Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

ROBUST SCREENING AND PARTITIONING FOR FEATURE SELECTION: A BINARY SIMULATION OPTIMIZATION PROBLEM

Ethan Houser[1] and Sara Shashaani[1]

[1]Edward P. Fitts Dept. of Industrial & Systems Eng., North Carolina State University, Raleigh, NC, USA

ABSTRACT

Feature selection is the process of eliminating irrelevant or redundant covariates in a dataset to construct generalizable interpretable predictions. It is an NP-hard combinatorial problem with only heuristic or model-dependent greedy solution methods. To improve robustness, previous work proposed a zero-one simulation optimization by independently replicating model construction and out-of-sample error calculation. Genetic algorithms were used to solve feature selection showing improved robustness compared to existing techniques, albeit being slow, sensitive to hyperparameters, and lacking convergence guarantees. We propose a stochastic binary search method based on nested partitioning informed by an initial rapid screening phase. With new mechanisms for sampling, we propose steps to efficiently reduce the feature space and obtain an intelligent partitioning scheme. Our experiments show that our proposed algorithm finds fewer irrelevant features, is less sensitive to the data size, and is faster than competing methods.

1 INTRODUCTION

In machine learning, feature selection is an essential step that, especially in datasets with many features, can reduce model complexity and result in both robustness (generalization) and interpretability. The primary objective of feature selection is to identify a subset of features that are most relevant to the response variable to train an accurate prediction model. This task is difficult as all combinations of features provides a set of $2^{\#\text{features}} - 1$ many solutions to consider.

Feature selection (FS) in machine learning is done with various types of algorithms, including filter methods that choose a set of features prior to training prediction models, wrapper methods that wrap a search algorithm around the model construction, and embedded methods that use the training mechanism to only use a subset of features. Due to limited space, we refer to Kuhn and Johnson (2013) for details on FS where well-known methods such as random forests are reviewed. Importantly, most existing techniques consider the training dataset as the population. However, variability in the data can be significant in that the training set could retract us from finding the true contributors to the response. This means that existing FS methods can suffer from lack of robustness (Rong et al. 2019).

We treat the FS problem as a wrapper method to provide a framework that is flexible for any training approach including the popular deep learning models. In a wrapper method, the objective function is the mean prediction loss that is evaluated by running the prediction model on a test dataset. Uniquely, we view the problem as a simulation optimization (SO) where multiple bootstraps of data, and therefore multiple trained models, are used to evaluate each single subset of features. In this reformulation, each simulation replication starts from training a new model instance and computing its resulting loss on a held-out dataset. Shashaani and Vahdat (2023) called this class of algorithms SO-based FS (SOFS). The optimization involves a binary decision space that is high-dimensional.

Common wrapper methods, such as forward/backward selection or other greedy methods, sequentially add or remove features to find their importance. Other approaches such as LASSO deal with FS with a regularization that does not directly lead to finding the correct features, as highly correlated features could

be selected. We instead seek combinatorial optimization algorithms that can track large subsets of the feature space which greedy search methods would miss.

In the literature of SO, there are far fewer solvers that can effectively search through a large binary space. Ranking and selection algorithms including those variants run in parallel (Hunter and Nelson 2017) are not applicable for datasets of more than 20 features. Heuristics, such as genetic algorithms, are often used for combinatorial stochastic problems via sample average approximation to estimate the objective function. Genetic Algorithm (GA) based SOFS was developed to find robust feature subsets and proved to be competitive with existing approaches (Vahdat and Shashaani 2020; Houser et al. 2024). The drawback was a slower convergence in terms of simulation replications, as each iteration of GA's evaluated a sizable pool of solutions. Subsequent efforts to reduce computation time and correct bias in the estimated objective function value were helpful but did not drastically reduce computation time (Vahdat and Shashaani 2023).

This work dwells on a rapid screening approach (Tsai 2013) to reduce the feature space dimension before launching an adapted nested partitioning procedure (Shi and Ólafsson 2000) for the FS problem. Rapid screening has the ability to discard some redundant features or features that are obviously irrelevant. But the originally designed process does not sufficiently explore the feature space to excel at screening larger datasets. We therefore propose a hybrid neighborhood search procedure to better balance the trade-off between exploration and exploitation. Since screening only eliminates inferior feature subsets, we also design a mechanism after screening to categorize features as good, bad, or undecided. After substantially reducing the feature space and ranking undecided features, we move on to the nested partitioning where the reduced feasible subregions are iteratively investigated to focus on the *most promising region*. In both of these steps, we carefully prescribe budget allocation and sampling distributions that empirically prove successful for a range of datasets with varying sizes. Our experiments show a drastic decrease in computation time despite reporting competitive results. The promise of this strategy for FS is also the convergence theory that, while currently a work in progress, can inform an effective implementation of this algorithm. In the remainder of the paper, we formalize the problem and introduce notations used throughout the paper (Section 2), review the related work (Section 3), propose our new algorithm (Section 4), perform a numerical comparison study (Section 5), and summarize our findings (Section 6).

2 PROBLEM STATEMENT

Let a dataset $\mathcal{D}_0 = \{(x_{0i}, y_{0i})\}_{i=1}^{n_0}$ of n_0 data points consist of features $x_{0i} = (x_{0i}^1, x_{0i}^2, \cdots, x_{0i}^d)$—superscripts in this paper refer to components of a vector—with each feature $x^j \in \mathcal{X}_j$ (real, integer, categorical, etc), and response variable $y \in \mathbb{R}$. We assume that $y(x) = b(x)\beta + \epsilon$, where $b : \mathbb{R}^d \to \mathbb{R}^p$ is the basis function of features, β, is the unknown coefficient vector, and $\epsilon(x)$ is the random noise with mean 0 and a finite variance for all $x \in \mathbb{R}^d$. In a linear regression, $b(x) = (1, x^1, x^2, \cdots, x^d)$, but a generalized version can include polynomial, logarithmic, or other nonlinear bases that can also include interactions and weighted mechanisms (Kleijnen et al. 1985) that capture more of the complex relationship in the data. Forming a matrix B with rows $b(x_{0i})$ and setting $y_0 = (y_{01}, y_{02}, \cdots, y_{0n_0})$, we can estimate the regression coefficients with $\hat{\beta}(\mathcal{D}_0) = (B^\intercal B)^{-1}(B^\intercal y_0)$. The prediction model then becomes $\hat{y}(x; \mathcal{D}_0) = b(x)\hat{\beta}(\mathcal{D}_0)$ and the residual error for this model evaluated at points that were not involved in training the model, a.k.a., the test set $\mathcal{D}_1 = \{(x_{1i}, y_{1i})\}_{i=1}^{n_1}$, $\mathcal{D}_1 \cap \mathcal{D}_0 = \varnothing$ will provide the metric to measure the prediction model's accuracy, i.e., the loss function or the prediction error as $L := \frac{1}{n_1}\sum_{i=1}^{n_1}(y_{1i} - \hat{y}(x_{1i}; \mathcal{D}_0))^2$. The model trained on the data will only use features in the selected subset. Suppose $\theta \in \{0,1\}^d$ is a binary vector determining which features are selected for training with a value 1 and 0 otherwise. Then the model $y(x, \theta) = b(x, \theta)\beta + \epsilon$, takes as an argument this feature subset and adjusts the basis function to exclude features set to 0 in θ. The corresponding loss function $L(\theta)$ then measures the performance. Loss function can be further regularized as in LASSO or have other forms, making the methodology broad.

With this regression model defined, we propose a simulation-based machine learning approach to generate training and test data instances to validate regression models. In a SO view to FS, one loss value

from one training and test dataset is one realization of the expected loss—the objective function:

$$\min_{\theta\in\{0,1\}^d} f(\theta) := \mathbb{E}[L(\theta)]. \tag{1}$$

The expectation concerns all possible datasets that the true unknown data generating function would generate. Assuming the solution with the true features θ^* will yield the lowest expected loss in (1), the optimization problem seeks to choose an iterative path to θ^* by (i) generating new values of θ based on the performance of θ's observed in the previous iterations, and (ii) spending the total simulation budget, that is the total number of model constructions (which is the heavy lift of the replication), most efficiently to accelerate convergence.

The consideration of balancing exploration, exploitation, and estimation in our algorithmic design is vital. To estimate $f(\theta)$ for a given θ, N independent and identically distributed (i.i.d.) *replications* of training and test sets $(\mathcal{D}_{0\ell}, \mathcal{D}_{1\ell})_{\ell=1}^{N}$ yield a sample average approximation of its loss

$$\bar{L}_N(\theta) = \frac{1}{N}\sum_{\ell=1}^{N} L_\ell(\theta) := \frac{1}{N}\sum_{\ell=1}^{N}\left(\frac{1}{n_{1\ell}}\sum_{i=1}^{n_{1\ell}}(y_{1\ell i} - \hat{y}(x_{1\ell i}, \theta; \mathcal{D}_{0\ell}))^2\right).$$

Such i.i.d. replications can be generated from the empirical distribution of the dataset at hand, that is by sampling with replacement or bootstrapping. This view to a machine learning problem, although resembling the cross-validation and repeated bootstrap steps commonly used, differs from existing approaches in that the number of these replications is to be deliberated as it directly affects the result of the optimization. See Figure 1 for an example. Also note that one can leverage common random numbers by using a fixed collection of training and test sets when comparing different solutions.

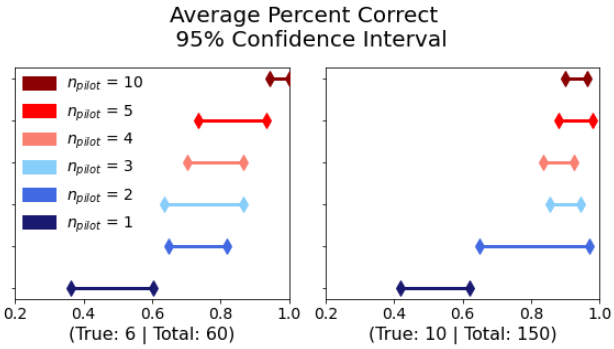

Figure 1: This figure captures the effect of an increased computational budget when solving the SO problem. As the initial budget is increased, the resulting average true positive rate improves remarkably.

It is noteworthy that resorting to a linear regression is to reduce the computational cost of repeated training of models. Although linearity is an oversimplification, since the goal of this study is not actually finding good predictions, we choose this restriction to gain insights into our proposed algorithms. The advantage of using regression is its low computational cost and, in various applications, its interpretability. Despite newer machine learning algorithms, still many applications such as those in healthcare engineering dominantly use regression models for their transparency (Alizadeh et al. 2024).

3 REVIEW OF RELATED WORKS

While many FS algorithms exist, we believe the SOFS view is not used in most schemes. We refer the interested reader for a more thorough literature on FS and competing methods to Shashaani and Vahdat (2023). Our previous work (Houser et al. 2024) has shown that in SOFS, incorporating data uncertainty when evaluating the effectiveness of a feature subset is crucial in finding the best features robustly. In this section, we focus on optimization algorithms most suitable for SOFS.

3.1 Genetic Algorithms for Simulation Optimization based Feature Selection

GA's are suitable for binary problems, due to their general formulation and bit-based definition of solutions. Its key idea is to preserve the "fittest" solutions and pass on their properties to the next population, leading to a gradual improvement over time. Each iteration uses four operations of evaluation, selection, crossover, and mutation with certain probabilities and rules. Sample average approximation can be used for a stochastic objective function to apply GA in a noisy setting. Despite sensitivity to its own parameters, GA has been widely implemented in applications. We achieved competitive results with SOFS at the cost of time. Adaptive and debiased evaluation steps were designed (Vahdat and Shashaani 2023) to incorporate input uncertainty and more carefully choose to replicate solutions when deemed helpful for the search.

Houser et al. (2024) also used a LASSO model in the GA's fitness function to refine the solutions generated in each iteration. In comparison, this overperformed support vector machine methods, random forest, and other decision tree techniques. However, the computational disadvantages of GA left the door open for future improvement. While evolutionary algorithms rely on the "best" performing solutions from previous iterations to guide the general direction of their search, the generation of new solutions in future generations remains more-or-less random and costly.

3.2 Nested Partitioning

Nested partitioning (NP) was initially developed as an SO variant of the partitioning methods such as branch and bound. The process involves dividing one subregion, known as the "most promising region" (MPR), in each iteration. The remaining subregions are merged into a single "surrounding region." Random solutions are sampled from each region, and their performance is estimated through simulation. The "promising index" is computed for each region based on the best estimated performance of its sampled solutions. The region with the lowest promising index becomes the new MPR. If the MPR consists of a single point, it cannot be further partitioned. If it is from the surrounding region, the method backtracks its steps to either its parent region or the entire feasible space. Otherwise, the MPR undergoes further partitioning. Shi and Ólafsson (2000) proved that their methods converges almost surely to the optimal solution. This method stems from the assumption that solutions located relatively close together have similar expected performances; NP was already used for a FS problem, however, it did not use the SOFS framework of replicating training to obtain independent random performances (Ólafsson and Yang 2005).

Ólafsson (2004) then expanded this method to use a ranking and selection to determine the best sampled solution from each region. Pichitlamken and Nelson (2003) present a nested partition method that also adds hill-climbing and restart steps and chooses the incumbent solution at each iteration differently. Xu and Nelson (2013) present and analyze a method that combines the branch-and-bound and nested partition approaches. A binary space could easily be managed with partitioning, making this class of algorithms most suitable for FS. The number of new partitions could change for a general case, but in a binary case the number of new partitions will always be two. In fact, Ólafsson and Yang (2005) did exactly that. However, that study was not scalable for high-dimensional feature spaces since too many partitions and possibly many backtracks are required. Partitioning can be viewed as a method for concentrating the search's sampling effort while backtracking broadens the sampling effort. Therefore, while backtracking allows for NP to correct potential mistakes, frequent corrections takes time and can hinder the search process.

We will be using a new variant of this method after a pre-processing step; see Section 4. To understand the details more clearly, we introduce some notation. Let Θ_k be the MPR to start iteration k. It gets partitioned following a partitioning rule to mutually exclusive subsets Θ_{k1} and Θ_{k2}. Let the surrounding region be $\Theta_{k3} = \Theta \backslash \Theta_k$. For each subregion $\Theta_{kj},\ j = 1, 2, 3$, then m many solutions are sampled uniformly, denoted by $\{\theta_{kjr},\ r = 1, 2, \cdots, m\}$. Ólafsson (2004) suggested sampling an initial number of solutions m_0 before choosing m total solutions with fixed-precision ranking and selection (Rinott's procedure). For each solution, N_{kjr} iid simulation runs estimate the objective function $\bar{L}_{N_{kjr}}(\theta_{kjr})$. The promising index function I_{kj} ranks the three regions, typically by the best performance in subregion j. If the subregion

with the best rank (lowest index) is the surrounding region, i.e., $\arg\min_{j=1,2,3} I_{kj} = 3$, then the algorithm backtracks by setting the next promising region as $\Theta_{k+1} = s(\Theta_k)$ where $s(\Theta_k)$ is the super-region of Θ_k or by setting it as the whole feasible region. Note, the latter makes a drastic change in sampling while the former does not. We then repeat until a decision has been made for every feature and the MPR contains a single point. NP is very sensitive to the partitioing scheme, and while Ólafsson and Yang (2005) used information gain metrics to facilitate this, the performance does not scale well with the problem size; see Section 5 for numerical experiments.

3.3 Rapid Screening

Rapid Screening (RS) is a highly adaptive method for evaluating solutions in zero-one SO problems. Through an adaptive neighborhood search, RS systematically samples new solutions from a neighborhood of top performing ones in the previous iteration while permanently eliminating inferior ones (Tsai 2013). The set of "surviving" solutions, denoted by V_t, are the top performing solutions kept by the algorithm. New solutions are then drawn from a defined neighborhood, following a distance sequence $\{\pi_t : t = 1, 2, \cdots, T\}$, containing all solutions that are at least π_t away from one of the "surviving" solutions, i.e.,

$$\mathcal{N}_{\pi_t}(V_t) := \left\{ \theta \in \{0,1\}^d : \bigcup_{\theta' \in V_t} \|\theta - \theta'\|_1 = \pi_t \right\}. \tag{2}$$

The 1-norm $\|\cdot\|_1$ is the sum of absolute differences between two vectors. The total number of solutions to compare in iteration t is fixed and can be specified depending on the size of the dataset, say, a_d. At the beginning of iteration t, V_t has fewer than a_d points as some where screened out in the previous iteration. Those points are replaced by the $m_t = a_d - |V_t|$ points $\{\theta_{t1}, \theta_{t2}, \cdots, \theta_{tm_t}\}$ uniformly selected from the neighborhood in (2) in the lead up to the screening procedure to form the set $\widetilde{V}_t = V_t \cup \{\theta_{t1}, \theta_{t2}, \cdots, \theta_{tm_t}\}$. To quickly screen inferior systems in this set, a fixed-budget ranking & selection procedure uses a fixed replication size of n_t (slowly growing with t) for each solution. The elimination set is therefore defined as

$$E_t = \left\{ \theta \in \widetilde{V}_t : \bar{L}_{n_t}(\theta') - \bar{L}_{n_t}(\theta) \leq -W_t(\theta, \theta') \text{ for some } \theta' \in \widetilde{V}_t, \theta' \neq \theta \right\}, \tag{3}$$

where $W_t(\theta, \theta') = t_{1-\alpha'/(a_d-1), n_t-1} \sqrt{\frac{\sum_{\ell=1}^{n_t} \left(L_\ell(\theta) - L_\ell(\theta') - (\bar{L}_{n_t}(\theta) - \bar{L}_{n_t}(\theta'))\right)^2}{(n_t-1)n_t}}$ is the half-width computed for each pair and the appropriate t-quantile with significance level α and $\alpha' = 1 - (1-\alpha)^{1/t}$; see Tsai (2013). The set of inferior solutions are those worse than their paired difference half-width with any other solution.

In the original algorithm, the distance sequence $\{\pi_t : t = 1, 2, \cdots, T\}$ can be decreasing, leading to a *shrinking neighborhood*, or a slowly increasing one, leading to a *nearest neighborhood*. In the latter, the slow increase is due to the requirement that all solutions in a neighborhood must be visited before moving to the next nearest neighborhood. Clearly, while the shrinking neighborhood emphasizes exploration, the nearest neighborhood aggressively exploits (locally searches). In summary, the nearest neighborhood search is most effective when working with small search spaces and when good solutions exist in the initial set of solutions. Otherwise, computational budget is spent on bad solutions early on given that the search method samples new solutions based on the surviving, yet bad, ones. This may take many more iterations to reach the optimal solution, or may not converge depending on the computational budget. The shrinking neighborhood avoids these issues by visiting a large and more diverse set of solutions in each iteration. However, the shrinking neighborhood less efficiently searches a promising region, assuming solutions close together in the solution space have similar expected performances which is the case for most SO problems.

Despite its use of ranking and selection, RS lacks global convergence guarantees and can be inefficient for larger datasets. Furthermore, when it comes to differentiating true features from redundant or noisy ones, RS typically struggles to identify the "harder-to-find" features (those with a measurably lesser relationship

to the response compared to informative features) despite efforts to efficiently and effectively explore the solution space. We will demonstrate these limitations in our numerical experiments.

3.4 Other Methods

Similar to GA, Simulated Annealing (Kirkpatrick et al. 1983) is another popular technique with mechanisms to escape local optima, often referred to as "hill climbing." The main idea is to generate solutions following a solution distribution and accept new solutions with a probability that depends on the performance compared to the incumbent (current optimal) solution and a *cooling schedule*. Simulation literature has more advances in convergent stochastic simulation annealing variants that use mechanisms to handle noisy function evaluations and its effect of the random solutions generated (Gutjahr and Pflug 1996; Alrefaei and Andradóttir 1999). For a high-dimensional binary space as in FS, however, the solution distributions (typically needing a transition probability matrix with certain properties such as irreducibility (Gelfand and Mitter 1989)) are hard to design. Stochastic ruler method (Yan and Mukai 1992) is another SO technique that, although different from simulated annealing in acceptance criteria, also needs a solution distribution and therefore, impractical for FS. Similar methods based on partitioning the feasible space, such as COMPASS (Hong and Nelson 2006) and BEESE (Andradóttir and Prudius 2009) entail global search steps that do not work well in high dimensions and their neighborhood structures in a binary space lose certain properties. Lastly, algorithms using pseudo-gradient-based steps in deterministic spaces such as R-SPLINE (Wang et al. 2013) do not apply in a binary space.

4 ROBUST SCREENING AND NESTED PARTITIONING FOR FEATURE SELECTION

Our proposed method, detailed in Algorithm 1, will combine a modified RS and NP sequentially. We modify the original procedure of RS for two uses: 1) dimension reduction, and 2) generation of an intelligent partitioning scheme. The first update to the algorithm relates to the search procedure and in the second update we implement a feature importance ranking scheme.

4.1 Obtaining Input Parameters

The methodology defined in Algorithm 1 requires several input parameters that should be carefully considered to efficiently obtain quality results. We refer the interested reader to Tsai (2013) for an in-depth discussion on selecting the inputs related to RS (i.e., n_{screen} initial screening replications, a_d screening solutions to evaluate, and γ replication growth rate.). The neighborhood distance sequence, $\{\pi_t\}_{t=1}^{T}$, should be determined based on the user's preference. The shrinking and nearest neighborhood search methods are outlined in Tsai (2013). For a hybrid neighborhood search, the decreasing sequence should be defined such that π_t begins no larger than the number of features in the dataset and resembles a decreasing function such as exponential decay. Section 4.3 details the acceptance and rejection of parameters, $p\%$ and $q\%$. Finally, the indifference zone δ, confidence level α, and pilot run size n_{pilot} can be set at the user's discretion.

4.2 Modified Neighborhood Search

Intelligent global search techniques emphasize exploration or exploitation. Exploration focuses on diligently searching the solution space to ensure a diverse and complete search is conducted. In the early stages of a search, exploration can be very important when little is known about the solution space. Solution exploration does not rely on obtaining a good set of solutions in early iterations to perform well. Rather, more sampling effort is spent identifying unique solutions that have not likely been visited. However, once good solutions are found, it is more difficult to find better solutions since the search remains so widespread.

Solution exploitation concentrates efforts based on knowledge gained throughout the search. Once good solutions have been identified, more sampling effort is placed on generating new solutions that are similar to the visited, good ones. Solution exploitation greatly relies on the solutions visited in early stages.

Algorithm 1 Robust Screening and Nested Partitioning for Feature Selection

Input: δ indifference zone, m number of solutions, n_{pilot} pilot run size, α confidence level, $p\%$ of accepted features, $q\%$ of rejected features, a_d screening solutions to evaluate, $\{\pi_t\}_{t=1}^{T}$ neighborhood distance sequence, n_{screen} initial screening replications, and γ replication growth rate.
Rapid Screening:
for $t = 1, 2, \cdots, T$ **do**
  Choose $m_t = a_d - |V_t|$ uniformly sampled new solutions from $\mathcal{N}_{\pi_t}(V_t)$ neighborhood to form $\widetilde{V}_t$.
  Run $n_t = \lceil n_{\text{screen}}\gamma^t \rceil$ iid simulations to compute $\bar{L}_{n_t}(\theta)$ for each $\theta \in \widetilde{V}_t$.
  Remove inferior solution in E_t defined in (3) and set $V_{t+1} = \widetilde{V}_t \backslash E_t$.
end for
Feature Importance Ranking:
Set $\bar{\theta} = \sum_{s=1}^{|V_T|} \theta_s$ (feature frequency in surviving solutions) and let $\rho(i)$ be a functional that returns the index of a feature (component in $\bar{\theta}$) with i-th most frequency (Section 4.3).
Nested Partitioning:
Set $\Theta_0 = \{\theta \in \{0,1\}^d : \ \theta^{\rho(1)} = \theta^{\rho(2)} = \cdots = \theta^{\rho(\lfloor dp \rfloor)} = 1, \ \& \ \theta^{\rho(\lceil dq \rceil + 1)} = \theta^{\rho(\lceil dq \rceil + 2)} = \cdots = \theta^{\rho(d)} = 0\}$ (Section 4.3).
Set $k = 1$ and $e = d - \lceil dq \rceil - \lfloor dp \rfloor$ as the number of undecided features.
while $k < e$ **do**
  Partition Θ_k into $\Theta_{k1} = \{\theta \in \Theta_k : \theta^{\rho(\lfloor dp \rfloor + k)} = 1\}, \Theta_{k2} = \{\theta \in \Theta_k : \theta^{\rho(\lfloor dp \rfloor + k)} = 0\}$.
  If $k > 0$, set the surrounding region as $\Theta_{k3} = \{0,1\}^d \backslash \Theta_k \cap \Theta_0$.
  for $j = 1, 2, 3$ **do**
    Obtain $\{\theta_{kj1}, \theta_{kj2}, \cdots, \theta_{kjm}\}$, m uniformly generated solutions from Θ_{kj}.
    for $r = 1, 2, \cdots, m$ **do**
      Run n_{pilot} iid simulations to compute $\hat{\sigma}^2_{kjr} = \frac{1}{n_{\text{pilot}}-1} \sum_{\ell=1}^{n_{\text{pilot}}} (L_\ell(\theta_{kjr}) - \bar{L}_{n_{\text{pilot}}}(\theta_{kjr}))^2$.
      Calculate $h(m, n_{\text{pilot}}, \alpha)$ according to Rinott's Procedure (Section 3.2)
      Set $N_{kjr} = \left\lceil \frac{h^2(m, n_{\text{pilot}}, \alpha)\hat{\sigma}^2_{kjr}}{\delta^2} \right\rceil$ and estimate $\bar{L}_{N_{kjr}}(\theta_{kjr})$ using $(N_{kjr} - n_{\text{pilot}})^+$ many iid simulations.
    end for
    Set the promising index of region j as $I_{kj} = \min_{r \in \{1,2,\cdots,m\}} \bar{L}_{N_{kjr}}(\theta_{kjr})$.
  end for
  Set $J_k = \arg\min_{j \in \{1,2,3\}} I_{kj}$
  Set $\Theta_{k+1} = \Theta_{kJ_k}$ if $J_k < 3$, and $\Theta_{k+1} = s(\Theta_k)$, i.e., Θ_k's super-region, otherwise. (Section 3.2)
  Increment $k = k + 1$.
end while

Simulation budget can be easily wasted if the initial set of solutions being evaluated is bad. Because the information gained from surviving solutions is exploited to generate new solutions, the rate at which this search converges is slow if low quality solutions are initially sampled.

The original RS emphasized exploration and exploitation by using either the shrinking or the nearest neighborhood procedure as described in Section 3.3. We propose a hybrid neighborhood search that spends early iterations exploring undiscovered subregions when little is known about the solution space before transitioning to an exploitation of promising regions in later screening iterations, i.e., $\pi_t = d(\eta_t t)^{-1}$ where η_t can be a step function (with small values early and large values in later iterations).

Furthermore, our adopted version of RS will only carry out the searching and screening phase of the original methodology, ignoring the stopping and selecting steps. The searching and screening phase will allow us to quickly gain information about the features based on the surviving solutions found at the end of a pre-defined number of screening iterations T. In our experiments we will demonstrate the effect of T in successfully screening a large number of inferior features.

4.3 Feature Ranking for Intelligent Partitioning

At termination of RS, we establish a feature importance ranking based on each feature's frequency of occurrence in the surviving solutions, which we track by summing all the surviving binary vectors, denoted by $\bar{\theta}$ in Algorithm 1. Based on this ranking, we can then classify features as: 1) accepted, 2) rejected, or 3) undecided. The top $p\%$ of features provide the most information about the response when chosen and are automatically accepted into the final solution. The bottom $q\%$ of features presume to have little to no information about the response and need no further consideration. Finally, features that rank between the $p\%$ and $q\%$ may or may not provide us valueable information and require further evaluation. These parameters can be chosen based on the aggressiveness of the FS (in forcing the number of selected features to be small) or it can be automated by checking the variability and other measures of features that appear in surviving solutions. This is currently an area of investigation. Once, the partitioning starts, the features that will partition the incumbent MPR will be sequentially picked from the ranked list.

5 NUMERICAL RESULTS

Experiments were conducted on our new algorithm, henceforth RSNP, to test its performance against the state of the art: 1) RS, 2) NP, and 3) GA used within SOFS. We explored various alternatives including search procedures, maximum number of iterations run, and termination criteria and performed *macroreplications* to show the variability of results from one solution method to another. The experiments involve synthetic datasets of increasing size and noise intensity in order to validate the success and scalability of each algorithm. In Section 5.1, we summarize the experiment settings necessary for an effective and efficient solution search. The dominating search procedure is then determined in Section 5.2. An implementation of this search procedure in RSNP is then compared to the other benchmark models in Section 5.3. Finally, the findings of the numerical analysis are summarized in Section 5.4.

5.1 Experiment Settings

We first tuned the number of screening iterations T. Larger T allows for a more comprehensive search, but also results in a higher computational cost. Ideally, the algorithm should perform as few screening iterations as possible while still conducting an effective solution search. Through experimentation, RSNP began to converge on a set of statistically indifferent solutions for $T \in [6, 10]$. The time at which convergence began usually increased with problem size d. A quick RSNP ($T \in [1, 5]$) was also evaluated to measure the algorithm's ability to separate true features from irrelevant ones. As expected, the total run time significantly reduced with the most benefit seen for the large dataset, and the algorithm was able to identify true features at a higher rate as T increased, moving from $\sim 75\%$ with $T = 3$ to $\sim 90\%$ with $T = 20$. Therefore, this increase provides a trade-off to consider: time vs. FS accuracy.

To further improve the computational efficiency of RSNP, termination criteria were implemented. It was determined that an adequate termination criteria to follow was to terminate after 5 consecutive screening iterations where we did not see a performance improvement greater than 3%. After implementing this new rule, the vast majority of macroreplications were able to terminate before running every screening iteration, thus reducing the overall time and computational effort required.

Finally, the set of solutions surviving the screening procedure can be converted into a feature importance ranking system. Ideally, all true features are ranked at the top of the list followed by the redundant and then the uninformative ones. The interpretation of this ranking system reveals which of the true features have a strong relationship to the response that can be quickly identified and which have a weaker relationship that is more difficult to distinguish. While we know the inherent relationships of the synthetic dataset, in reality the total number of true features is unknown. Therefore, it is important to set thresholds such that uninformative features are not accepted and true features that are difficult to identify are not rejected. We were able to comfortably set the acceptance threshold to $p = 5\%$ and the rejection threshold to $q = 70\%$.

5.2 Comparing Search Methods

Preliminary experiments determined that our hybrid procedure dominates the nearest procedure in every category of performance, excluding it from consideration. We will examine the performance of the hybrid and shrinking search procedures with respect to FS and computational efficiency.

Selection of Features Effective FS techniques can maximize the number of true features selected while minimizing the number of uninformative or redundant features selected. Figure 2 is a heat map of selected features across all experiment macroreplications run with the large dataset. A higher intensity of color is indicative of a feature being selected across multiple macroreplications. It is evident that the hybrid procedure identifies true features at a higher rate than the shrinking procedure, irrespective of the number of screening iterations. Ultimately, the hybrid procedure over 20 screening iterations leads to a more accurate identification of true features while the selection of uninformative features was competitive.

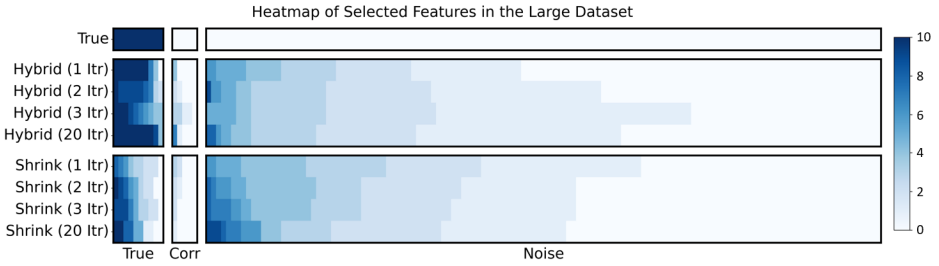

Figure 2: This heat map shows the frequency of selection for each feature where darker cells are indicative of a higher rate of selection across experiment macroreplications. For reference, the true model containing only informative features is shown in the top row.

Computational Requirements For small datasets, the difference in computational effort required is nominal between all models. For medium datasets, the shrinking procedure generally requires less of the computational budget than that of the hybrid procedure when comparing models run over the same number of iterations. The same is true when comparing models using the large dataset are run over a few screening iterations. However, as the number of screening iterations increases, the computational effort required by the shrinking procedure grows at a faster rate than that of the hybrid search procedure.

Summary In just a few screening iterations, the hybrid and shrinking procedures have similar run times. However, as more screening iterations are taken, the shrinking procedure's run time increases at a faster rate than the hybrid procedure. Meanwhile, the hybrid procedure dominates in terms of FS as the dataset grows and as the number of screening iterations increases. The hybrid procedure is more capable of identifying true features and discarding irrelevant ones compared to the shrinking procedure. Overall, the hybrid search procedure run over 20 screening iterations has the ability to effectively and efficiently identify only the true features compared to the shrinking and nearest search procedures.

5.3 Comparing to Benchmarks

In this section, we compare the hybrid search procedure run over 20 screening iterations to the original RS and NP algorithms along with the GA via SOFS (Houser et al. 2024) from our previous work. Experiments were conducted on datasets of increasing size and noise intensity to measure each model's robustness.

Selection of Features When comparing the ability to effectively select features, RSNP can correctly identify true features at a high rate of success that dominates NP and is competitive with GA and RS over varying screening lengths. However, RSNP can provide a higher level of interpretability in the final solution compared to RS and NP. As the number of features grows, the total number of features returned by both RS and NP grows at a staggering rate, as seen in Figure 3. This can be seen very clearly in Figure 4 where the majority of all features are being selected in the RS and NP cases. GA remains a competitive option, but RSNP achieves a higher true feature accuracy and a smaller selection of total features.

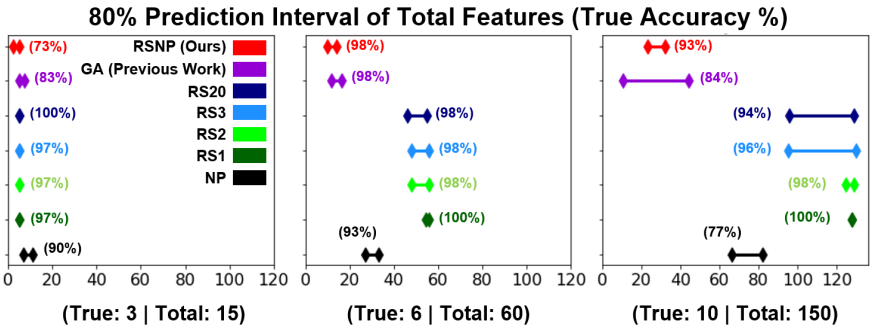

Figure 3: Overall, RSNP maintains a high true feature accuracy while keeping the total number of features relatively low. Only GA competes with RSNP in both categories as the dataset grows and the level of noise intensifies.

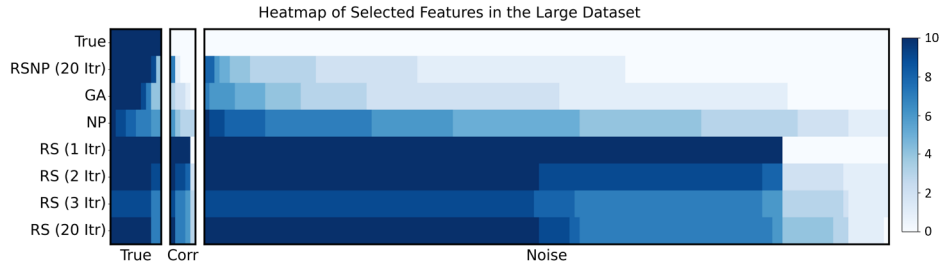

Figure 4: RSNP effectively identifies true features while ignoring correlated and noisy features. GA remains competitive but at the cost of an increased number of noisy features being selected. RS and NP accept a majority of features irrespective of their relationship to the response. For reference, the true model containing only informative features is shown in the top row.

Computational Requirements As seen in Figure 5, we outperform RS and are competitive with NP in terms of total time as the total number of features increases. As the number of features grows, the time it takes to run the RS procedure grows exponentially while the time it takes to run NP and RSNP grows linearly. It can be seen that RSNP run over 20 iterations is competitive with NP while RSNP run over 1, 2, or 3 iterations takes the least amount of time. We outperform RS and are competitive with NP in terms of total budget as the total number of features increases.

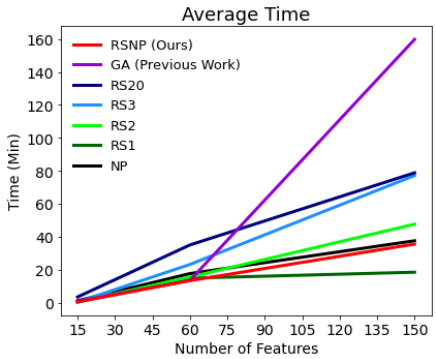

Figure 5: When plotting the average run time of each model on each dataset, we can see that RSNP is substantially quicker or as quick as every other model except RS run over 1 screening iteration as the size of the data increases.

5.4 Summary of Models

The procedures for RS had only been tested on small scale problems (Tsai 2013). As the size of the dataset and level of noise increases, we begin to see RS struggle. The total run time for RS remains relatively low when only conducting a few screening iterations, but as the number of iterations increases, the total computational effort required increases at a staggering rate becoming potentially alarming. No matter the number of screening iterations conducted, the total number of features selected by the algorithm increases along with the dataset size, losing interpretability power. While the accuracy of identifying true features is very higher, this can be attributed to the excessive increase in total features selected by the RS procedure.

In the case of NP, the required computational effort increases linearly as the size of the problem grows, which is very manageable. However, when implementing NP on larger datasets, its effectiveness greatly decreases. Not only does the NP procedure begin to accept a higher number of total features, but the percentage of true features accepted decreases. Similar to RS, NP is effective and computationally efficient for smaller-scaled problems, but lacks the robustness to maintain its performance when being implemented on larger and significantly more noisy problems.

Finally, similar conclusions were drawn from our previous work with SOFS employing GA (Houser et al. 2024) in this numerical analysis. While the level of accuracy in identifying true features slightly declines when using larger datasets, the overall prediction model remains competitively accurate and interpretable. The downside to implementing the GA, which we've seen in previous research studies, is that GA struggles to maintain an adequate level of computational efficiency. In fact, as the size of the problem grows, the computational effort required by the GA increases exponentially.

6 CONCLUSION

FS is an increasingly difficult problem as the size of the data and level of noise increases. An ideal FS procedure is effective, computationally efficient, and robust. Existing FS methods are able to accomplish either effectiveness, computational efficiency, or some balance of them both. However, they lack the robustness to noise that would scale well with the size of the data.

Through our numerical analysis, RSNP has shown its ability to effectively identify true features at a high level of accuracy while accepting very few redundant or irrelevant features. The total computational effort required to run RSNP is also low compared to the other procedures. Finally, RSNP has shown its robustness in its ability to maintain its FS effectiveness and computational efficiency even as the size of the problem grows and the level of noise increases.

Future work will focus on further reducing the computational effort of RSNP while maintaining or improving its performance. There are plenty of mechanisms in the procedure that await further testing and refinement. For example, moving away from uniform random sampling of solutions to establish a more intelligent strategy may allow RSNP to more efficiently navigate the solution space. Examining the effects of a ranking and selection procedure for solution sampling versus simulation replications may also provide insight into the algorithm's performance and efficiency. Finally, with its established effectiveness, we can move towards larger datasets and data from real-world applications.

ACKNOWLEDGMENTS

This work was partially supported by National Science Foundation grant CMMI-2226347.

REFERENCES

Alizadeh, N., K. Vahdat, S. Shashaani, J. L. Swann and O. Y. Özaltın. 2024. "Risk Score Models for Urinary Tract Infection Hospitalization". *Plos One* 19(6):e0290215.

Alrefaei, M. H. and S. Andradóttir. 1999. "A Simulated Annealing Algorithm with Constant Temperature for Discrete Stochastic Optimization". *Management Science* 45(5):748–764.

Andradóttir, S. and A. A. Prudius. 2009. "Balanced Explorative and Exploitative Search with Estimation for Simulation Optimization". *INFORMS Journal on Computing* 21(2):193–208.

Gelfand, S. B. and S. K. Mitter. 1989. "Simulated Annealing with Noisy or Imprecise Energy Measurements". *Journal of Optimization Theory and Applications* 62(1):49–62.

Gutjahr, W. J. and G. C. Pflug. 1996. "Simulated Annealing for Noisy Cost Functions". *Journal of Global Optimization* 8:1–13.

Hong, L. J. and B. L. Nelson. 2006. "Discrete Optimization via Simulation using COMPASS". *Operations Research* 54(1):115–129.

Houser, E., S. Shashaani, O. Harrysson, and Y. Jeon. 2024. "Predicting Additive Manufacturing Defects with Robust Feature Selection for Imbalanced Data". *IISE Transactions* 56(9):1000–1019.

Hunter, S. R. and B. L. Nelson. 2017. *Parallel Ranking and Selection*, 249–275. Cham: Springer.

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi. 1983. "Optimization by Simulated Annealing". *Science* 220(4598):671–680.

Kleijnen, J. P. C., P. Cremers, and F. Van Belle. 1985. "The Power of Weighted and Ordinary Least Squares with Estimated Unequal Variances in Experimental Design". *Communications in Statistics-Simulation and Computation* 14(1):85–102.

Kuhn, M. and K. Johnson. 2013. *Applied Predictive Modeling*. "New York, NY": Springer.

Ólafsson, S. 2004. "Two-Stage Nested Partitions Method for Stochastic Optimization". *Methodology and Computing in Applied Probability* 6:5–27.

Ólafsson, S. and J. Yang. 2005. "Intelligent Partitioning for Feature Selection". *INFORMS Journal on Computing* 17(3):339–355.

Pichitlamken, J. and B. L. Nelson. 2003. "A Combined Procedure for Optimization via Simulation". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 13(2):155–179.

Rong, M., D. Gong, and X. Gao. 2019. "Feature Selection and Its Use in Big Data: Challenges, Methods, and Trends". *IEEE Access* 7:19709–19725 https://doi.org/10.1109/ACCESS.2019.2894366.

Shashaani, S. and K. Vahdat. 2023. "Improved Feature Selection with Simulation Optimization". *Optimization and Engineering* 24(2):1183–1223.

Shi, L. and S. Ólafsson. 2000. "Nested Partitions Method for Stochastic Optimization". *Methodology and Computing in Applied Probability* 2:271–291.

Tsai, S. C. 2013. "Rapid Screening Procedures for Zero-One Optimization via Simulation". *INFORMS Journal on Computing* 25(2):317–331 https://doi.org/10.1287/ijoc.1120.0504.

Vahdat, K. and S. Shashaani. 2020. "Simulation Optimization Based Feature Selection, a Study on Data-Driven Optimization with Input Uncertainty". In *2020 Winter Simulation Conference (WSC)*, 2149–2160 https://doi.org/10.1109/WSC48552.2020.9383862.

Vahdat, K. and S. Shashaani. 2023. "Adaptive Ranking and Selection Based Genetic Algorithms for Data-Driven Problems". In *2023 Winter Simulation Conference (WSC)*, 3424–3435 https://doi.org/10.1109/WSC60868.2023.10408610.

Wang, H., R. Pasupathy, and B. W. Schmeiser. 2013. "Integer-Ordered Simulation Optimization using R-SPLINE: Retrospective Search with Piecewise-Linear Interpolation and Neighborhood Enumeration". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 23(3):1–24.

Xu, W. L. and B. L. Nelson. 2013. "Empirical Stochastic Branch-and-Bound for Optimization via Simulation". *IIE Transactions* 45(7):685–698.

Yan, D. and H. Mukai. 1992. "Stochastic Discrete Optimization". *SIAM Journal on Control and Optimization* 30(3):594–612.

AUTHOR BIOGRAPHIES

ETHAN HOUSER is a second year Ph.D. student in the Edward P. Fitts Department of Industrial and System Engineering with research interests in simulation and stochastic optimization. His research includes feature selection, big data, and data-driven modeling. His email address is ephouser@ncsu.edu.

SARA SHASHAANI is an Assistant Professor and Faculty Scholar in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University. Her research interests are simulation optimization and probabilistic data-driven models. She is a co-creator of SimOpt library. Her email address is sshasha2@ncsu.edu and her homepage is https://shashaani.wordpress.ncsu.edu/.