected features F'

begin
  Compute the accuracy A evaluated with learner L on feature subset F';
  Initialize the cool-down factors facs for all the features in the currently selected feature set F' with value $\sqrt{|F'|}$ and features in the residual feature set $F-F'$ with $\sqrt{|F-F'|}$;
  repeat
    Compute feature candidate group sizes g and g' based on Eq. (1);
    Compute the selection probabilities $P_{F'}$ and $P_{F-F'}$ for features in F' and $F-F'$ based on Eq. (6);
    Uniformly randomly pick one element from the sets 1, 2, and 3;
    if *the picked element equals* 1 **then**
      Pick a set F_g of g random features from $F-F'$ with $P_{F-F'}$, evaluate the performance of each combination $F'+f$, $\forall f \in F_g$ with learner L, and add the best of F_g [†] to F' to get F'' ;
      Pick a set $F_{g'}$ of g' random features from F' with $P_{F'}$ and remove the weakest of $F_{g'}$ from F'' to get new F'' and get the new accuracy A';
    else if *the picked element equals* 2 **then**
      Pick a set F_g of g random features from $F-F'$ with $P_{F-F'}$, choose the best of F_g with L and add to F' to get F'', and compute the new accuracy A';
    else
      Pick a set $F_{g'}$ of g' random features from F' with $P_{F'}$ and remove the weakest of $F_{g'}$ with L to get F'' and compute the new accuracy A';
    end
    Decrement all the fac whose values are greater than 1 by 1 ;
    Update fac for the features in g and/or g' in previous evaluation ;
    if $A' > A$ **then**
      Update A and F' with A' and F'' and perform a new search from F'';
    else
      With probability u, perform another search from F';
      With probability $1-u$, update A and F' with A' and F'' and perform search from F'';
    end
  until the predefined stopping condition C_{stop} is satisfied;
  Return F' ;
end

Note: [†]The best of F_g denotes the feature in F_g that can contribute the most to the accuracy after combining with F' but not the accuracy of the feature alone. The same applies for the weakest of F_g'.

move only depends on the current node, so that we model the process as a Markov chain. We further use $A(v)$ to denote the accuracy of the specific feature combination of the node v. Let $N(v)$ stand for the set of all neighbors of node v and let N denote the size of $N(v)$. For a fair theoretical comparison, we use the same state transition probabilities as in RFS[38], and we write the transition probability $P_{vv'}$ from node v to v' as

$$P_{vv'} = \begin{cases} 0, & v' \notin N(v); \\ \min(1, \exp(c \cdot (A(v') - A(v)))), & v' \in N(v) \end{cases} \tag{7}$$

where c is a constant. If we are at some node v in a given time step, it is not necessary to move to a different node in the next time step. We could remain in the same node v with a certain probability when no better neighbors can be found.

The main difference between the RFS and Semi-RFS is the selection of moves. RFS picks one neighbor randomly for evaluation and makes the move based on its accuracy, while Semi-RFS makes the move by evaluating a group of neighbors. We *rank* the neighbors of v in the following manner. We will set the rank of a neighbor v' to be 1 if v' provides the least accuracy improvement in the neighbor set $N(v)$; we will set the rank of v' to be 2 if it has the second least accuracy improvement; and so on. Please note that the accuracy improvement can be a negative number if $A(v')$ is smaller than $A(v)$. We formulate the relation among the rank of node v', the group size g, and neighbor size N. We utilize the sampling lemma from randomized sorting[44] in our analysis.

Lemma 1 Let a subset of neighbors of node v be denoted with $F_g(v) \subseteq N(v)$ and the size of the subset $|F_g(v)| = g$. For any neighboring node $v'_{\text{ar}} \in F_g(v)$, let its rank in $F_g(v)$ be j and let its rank in $N(v)$ be r_j. Then, j and r_j are related as follows:

$$\text{Prob.}\left(|r_j - j\frac{N}{g}| > \sqrt{4\alpha}\frac{N}{\sqrt{g}}\sqrt{\log N}\right) < N^{-\alpha} \tag{8}$$

where α is a constant. From the Lemma in Formula (8), rank r_j in $N(v)$ is tightly bounded with its rank j in feature subset F_g. In F_g, the neighbor v' with the highest accuracy will have rank g in F_g, so that by substituting j with g, we have

$$\text{Prob.}\left(N - r_g > \sqrt{4\alpha}\frac{N}{\sqrt{g}}\sqrt{\log N}\right) < N^{-\alpha} \tag{9}$$

In Formula (9), $N - r_g$ represents the number of neighbors that have a better accuracy than node v. When the group size g is reasonably small as $k \cdot \log N$, we have

$$\text{Prob.}\left(N - r_g > \frac{N}{\sqrt{k/4\alpha}}\right) < N^{-\alpha} \tag{10}$$

For static Semi-RFS, k is a constant and for adaptive Semi-RFS, k is a parameter characterized in Eq. (1). As a result, in Semi-RFS, in each iteration, the worst gradient according to Eq. (7) is

$$\Delta' = \min_{v' \in N(v), \text{ and } \text{rank}(v') \geqslant N - \frac{N}{\sqrt{k/4\alpha}}} A(v') - A(v) \tag{11}$$

with a high probability. While in RFS, the worst gradient is

$$\Delta = \min_{v' \in N(v)} A(v') - A(v) \tag{12}$$

Therefore, Semi-RFS can lead to a steeper descent.

We use the same assumption in Refs. [38, 45] that an algorithm is said to have converged if the underlying Markov chain had been in a globally optimal state at least once. We use D to denote the diameter of $G(V, E)$. Given n features, the expectation of D is $\Theta(n)$. A path must exist from the starting node v_{start} to the optimal node v_{opt} with length $\leqslant D$.

According to Eq. (7), node v will move to its neighboring node v' with a normalized probability $\frac{1}{N}P_{vv'} \geqslant \frac{1}{N}\exp(c\Delta')$. For an arbitrary starting node v_{start}, the expected path length to reach v_{opt} is

$$\text{E(steps)} \leqslant N^D \cdot \exp\left(-c\sum_{i=1}^{D}\Delta'_i\right) = [N \cdot \exp(-c\overline{\Delta'})]^D \tag{13}$$

It can be proved by induction easily that Semi-RFS converges within $\leqslant 2m[N \cdot \exp(-c\overline{\Delta'})]^D$ steps, with a probability of $\geqslant (1 - 2^{-m})$, for any integer m. We have a feature group size $k \cdot \log N$ and the number of feature candidate evaluations is $\leqslant 2m \cdot k \cdot \log N[N \cdot \exp(-c\overline{\Delta'})]^D$ with probability $\geqslant (1 - 2^{-m})$. Comparing to the number of total evaluations $2m[N \cdot \exp(-c\overline{\Delta})]^D$ in RFS[38], Semi-RFS can take a less number of steps when the following condition is satisfied:

$$\overline{\Delta'} - \overline{\Delta} \geqslant \frac{\log k + \log\log(N)}{c \cdot D} \tag{14}$$

Given $|F - F'| = n' \geqslant 0$, we have $|F'| = n - n' \geqslant 0$, then,

$$\begin{aligned} N &= |F - F'| + |F'| + |F - F'| \cdot |F'| = \\ &\quad n' + (n - n') + n' \cdot (n - n') = \\ &\quad n \cdot n' + n - n'^2 \leqslant n^2 \end{aligned} \tag{15}$$

Since $D = \Theta(n)$, the RHS of Formula (14) satisfies the following condition:

$$\frac{\log k + \log\log(N)}{c \cdot D} \leqslant \frac{\log k + \log\log(n^2)}{c' \cdot n} \tag{16}$$

where c' is another constant related to $D = \Theta(n)$, and

$$\lim_{n\to+\infty} \frac{\log k + \log\log(n^2)}{c' \cdot n} = \lim_{n\to+\infty} \frac{\log\log(n)}{c' \cdot n} = 0 \tag{17}$$

So that the condition in Formula (14) can easily be satisfied given a comparatively large n.

4.2 Parallelization of Semi-RFS

In this subsection, we show that the proposed Semi-RFS naturally supports parallelization. The efficiency of our proposed feature selection approach can be further improved with multi-processor machines.

We illustrate the parallelization potentials by employing the Parallel Random Access Machine (PRAM) as an example. PRAM is a collection of Random Access Machines (RAMs) working in synchrony and communication among the RAMs is relying on a common shared memory block[46]. Concurrent Read and Concurrent Write (CRCW) PRAM is a common type of PRAM, in which data can be read or written concurrently into the same unit in memory at the same time. In Algorithm 5, we show a parallel version of Semi-RFS.

In wrapper based feature selection approaches, feature candidate evaluations become the dominant part of overall computation, since in each iteration, learners will be trained from scratch and feature candidates are evaluated independently. Our model naturally supports parallel feature candidate evaluations. Each feature candidate in the candidate group can be assigned to one single processor for evaluation in parallel. Considering one single evaluation as the basic time unit, given a group size of $O(k \cdot \log N)$ in Section 4.1, one round of evaluations can be done in $O(1)$ time using $\log n$ CRCW PRAM processors with $N \leqslant n^2$. Locating the best feature candidate can be done in $O(\log\log\log n)$ time following the divide-and-conquer algorithm using $\log n$ CRCW PRAM processors[47]. As a result, the parallel Semi-RFS algorithm is asymptotically work optimal. Given $\log n$ CRCW PRAM processors, the sequential evaluation time can be reduced from $2m \cdot k \cdot \log N[N \cdot \exp(-c\overline{\Delta'})]^D$ to $2m \cdot [N \cdot \exp(-c\overline{\Delta'})]^D$.

Algorithm 5 Parallel Semi-RFS

Input: F' (a initial feature subset), F (the entire feature set), and L (a predefined learner)

1 Compute the accuracy A evaluated with learner L on feature subset F';
2 **repeat**
3 Compute feature group sizes g and g';
4 Compute A' for each generated feature candidate **in parallel**;
5 Locate the locally best feature candidate based on accuracy **in parallel**;
6 Update A and F';
7 **until** A predefined stopping condition is satisfied;
8 Return F' ;

5 Experimental Evaluation

In this section, we conduct an experimental study and evaluate the performance of our proposed feature selection algorithms. We first apply our three speedup approaches to NIPS 2003 feature selection challenge dataset[48] and cross-compare their performance with an existing randomized algorithm[38]. Then, we conduct an empirical study to investigate hyper-parameter sensitivity in adaptive Semi-RFS. We further apply our algorithms to solve real world problems, including polymer descriptor discovery for bandgap and dielectric constant prediction[30] and smart city restaurant revenue prediction[49], and compare the feature selection performance.

We conduct our experiments with a Dell Precision Workstation T7910. The workstation includes 256 GB RAM and two CPU sockets, containing 8 Dual Intel Xeon Processors E5-2667 (8C 16HT, 20 MB Cache, 3.2 GHz) each. Red Hat Enterprise Linux 7.0 operating system is running on the machine.

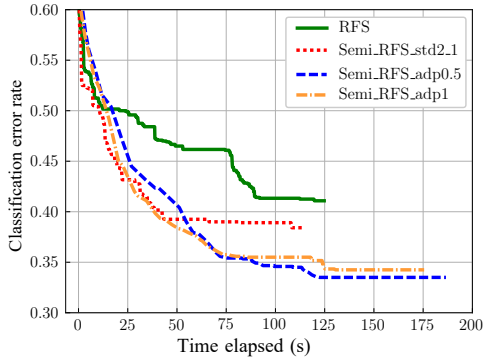
5.1 Synthetic data analysis

We evaluate our techniques on NIPS 2003 feature selection challenge dataset. The synthetic dataset is generated with Scikit-Learn Machine Learning Toolbox[50] following the rules in Ref. [48]. Specifically, *make_classification* function is used to introduce noise by the way of correlated, redundant, and uninformative features. We generate a dataset with 300 examples and 500 features. Among the 500 features, 30 of them are informative features and 4 out of the 30 informative features are redundant. The dataset has in total 3 classes with single cluster for each class. Gradient Boosting Tree is used as the default learner. Each experiment is run with a 5-fold cross-validation and we use the average results of 4 different runs for comparison.

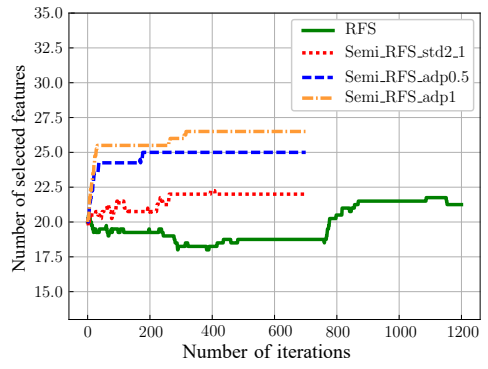
We first evaluate and compare Semi-RFS algorithms with RFS[38]. For static Semi-RFS, the static feature group size g from residual features is set to be 2. And for adaptive case, we set the hyper-parameter α to be 0.5 and 1, and we set β to be 5. A group size of 1 from

the currently selected feature set F' is set for all cases for a fair comparison. The initial feature set is with 20 randomly picked features. We use *Semi_RFS_std2_1* to denote the static Semi-RFS algorithm with residual group size 2 and selected feature group size 1, and use *Semi_RFS_adp0.5* and *Semi_RFS_adp1* to denote the two adaptive Semi-RFS algorithms with α to be 0.5 and 1, respectively. In contrast to RFS, Semi-RFS evaluates more than one feature candidate in each iteration. So, for a fair comparison, the elapsed time is used instead of the number of iterations for error rate convergence comparison. As we can see in Fig. 3a, the converge speeds of both static and adaptive Semi-RFS algorithms are much higher than that of RFS. More promisingly, based on search stage estimation, our adaptive Semi-RFS approach can adaptively control the feature group size for candidate selection (shown in Fig. 3c) and contribute to a faster convergence speed. From Fig. 3b, we can tell that Semi-RFS algorithms locate more informational features which potentially lead to a higher accuracy.

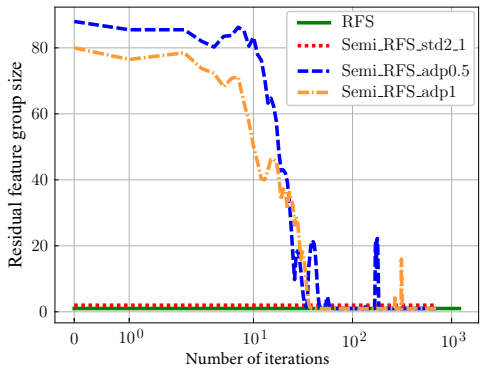
We then apply the warm start technique on both adaptive Semi-RFS and RFS algorithms, and cross compare the performance. In Fig. 4, we use *warm* to denote the algorithms initialized by warm start, such as *warm_RFS* and *warm_Semi_RFS_adp0.5*. The other naming conventions are following the legends in previous figures. In warm start, a preliminary feature set of size 20 is pre-generated with a Low-depth Extra



(a) Error rate

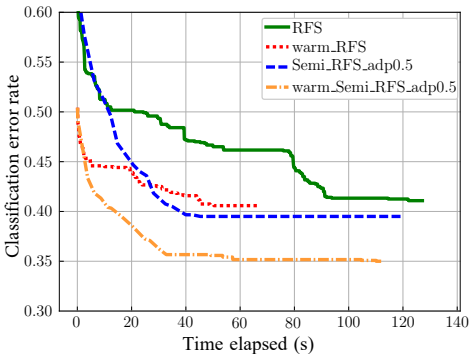


(b) Number of selected features

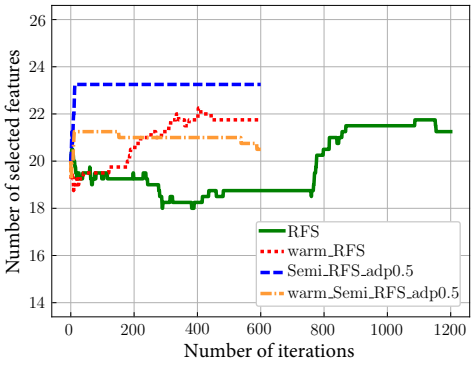


(c) Group size

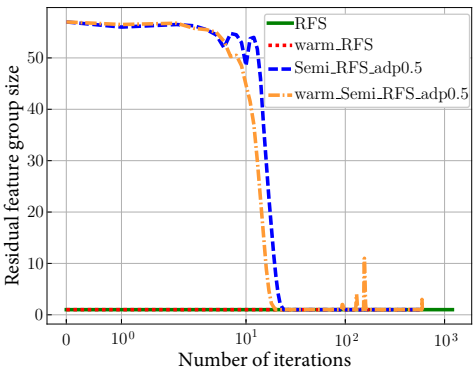
Fig. 3 Feature selection result comparison of Semi-RFS and RFS.



(a) Error rate



(b) Number of selected features

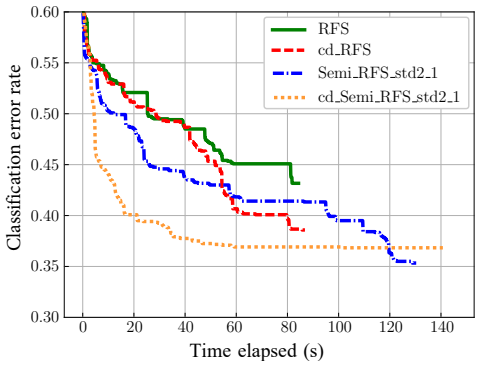


(c) Group size

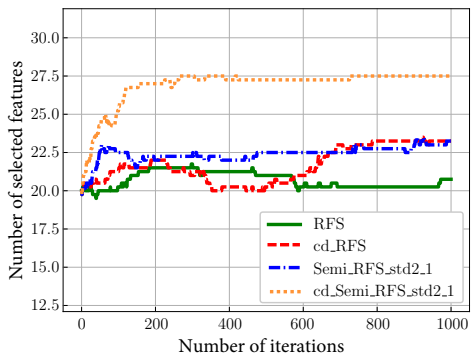
Fig. 4 Feature selection result comparison of warm start with Semi-RFS and RFS.

Trees Classifier. We first compare the convergence speed in Fig. 4a. The result shows that, in the initialization step, warm start warmly initializes the feature selection. With a finer feature group size estimation, warm start gives a smooth and fast transition from the preliminary feature set and accelerates the whole convergence speed of Semi-RFS. From Fig. 4c, we can infer that the warm start helps the group size shrink earlier and improves the total feature selection efficiency. For the selected feature size, we can see in Fig. 4b that warm start can quickly locate a more concise set of features with a comparatively higher accuracy and increase the model's generalizability.

We also combine our cool down approach with Semi-RFS and RFS algorithms and evaluate the performance. In Fig. 5, we use the prefix cd to denote feature selection schemes embedded with cool down. The other naming conventions in the legend are following the ones in all previous plots. Figure 5a shows that, for both RFS and Semi-RFS, the cool down technique can largely accelerate the convergence, in which feature candidates with high accuracy improvement potentials are reasonably selected and evaluated with higher probabilities. We can see from Fig. 5b, the cool down approach locates more meaningful features which contribute to a higher accuracy.



(a) Error rate



(b) Number of selected features

Fig. 5 Feature selection result comparison of cool down with Semi-RFS and RFS.

5.2 Hyper-parameter sensitivity study

In this subsection, we conduct an empirical study to evaluate the sensitivity of two hyper-parameters in adaptive Semi-RFS algorithm, namely α and β. As shown in Eq. (1), the hyper-parameter α helps to adjust the impact of search stage estimation on group size calculation and β generally helps to control the group size and balance the influence of stage estimation. In the experiment, we follow the same settings and notations as in Section 5.1. We use different combinations of hyper-parameters α and β and compare their convergence speed in terms of classification error.

In Fig. 6, the β values used are 1, 2, 4, and 8 for each sub-figure, respectively. Inside each sub-figure, we compare the convergence speed of RFS and Semi-RFS with α values of 0.5, 1, and 2. We can see that, when $\beta = 1$ or 2, different α values bring no significant difference to the convergence. When $\beta = 4$ or 8, the convergence becomes less stable, since the group size becomes smaller and more irrelevant features are involved due to an increase in randomness. As a result, the hyper-parameter α is not sensitive when β is relatively small. In general, assigning value 1 to both hyper-parameters can lead to a stable convergence.

5.3 Materials property prediction

In this section, we apply our proposed feature selection approach on real polymer datasets. In materials genomics research, it is a central problem to predict material properties through machine learning and computational methods. In this application, we apply our algorithms for polymer property prediction through materials fingerprinting. Fingerprinting is a crucial step of data-driven machine learning approach where the geometric and chemical information on the polymers are converted to a numerical representation. The entire dataset contains 242 features describing polymers from several hierarchical levels[30].

We cross compare our approach with RFS[38] and an SVM based Feature Elimination (FE) algorithm employed in Ref. [30]. We follow the same experiment setup and use the same Gaussian Process regressor[30] for polymer bandgap and dielectric constant predictions, respectively. We evaluate the results through a 5-fold cross validation and calculate the average values of 10 runs. We evaluate feature selection from different perspectives, including selected number of features,

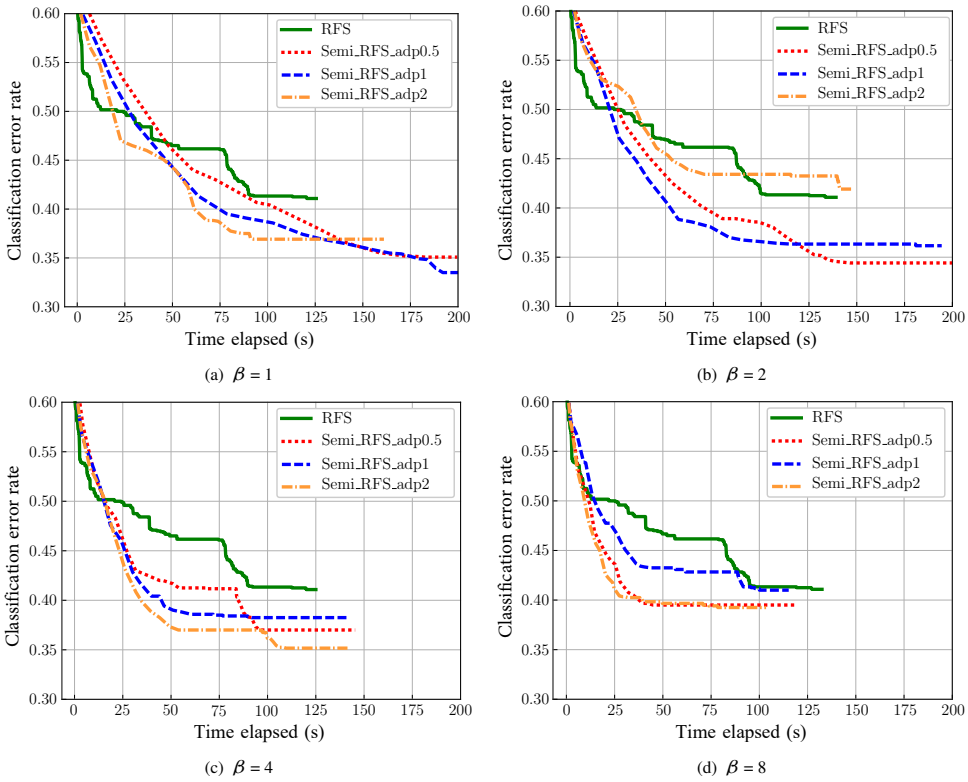


Fig. 6 Convergence rate comparison with different hyper-parameter values.

RMSE, and runtime. We compare our randomized approaches directly with the experimental results in Ref. [30].

As seen from Table 2, for both bandgap and dielectric predictions, randomized approaches can locate more concise feature sets. Our approach selects a feature set of size 33.3 which contains only 38% of the size of the final feature set discovered with the FE approach. Moreover, with less features, we achieve better prediction accuracy in both cases. Furthermore, our Semi-RFS approach converges twice as fast as the RFS algorithm. A concise material descriptor set will greatly enhance the feature explainability and it will provide more data evidence to material researchers for targeted experiments. Also, the highly efficient selection enables material investigations with larger data size and larger feature size.

5.4 Restaurant revenue prediction

Restaurant revenue prediction is an important index on analyzing customer demand and sales forecasting and thus plays an important role in smart city[51]. However, a large number of features require a high cost and a long time delay in the data collection phase. Introducing feature selection into restaurant revenue prediction can reduce the number of required features, and thus reduce the cost and improve the efficiency of the whole smart city network.

Restaurant revenue prediction dataset contains 137 restaurants from different locations from all over the world[49]. The features include the open date, location, city type of the restaurant, together with the demographic information and the commercial information nearby. Demographic information includes the age and gender distribution of the population and commercial data include nearby schools, banks, and commercial facilities.

The original dataset has 37 different features including categorical features and numerical features. We employ one-hot encoding on the categorical features such as location and city type, and expand the feature dimension to 108. In this problem, we use Gradient Boosting Tree

Table 2 Result comparison of material property prediction.

Algorithm	Number of features		RMSE		Runtime (s)	
	bandgap	dielectric	bandgap	dielectric	bandgap	dielectric
FE[30]	88	35	0.47	0.48	–	–
RFS[38]	33.1	28.4	0.466	0.471	2456	1942
Semi-RFS	33.3	26.7	0.464	0.472	1204	1015

regressor as the learner to evaluate our feature selection algorithm. Due to the limited number of data samples, we use a 4-fold cross-validation and the average results of 5 independent runs.

Table 3 summarizes the experimental comparison of RFS, Semi-RFS, and the base features. We compare the feature selection algorithms' performance with respect to different initial base feature sets of sizes $|F|$ (10, 30, 108) and their corresponding base RMSEs (2.64, 2.6, 2.62)$\times 10^6$. The initial features are selected according to the generated feature importance metric while training the gradient boosting tree regressor. Compared with the base model results, both RFS and Semi-RFS achieve better RMSEs and the feature search speed of Semi-RFS is faster than that of RFS. With different initial feature sizes, we can see that the RMSE increases when the initial feature size increases due to the ratio of the feature size and the number of training samples. This shows that our randomized approach is very suitable for identifying a smaller subset of features to reduce the error caused by model overfitting. From the feature selection process, we discover that the city of a restaurant and whether or not a restaurant supports drive through play an important role in the revenue prediction.

6 Conclusion

In this paper, we propose computational and memory efficient randomized feature selection algorithms based on semi-randomized selection, warm up, and cool down techniques. The techniques are highly generic and can be used in different randomized selection algorithms with arbitrary machine learning models. Our algorithms are naturally parallelizable and memory efficient. We conduct theoretical analysis and extensive experiments on different datasets with different learners, and our approaches show promising results.

Acknowledgment

This work was supported in part by the National Science Foundation (NSF) (Nos. 1447711, 1743418, and 1843025).

Table 3 Result comparison of restaurant revenue prediction.

Algorithm	$\|F\|$ base	$\|F\|$	RMSE base ($\times 10^6$)	RMSE ($\times 10^6$)	Time (s)
RFS	10	12	2.64	2.23	31.9
Semi-RFS	10	14.4	2.64	2.18	22.4
RFS	30	28.2	2.6	2.36	39.6
Semi-RFS	30	33.8	2.6	2.2	27.8
RFS	108	83.8	2.62	2.42	59.6
Semi-RFS	108	85.2	2.62	2.37	34.1

References

[1] Z. G. Wang and S. Rajasekaran, Efficient randomized feature selection algorithms, in *Proc. 2019 IEEE 21st Int. Conf. High Performance Computing and Communications*; *IEEE 17th Int. Conf. Smart City*; *IEEE 5th Int. Conf. on Data Science and Systems* (*HPCC/SmartCity/DSS*), Zhangjiajie, China, 2019, pp. 796–803.

[2] R. Ounit, S. Wanamaker, T. J. Close, and S. Lonardi, CLARK: Fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers, *BMC Genomics*, vol. 16. no.1, p. 236, 2015.

[3] P. Menzel, K. L. Ng, and A. Krogh, Fast and sensitive taxonomic classification for metagenomics with Kaiju, *Nat. Commun.*, vol. 7, p. 11 257, 2016.

[4] Y. X. Tong, Y. Q. Chen, Z. M. Zhou, L. Chen, J. Wang, Q. Yang, L. P. Ye and W. F. Lv, The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms, in *Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2017, pp. 1653–1662.

[5] C. F. Zhang, M. X. Dong, T. H. Luan, and K. Ota, Battery maintenance of pedelec sharing system: Big data based usage prediction and replenishment scheduling, *IEEE Trans. Network Sci. Eng.*, vol. 7, no. 1, pp. 127–138, 2019.

[6] H. Moeini, W. X. Zeng, I. L. Yen, and F. Bastani, Toward data discovery in dynamic Smart city applications, in *Proc. 2019 IEEE 21st Int. Conf. High Performance Computing and Communications*; *IEEE 17th Int. Conf. Smart City*; *IEEE 5th Int. Conf. Data Science and Systems* (*HPCC/SmartCity/DSS*), Zhangjiajie, China, 2019, pp. 2572–2579.

[7] X. R. Zhang, C. A. Huang, M. Y. Liu, A. Stefanopoulou, and T. Ersal, Predictive cruise control with private vehicle-to-vehicle communication for improving fuel consumption and emissions, *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 91–97, 2019.

[8] A. Gledson, T. B. Dhafari, N. Paton, and J. Keane, A smart city dashboard for combining and analysing multi-source data streams, in *Proc. 2018 IEEE 20th International Conference on High Performance Computing and Communications*; *IEEE 16th International Conference on Smart City*; *IEEE 4th International Conference on Data Science and Systems* (*HPCC/SmartCity/DSS*), Exeter, UK, 2018, pp. 1366–1373.

[9] M. Mohammadi and A. Al-Fuqaha, Enabling cognitive smart cities using big data and machine learning: Approaches and challenges, *IEEE Commun. Magaz.*, vol. 56, no. 2, pp. 94–101, 2018.

[10] S. Wold, K. Esbensen, and P. Geladi, Principal component analysis, *Chemometr. Intellig. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[11] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers, Fisher discriminant analysis with kernels, in *Proc. Neural Networks for Signal Processing IX*: *Proc. 1999 IEEE Signal Processing Society Workshop*, Madison, WI, USA, 1999, pp. 41–48.

[12] L. P. Wang, Y. L. Wang, and Q. Chang, Feature selection methods for big data bioinformatics: A survey from the search perspective, *Methods*, vol. 111, pp. 21–31, 2016.

[13] D. Mladenić, Feature selection in text mining, in *Encyclopedia of Machine Learning*, C. Sammut and G. J. Webb, eds. Boston, MA, USA: Springer, 2011, pp. 406–410.

[14] K. Huang and S. Aviyente, Wavelet feature selection for image classification, *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1709–1720, 2008.

[15] J. L. Tracy, A. Trabucco, A. M. Lawing, J. T. Giermakowski, M. Tchakerian, G. M. Drus, and R. D. Coulson, Random subset feature selection for ecological niche models of wildfire activity in Western North America, *Ecol. Modell.*, vol. 383, pp. 52–68, 2018.

[16] C. Liu, D. X. Jiang, and W. G. Yang, Global geometric similarity scheme for feature selection in fault diagnosis, *Exp. Syst. Applicat.*, vol. 41, no. 8, pp. 3585–3595, 2014.

[17] J. D. Li, K. W. Cheng, S. H. Wang, F. Morstatter, R. P. Trevino and J. L. Tang, Feature selection: A data perspective, *ACM Comput. Sur.*, vol. 50, no. 6, p. 94, 2017.

[18] A. Jović, K. Brkić, and N. Bogunović, A review of feature selection methods with applications, in *Proc. 2015 38th Int. Convention on Information and Communication Technology, Electronics and Microelectronics* (*MIPRO*), Opatija, Croatia, 2015, pp. 1200–1205.

[19] Z. H. Zhang and E. R. Hancock, A graph-based approach to feature selection, in *Proc. Int. Workshop on Graph-Based Representations in Pattern Recognition*, Münster, Germany, 2011, pp. 205–214.

[20] A. K. Das, S. Goswami, A. Chakrabarti, and B. Chakraborty, A new hybrid feature selection approach using feature association map for supervised and unsupervised classification, *Exp. Syst. Appl.*, vol. 88, pp. 81–94, 2017.

[21] P. Xiao, Z. G. Wang, and S. Rajasekaran, Novel speedup techniques for parallel singular value decomposition, in *Proc. 2018 IEEE 20th Int. Conf. High Performance Computing and Communications*; *IEEE 16th Int. Conf. Smart City*; *IEEE 4th Int. Conf. Data Science and Systems* (*HPCC/SmartCity/DSS*), Exeter, UK, 2018, pp. 188–195.

[22] G. Chandrashekar and F. Sahin, A survey on feature selection methods, *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.

[23] V. Fonti and E. Belitser, Feature Selection Using Lasso, https://pdfs.semanticscholar.org/24ac/d159910658223209433cf4cbe3414264de39.pdf, 2017.

[24] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, Efficient projections onto the l_1-ball for learning in high dimensions, in *Proc. 25th Int. Conf. Machine Learning*, Helsinki, Finland, 2008, pp. 272–279.

[25] N. Grgić-Hlača, M. B. Zafar, K. P. Gummadi, and A. Weller, Beyond distributive fairness in algorithmic decision making: Feature selection for procedurally fair learning, in *Proc. Thirty-Second AAAI Conf. on Artificial Intelligence* (*AAAI-18*)*, the 30th Innovative Applications of Artificial Intelligence* (*IAAI-18*)*, and the 8th AAAI Symp. on Educational Advances in Artificial Intelligence*, New Orleans, LA, USA, 2018.

[26] X. Zhang, M. M. Khalili, C. Tekin and M. Liu, Group retention when using machine learning in sequential decision making: The interplay between user dynamics and fairness, presented at the 33rd Conf. Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2019, pp. 15 243–15 252.

[27] T. Zhang, T. Q. Zhu, P. Xiong, H. Huo, Z. Tari, and W. L. Zhou, Correlated differential privacy: Feature selection in machine learning, *IEEE Trans. Ind. Inf.*, vol. 16, no. 3, pp. 2115–2124, 2019.

[28] X. R. Zhang, M. M. Khalili, and M. Y. Liu, Improving the privacy and accuracy of ADMM-based distributed algorithms, in *Proc. 35th Int. Conf. on Machine Learning*, Stockholm, Sweden, 2018, pp. 5796–5805.

[29] C. Li, D. R. De Celis Leal, S. Rana, S. Gupta, A. Sutti, S. Greenhill, T. Slezak, M. Height, and S. Venkatesh, Rapid Bayesian optimisation for synthesis of short polymer fiber materials, *Sci. Rep.*, vol. 7, p. 5683, 2017.

[30] C. Kim, A. Chandrasekaran, T. D. Huan, D. Das, and R. Ramprasad, Polymer genome: A data-powered polymer informatics platform for property predictions, *J. Phys. Chem. C*, vol. 122, no. 31, pp. 17 575–17 585, 2018.

[31] P. M. Narendra and K. Fukunaga, A branch and bound algorithm for feature subset selection, *IEEE Trans. Comput.*, vol. C-26, no. 9, pp. 917–922, 1977.

[32] J. Doak, An evaluation of feature selection methods and their application to computer security, Tech. Rep. CSE-92-18 UC Davis, Department of Computer Science, University of California, Davis, CA, USA, 1992.

[33] S. Nakariyakul and D. P. Casasent, An improvement on floating search algorithms for feature subset selection, *Patt. Recognit.*, vol. 42, no. 9, pp. 1932–1940, 2009.

[34] S. Russell and P. Norvig, *Artificial Intelligence*: *A Modern Approach*. New York, NY, USA: Prentice Hall, 2009.

[35] B. Xue, M. J. Zhang, W. N. Browne, and X. Yao, A survey on evolutionary computation approaches to feature selection, *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, 2016.

[36] P. Ghamisi and J. A. Benediktsson, Feature selection based on hybridization of genetic algorithm and particle swarm optimization, *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 2, pp. 309–313, 2015.

[37] D. J. Stracuzzi and P. E. Utgoff, Randomized variable elimination, *J. Mach. Learn. Res.*, vol. 5, pp. 1331–1362, 2004.

[38] S. Saha, S. Rajasekaran, and R. Ramprasad, Novel randomized feature selection algorithms, *Int. J. Found. Comput. Sci.*, vol. 26, no. 3, pp. 321–341, 2015.

[39] A. Brankovic, A. Falsone, M. Prandini, and L. Piroddi, A feature selection and classification algorithm based on randomized extraction of model populations, *IEEE Trans. Cybernet.*, vol. 48, no. 4, pp. 1151–1162, 2018.

[40] I. Foroutan and J. Sklansky, Feature selection for automatic classification of non-Gaussian data, *IEEE Trans. Syst. Man Cybernet.*, vol. 17, no. 2, pp. 187–198, 1987.

[41] P. Pudil, J. Novovičová, and J. Kittler, Floating search methods in feature selection, *Patt. Recognit. Lett.*, vol. 15, no. 11, pp. 1119–1125, 1994.

[42] D. J. Stracuzzi, Randomized feature selection, in *Computational Methods of Feature Selection*, H. Liu and H. Motoda, eds. London, UK: Chapman and Hall/CRC, 2007, pp. 53–74.

[43] S. W. Lin, Z. J. Lee, S. C. Chen, and T. Y. Tseng, Parameter determination of support vector machine and feature selection using simulated annealing approach, *Appl. Soft Comput.*, vol. 8, no. 4, pp. 1505–1512, 2008.

[44] S. Rajasekaran and J. H. Reif, Derivation of randomized sorting and selection algorithms, in *Parallel Algorithm Derivation and Program Transformation*, R. Paige, J. Reif, and R. Watcher, eds. Boston, MA, USA: Springer, 1993, pp. 187–205.

[45] S. Rajasekaran, On simulated annealing and nested annealing, *J. Glob. Optim.*, vol. 16, no. 1, pp. 43–56, 2000.

[46] J. Jájá, *An Introduction to Parallel Algorithms*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1992.

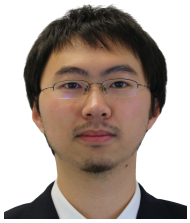
[47] E. Horowitz, S. Sahni, and S. Rajasekaran, *Computer Algorithms C++: C++ and Pseudocode Versions*. Stuttgart, Germany: Macmillan, 1997.

[48] I. Guyon, S. Gunn, A. B. Hur, and G. Dror, Design and analysis of the NIPS2003 challenge, in *Feature Extraction*, I. Guyon, M. Nikravesh, S. Gunn, and L. A. Zadeh, eds. Berlin, Germany: Springer, 2006, pp. 237–263.

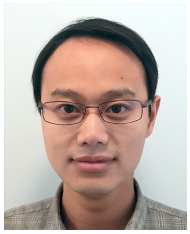
[49] T. F. Investment and Kaggle, Restaurant revenue prediction: Predict annual restaurant sales based on objective measurements, https://www.kaggle.com/c/restaurant-revenue-prediction, 2019.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[51] A. Lasek, N. Cercone, and J. Saunders, Smart Restaurants: Survey on customer demand and sales forecasting, in *Smart Cities and Homes*, M. S. Obaidat and O. Nicopolitidis, eds. Amsterdam, Netherlands: Elsevier, 2016, pp. 361–386.



Zigeng Wang received the BS degree from Xi'an Jiaotong University, China in 2013. He is currently pursuing the PhD degree under the guidance of Dr. Sanguthevar Rajasekaran at the Department of Computer Science and Engineering, University of Connecticut, USA. His research interests focus on designing efficient machine learning models, including deep learning model compression and acceleration, and feature selection. He also has research experiences in time series analysis, natural language processing, randomized and parallel algorithms, and networked system design.



Xia Xiao received the BS degree from Huazhong University of Science and Technology, China in 2011, and the MS degree in communication engineering from Chinese Academy of Sciences, China in 2014. He is currently pursuing the PhD degree in computer science and engineering at University of Connecticut, USA. His research interests include deep neural networks, efficient neuron architecture search, model compression, and network sparsification. He also conducts research in computer vison, such as object detection, image segmentation, and speedup techniques on computer vision model for real time inference systems.

Sanguthevar Rajasekaran received the MEng degree in automation from the Indian Institute of Science in 1983, and the PhD degree in computer science from Harvard University in 1988. Currently, he is the head of the Computer Science and Engineering (CSE) Department, board of trustees distinguished professor, and United Technologies Corporation (UTC) chair professor of CSE at University of Connecticut. Before joining University of Connecticut, he has served as a faculty member at Computer & Information Science & Engineering (CISE) Department of the University of Florida and Computer and Information Science (CIS) Department of University of Pennsylvania. During 2000–2002, he was the chief scientist for Arcot Systems. His research interests include big data, bioinformatics, algorithms, data mining, randomized computing, and HPC. He has published over 350 research articles in journals and conferences. He has co-authored two texts on algorithms and co-edited six books on algorithms and related topics. He has been awarded numerous research grants from such agencies as NSF, NIH, DARPA, Industry, and DHS (totaling around $20M). He is a fellow of the IEEE and the AAAS. He is also an elected member of the Connecticut Academy of Science and Engineering.