Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

ROBUST DATA-DRIVEN OPTIMIZATION USING MACHINE LEARNING AND MONTE-CARLO SIMULATION

Vincent Béchard

Différence S.E.N.C.
CP 443, Succ. Ahuntsic
Montréal (QC), H3L 3N9, CANADA

ABSTRACT

This paper presents a framework that was developed to achieve data-driven robust optimization of processes. The main idea is to automatically build a single and global predictive model using a machine learning technique (random forests), and then to use a derivative-free black-box optimization technique (MADS) to maximize a performance criterion. Monte-Carlo simulation is used to estimate pointwise prediction variance. This automated framework is designed to find optimal variance-stabilizing solutions while preserving the noisiness, non-smoothness, non-linearity, non-convexity and disjoint nature of real-life data. The time it takes to prepare the dataset depends on the volume of data, but this step occurs only once in the procedure. The time it takes to iterate during optimization depends on three user-specified quantities; therefore, the iterative portion is insensitive to data volume. Implementation was done using the R programming language which offers a wide variety of data processing, modelling and optimization capabilities.

1 INTRODUCTION

When performance data is available, it might seem straightforward to seek for optimal or most desirable conditions. But when variability (or noise) is present and several decision variables must be manipulated simultaneously, this task quickly becomes complex. Simply sorting the data is not guaranteed to work since many combinations could be "optimal" and the user would still have to make judgment call to choose the most desirable one. Several alternatives are possible to reduce this complexity, such as developing a simulation model or applying predictive machine learning techniques to compress the volume of data and to obtain a simplified process representation. Then a mathematical programming algorithm can be used to search for most desirable conditions.

In this research, the focus was set on exploiting available performance data as directly as possible to support the optimization task. A framework was developed to integrate techniques from the fields of data science (predictive analytics) and mathematical programming (prescriptive analytics). The need addressed by this research is to have an automated workflow to discover the data domain, predict the performance of new trial points, and search for points with the highest desirability while remaining within the data domain.

In this paper, X is the dataset of predictor observed values, Y is the dependent output, $f(X)$ is the predictions of Y using X, $\Omega(X)$ is the data domain, x is a point (or solution) in the space of X, and x^* is the optimal solution. Hence, given a dataset, the designed workflow must be capable of mapping of the data domain $\Omega(X)$ structure, building a predictive model $f(X)$ valid over $\Omega(X)$, and solving an optimization problem which considers the impact of variability and modelling error.

To limit the current research scope, it is assumed that: the problem is specified (known independent actionable variables and dependent outcome), data collection-aggregation-cleansing is done *a priori*, there is one continuous Y and several quantitative (discrete or continuous) X variables, and all non-actionable variables are removed from the modelled dataset (their uncontrollable influence will impact the model's prediction error). The following practical aspects and conceptual features were considered during the framework design process:

- Unsupervised discovery of $\Omega(X)$: must be robust to the presence of outliers, able to map disjoint subgroups or clusters, does not assume dataset is enclosed in a convex hull;
- Universality of $f(X)$: this is the predictive black-box, it must implicitly manage interactions, multicollinearity, non-linearities;
- Optimizer robustness: must deal with a black-box, which means working without assuming differentiability, convexity, smoothness, compact data domain, etc.;
- Solution robustness: for similar predicted levels, the procedure must recommend robust solutions, or, solutions for which the outcome is insensitive to perturbations of the predictors;
- Automatic building of $\Omega(X)$ and $f(X)$: little to no human intervention is required to tune parameters allowing models preparation and calibration;
- Scalability: the procedure must be capable of handling large datasets in reasonable time.

This paper is divided as follow. Section 2 provides explanations regarding the sources of complexity and possible strategies from the literature. Section 3 describes the designed framework with rationales supporting design choices, completed by a schematic overview of the implementation. Section 4 presents two applications of the framework with comparison between optimization algorithms. Finally. Section 5 summarizes the findings and suggests future works.

2 CHALLENGES IN DATA-DRIVEN OPTIMIZATION

Combining predictive analytics (what is the value of Y given X?) and prescriptive analytics (which x^* gives to most desirable $f(x)$?) is a complicated endeavor, becoming even worse in the presence of voluminous data, as explained in Hertog and Postek (2016). Running an iterative optimization algorithm on top of large datasets is time consuming. Simplification is a necessity to overcome this issue and to provide faster advises. Simplification can take the form of simulations, sets of equations, or machine learning algorithms for instance. The information from the data must be compressed somehow into a fast-response mechanism.

Building a simplified representation must be done with care, since one must preserve the process behavior and system dynamics. Characteristics such as non-linearity, multicollinearity or correlations structure, and intrinsic variability still must appear in the simplified data representation.

An optimization algorithm will create trial solutions x and evaluate them through the simplified data representation $f(x)$. Hence, if the relationship $f(x)$ is non-linear, noisy or stochastic, non-differentiable, the optimization task is more challenging. Another important aspect to consider is: trial solutions have to belong to "known" data regions, or: $x \in \Omega(X)$. Some interpolation is acceptable, but extrapolation to regions with no data supporting it can lead to "false" optimum, meaning the optimizer thinks there exist a very desirable solution but no data can confirm it.

Finally, real life optimization problems often come with constraints, and/or bounds on variables. Due to the presence of variability, not all solutions have the same interest: for similar performance levels, a practitioner will usually prefer the lesser variable one, or, the more robust solution.

2.1 Simplified Data Representations

The general idea is to compress voluminous data into a synthetic fast-calculation $f(x)$ valid when $x \in \Omega(X)$. This is achieved by simplifying the information from the data to its essential behaviors and features.

In one hand, data can be utilized to understand a process behavior. Then this understanding is translated into sets of equations or models to be evaluated without needing the original data. The term "simulation" used in this paper refers to this approach. Contemporary simulations are typically based on differential and algebraic equations to represent steady or dynamic states, or they could represent cascading discrete events. In all cases, process data is used to calibrate the simulation and to validate the outputs.

Simulations have the advantages of capturing behavior complexity, non-linearities and discreteness. However, calculation time can range from seconds to hours. The biggest drawbacks are the time required to setup complex simulations and the need for specialized resources to model each different system. The use of simulations into an optimization context has been discussed, for instance, in Amaran et al. (2014), by Béchard et al. (2008), and in numerous books and journals.

In the other hand, simulation-free representations based purely on process data do exist. This empirical predictive modelling is the "machine learning" approach: different modelling techniques establishing a multivariate relationship between *X* and *Y* variables. There exist well-known techniques such as: Generalized Linear Regression, Partial Least Squares, Neural Networks, Random Forests, Gradient Boosting, etc. Interesting descriptions and benchmarking of these popular techniques are presented in Malakar et al. (2018) and Nawar and Mouazen (2017).

Machine learning approaches are typically easier to setup than simulations since they consist of generic preset rules iteratively processing data. They make direct use of data and attempt to capture the underlying $f(x)$ relationships. However, some methods can have difficulties dealing with nonlinearity, collinearity or outliers. The reason for that is the absence of expert-provided knowledge for each specific case, as for simulation.

In general, training a machine learning model implies splitting the sample in two parts (training, validation), or using cross-validation to ensure the predictions can generalize (prevent overfitting). Using machine learning techniques in the context of process optimization has been discussed, for example, in Gröger et al. (2012) and in Shi (2005).

2.2 Optimization Strategies

Mathematical programming approaches for the process optimization context have been extensively studied and documented. The retrospective by Biegler and Grossmann (2004) provides a comprehensive description of methods and typical applications. The major interest of these methods is the theoretical support: at convergence, the solution is proven to be at least a local optimum. However, due to the nonlinear and noisy nature of historical data, it is sometimes challenging to use them, especially when it is not possible to calculate a gradient.

Biegler and Grossmann (2004) pointed out the interest of direct search methods. Direct search methods have not always been popular. Their pre-2000's history was well documented in Wright (1997), all summarized in the title: "Direct Search Methods: Once Scorn, Now Respectable". These methods became popular again in the 2000's. The revival of direct search and a general definition of this class of algorithms can be found in Lewis et al. (2000). Several heuristic and derivative-free approaches are today common in software: simulated annealing, ant colony, particle swarm, pattern search, etc.

Direct search methods are suited for noisy black-box type of problems where no derivative is available. They use direct values (predicted by simulation or by machine learning) and adapt their search scheme to seek for desirable solutions; a comparative description is outlined in Amaran et al. (2014). Despite the fact that they usually perform well in practice, these methods are not systematically supported by thorough mathematical convergence analyzes.

2.3 Gap Between Prediction and Optimization

As coined by Hertog and Postek (2016), a new methodology is required to close the bridge between predictive techniques (simulation or machine learning) and optimization, referred as prescriptive techniques. Several methodologies going into that direction have been proposed for instance in Gröger et al. (2012) and Shi (2005), although these papers focused on data engineering and predictive analytics.

Some methods attempt to close this gap by iteratively fitting a local response surface model (usually using kriging), searching the best solution in the region, pursue the best descent direction, then sample and fit a new model, etc. Examples of these sequential fitting-searching strategies can be found in Jones et al.

(1998) and Liu et al. (2014). However, these methods do not use a global predictive model, and local model fitting can require different parameters depending on local properties of the data.

This current research explores recommendations by Hertog and Postek (2016), assuming data is available. The practical framework presented in the next section is implementing recently available predictive techniques and prescriptive analytics using direct search methods.

3 PROPOSED FRAMEWORK

The data-driven performance optimization task and its R implementation are presented in this section. The concept of black-box optimization is explained, followed by a description of all selected features linking the raw data to the optimizer output in the framework.

3.1 The Black-Box Approach

The idea of using a black-box in optimization is summarized in Figure 1: the optimizer asks the "problem" to evaluate a candidate x, then the "problem" returns a value $f(x)$ if it succeeded in evaluating it for x. Detailed description of this principle can be found in Audet et al. (2008) and in Le Digabel et al. (2018).

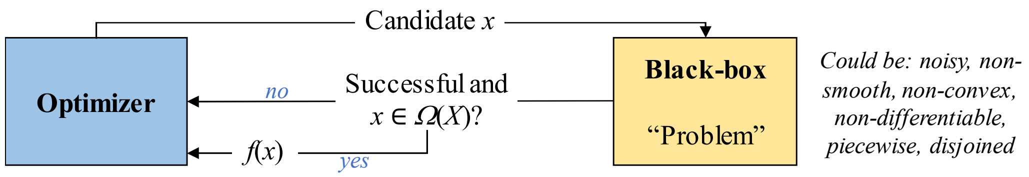

Candidate x
Optimizer
no
Successful and x ∈ Ω(X)?
f(x)
yes
Black-box
"Problem"
Could be: noisy, non-smooth, non-convex, non-differentiable, piecewise, disjoined

Figure 1: Schematic black-box optimization process.

The black-box can be any imaginable way to relate X to Y: a simulation, direct adjustments and readings on the live process, an empirical model or any predictive technique from modern data science toolbox. Because knowledge on the nature of "problem" is not provided, optimizing it has to be performed under weak assumptions: differentiability may not be defined, convexity is not guaranteed, smoothness is likely not probable, $f(x)$ could be piecewise or disjoint (if it can be evaluated), the domain (how X is spread) could be disjoint and non-convex.

In the R community, several packages offer optimization capabilities. The webpage maintained by Theussel et al. (2019) presents these packages. The derivative-free algorithms Hooke-Jeeves and Nelder-Mead offered in "*dfoptim*" (Varadhan and Borchers 2018) were considered for this research.

In addition to these robust techniques, the Mesh Adaptive Direct Searches (MADS) (Audet et al. 2008) algorithm has been implemented in R during this research project. This algorithm is a modern derivative-free technique supported by thorough mathematical convergence analyzes (Le Digabel et al. 2018). It has been designed for constrained optimization of black-boxes as described earlier.

3.2 Model for the Data Domain

To remain valid, the optimal solution will have to be supported by observations from the historical data. This adds an implicit constraint during the optimization process: candidates x have to be within the data domain or envelop $\Omega(X)$. In practice, how can this condition be verified?

When the dataset is "not too big", the k-Nearest Neighbors method can be used. For a chosen small k, the method finds the k nearest points to current observation and then determines the maximum distance. Then any new x can be compared against the entire dataset to verify if it is sufficiently close to several neighbors. However, the test for $x \in \Omega(X)$ will be executed a large number of times during the optimization process. If the historical dataset is large, this is a known performance issue.

Techniques exist to identify multivariate local outliers such in Fan et al. (2006) or in Chepenko (2018). They bring the concept of "reachability", and outliers or those unreachable observations. Combining this idea to clustering is an approach than could resolve the performance issue for large datasets: clusters can reduce the essential information to centroids having their own influence sphere (assuming dataset is normalized). Using many centroids, for instance 50 or 100, could result in covering the data domain through a series of overlapping spheres.

Many clustering algorithms exist in R packages, going from the classical *k*-means, to *k*-medoids, PAM, CLARA, mini-batch *k*-means, etc. (Leisch and Gruen 2019). Comparative studies show that although not the most accurate, the *k*-means algorithm remains one of the fastest (Kondal 2017; Schubert and Rousseeuw 2018). Centroids accuracy is not critical in our specific context of covering a dataset, but speed and ability to reach all zones where data is present is critical. Therefore, the *k*-means algorithm from the base R program is the selected approach.

The designed framework uses a "large" number of centroids (by default 100) to cover the range of data. Then, distances of each observation to its assigned centroid is calculated. The value of k (number of centroids) can be determined by trial and error until at least 95% of the dataset is covered. The influence sphere of a centroid is defined as the 99th percentile of associated distances. Lower quantile values such as 95% are possible but could be more aggressive and prune interesting regions from the domain. This allows filtering very extreme values that would likely not be interesting optimal solutions. Finally, any new candidate is "within reach" as long as it is included in one of the centroids influence sphere (relative distance to centroid ≤ 1).

Several features of this approach are illustrated on Figure 2. Centroid #1 represents a disjoint sub-group while centroids #2 to #4 are overlapping spheres. Point A is out of reach and would not be accepted as potential optimal candidate. Point B is closer to centroid #4 in absolute distance but out of its influence sphere. But Point B is within centroid #2 influence sphere, thus it is a valid candidate in $\Omega(X)$.

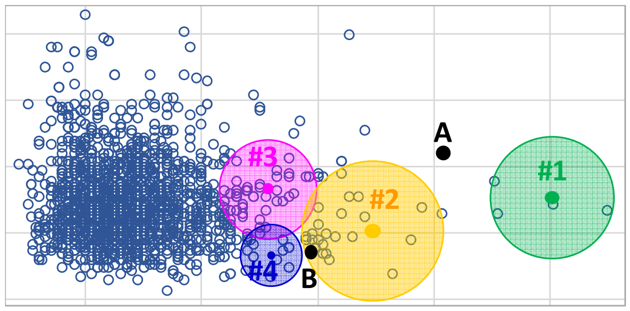


Figure 2: Illustration of the multiple centroids data domain on hypothetical X_1 and X_2.

Using this approach has some advantages: it is distribution-free, non-convex and non-linear shapes can be captured, disjoint sub-groups (data sub-groups far for the bulk of the data) can be mapped with a centroid, and internal X correlation structures are preserved. For instance, if two variables X_1 and X_2 are highly correlated, the plot X_1 versus X_2 will look like a straight line and all centroids will be along that line.

The drawbacks of this approach are that $\Omega(X)$ has a non-convex hull (intersections between overlapping spheres are indeed concave) and $\Omega(X)$ is possibly disjoint. The black-box optimizer must be capable of searching in that kind of domain. Algorithms from the Generalized Reduced Gradient (Biegler and Grossmann 2004) family are not recommended.

Consequently, this model of $\Omega(X)$ will allow reasonable interpolation within centroids influence spheres, but extrapolation far from $\Omega(X)$ will be disallowed. During the optimization process, a candidate x is compared to each of the limited number of centroids, not to entire dataset as with the k-Nearest Neighbors. This should not impose performance issues.

3.3 Model for Performance Prediction

This is the classical application of predictive modelling in machine learning toolbox. The idea is to obtain the most accurate and generalizable (to new data) predictions for $f(x)$ over $\Omega(X)$. Explanatory features and statistical inference would be a great by-product, but prediction capability as a black-box must remain the key ingredient required for the later optimization algorithm. The distinction between explanatory and predictive modelling is discussed in Shmueli (2010).

There exist numerous predictive modelling techniques, for instance: generalized regression, regression trees, neural networks, support vector machines, gradient boosting, random forests, etc. (Malakar et al. 2018). Benchmarking studies show that, in general, neural networks, gradient boosting and random forests approaches tend to be better predictors (Gröger et al. 2012; Han et al. 2017; Malakar et al. 2018; Nawar and Mouazen 2017). To train predictive models, the typical methodology requires to split the sample in two portions: training and validation sets. Models are calibrated using the training set and their performance is evaluated using the validation set. Alternatively, cross-validation could be used: this is an iterative automated version of sample splitting.

Random forests approach was retained for the framework because of its "out-of-bag" feature. The principle behind random forests is to build a large set of regression trees using different subsets of predictors and bootstrap samples; then predictions from each tree are combined. Figure 1 in Han et al. (2017) illustrates very well the random forest mechanism. This modelling technique can handle nonlinearities and interactions while being robust to the presence of outliers and missing values. Moreover, building a random forest is, generally speaking, faster than training neural networks or gradient boosting.

The out-of-bag mechanism of random forests could be seen as implicit cross-validation. Unused observations of each tree (the "out-of-bags") are used to evaluate the tree performance, and the forest's performance statistics are built using each tree out-of-bag observation. This mechanism can be used to build a predictive model without explicitly splitting the sample or using cross-validation. This point addresses some design criteria outlined in the Introduction: universality of $f(x)$ and automatic training of $f(x)$.

In the R community, several packages can perform random forests. The later optimization algorithm will repetitively predict new data (trial points) using a pre-trained forest. The "ranger" package has been selected because it shows the fastest running time (Wright and Ziegler 2017).

3.4 Model for the Robustness Estimation

The ultimate framework objective is to identify x^*, the most desirable and robust solution. Maximizing or minimizing $f(x)$ over $\Omega(X)$ could produce solutions with a large variability which in practice would not be satisfactory solutions. Therefore, if it is possible to find several x with similar $f(x)$ levels but with different local $f(x)$ standard deviations (here: $s(f(x))$), then the x with minimal $s(f(x))$ will be preferred.

To obtain an estimate of $s(f(x))$, a Monte-Carlo simulation used. This consists in generating several points randomly spreading around x in a radius r, to evaluate them and to calculate the standard deviation of the predicted values. Therefore, we denote $s(x;r)$ being the estimate of $s(f(x))$ for perturbed X sphere of radius r.

The ratio $s(x;r)/r$ provides useful information (for normalized values): if <1, the solution x is a variance stabilizer (variability of the outcome lesser than the income); if >1, the solution is unstable (variability is amplified). Figure 3 illustrates variance amplification (point a) and variance stabilization (point b): same perturbation width on the x axis, different width on the $f(x)$ axis even though $f(a)=f(b)$.

Let's consider that the predictive model obtained from Section 3.3 has an average error of amplitude *RMSE* (root means square error of the random forest out-of-bag predictions). The measure of robustness $m(x)$ considered for this framework is given by $m(x) = RMSE \times t(0.95, n\text{-}1) \times s(x;r) / r$, where $t(\bullet)$ is the Student distribution and n is the number of valid simulated points used to calculate $s(x;r)$. In other words, $m(x)$ is a 95% confidence prediction interval built using a model error corrected by the variability increase or decrease due to local model non-linearity.

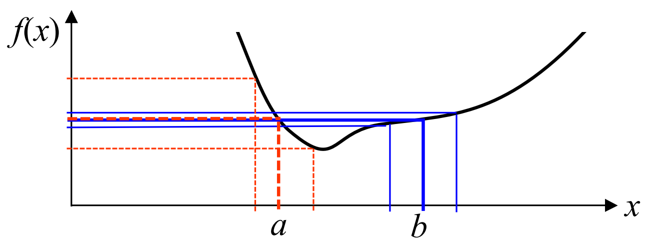


Figure 3: Illustration of variance stabilization and amplification.

3.5 Statement of the Optimization Problem

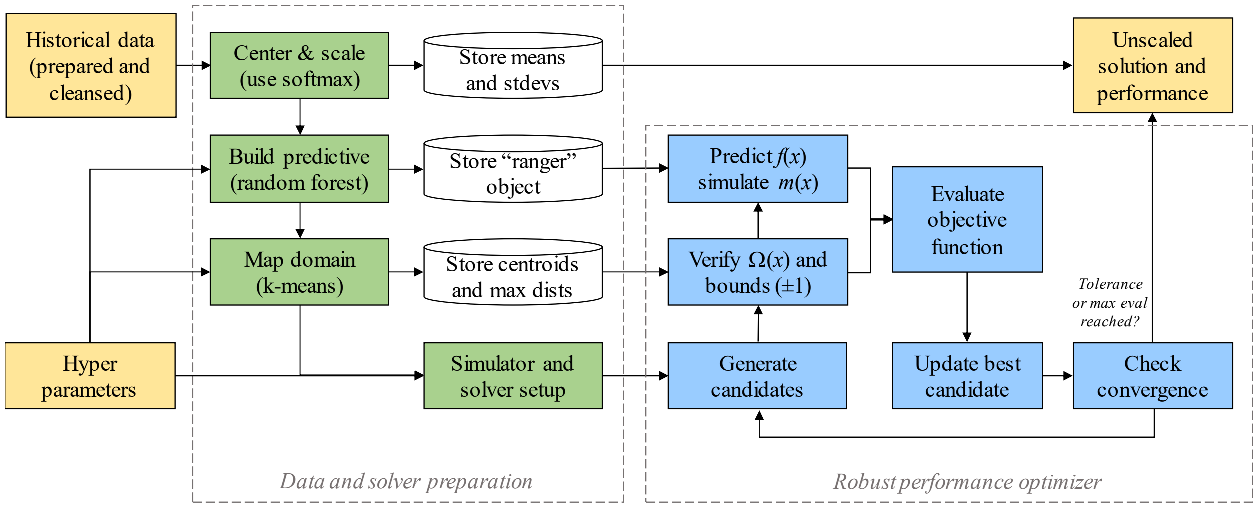
Without loss of generality, the fundamental optimization problem to be solved is find x^* such that $f(x)$ and $s(f(x))$ are minimized simultaneously while $x^* \in \Omega(X)$. In case of maximization, the goal is to minimize $-f(x)$; in case of target matching, the goal is to minimize $|f(x)\text{-}target|$. Hence, the direct formulation is a bi-objective constrained minimization problem.

The nature of the black-box and its domain defined in Sections 3.2 and 3.3 with the addition of the robustness measurement in Section 3.4 already impose the need for derivative-free optimization methods. Although there exist algorithms for constrained bi-objective black-box (Le Digabel et al. 2018), we defined an unconstrained formulation which is more practical to solve.

The chosen strategy is to minimize the bound-constrained objective function $\boldsymbol{(f(x) + m(x)) \times d(x)}$, where $f(x)$ is the random forest predicted values as described in Section 3.3, $m(x)$ is the robustness estimate as described in Section 3.4 and $d(x)$ is the maximum between 1 and the least relative distance between x and a centroid as in Figure 2; if the relative distance is >3, $f(x)$ is forced to $+\infty$. The threshold value of 3 has been tuned empirically to give good results on various datasets. The rationale behind $d(x)$ is to penalize distances outside of $\Omega(X)$ while not favoring any solution inside $\Omega(X)$ (this is the penalty version of constraint $x \in \Omega(X)$). Therefore, the optimizer will search for solutions inside or close to the data domain that decrease the predicted value and stabilize the prediction's variance.

3.6 Putting It All Together

The diagram in Figure 4 summarizes the implemented workflow supporting the framework. Elements in yellow are user-provided or user-delivered facts, elements in green are the machine learning features, and elements in blue are the black-box optimization features.



Historical data (prepared and cleansed)
Center & scale (use softmax)
Store means and stdevs
Unscaled solution and performance
Build predictive (random forest)
Store "ranger" object
Predict $f(x)$ simulate $m(x)$
Evaluate objective function
Map domain (k-means)
Store centroids and max dists
Verify $\Omega(x)$ and bounds (±1)
Tolerance or max eval reached?
Hyper parameters
Simulator and solver setup
Generate candidates
Update best candidate
Check convergence
Data and solver preparation
Robust performance optimizer

Figure 4: Workflow of the proposed framework.

The hyper parameters mentioned in Figure 4 are the parameters exposed to the user such as: number of centroids, confidence level for distance cut-off, number of trees and number of terms per tree in the random forest, number of Monte-Carlo iterations and perturbation radius *r*, number of points in the pre-optimization random search, choice of optimization method, optimizer tolerance and maximum number of evaluations, and the random seed.

Datasets can be arbitrarily large. But this workflow has two compression mechanisms: centroids & their influence zones, and random forests. Their preparation time is indeed influenced by the dataset size, although this step happens once, before starting the optimization iterations. Evaluating trial points consists in comparing to centroids plus evaluating a random forest for each point generated by the Monte-Carlo simulations. Thus, it is expected that the optimization time will be influenced by the number of centroids, number of trees in the forest (which is influenced by the number of *X* variables), and number of Monte-Carlo iterations; gladly this is insensitive to original dataset size.

4 EXAMPLES OF APPLICATION

In this section, two examples are presented. They illustrate the behavior of the framework in terms of data domain mapping, random forest quality of fit, and comparison between three derivative-free optimizers (Hooke-Jeeves and Nelder-Mead from the "*dfoptim*" package, and MADS implementation in R). Additional charts have been built to illustrate the trade-off between predicted levels and prediction variance.

4.1 Improving White Wine Quality

This is an open-source machine learning dataset by Cortez et al. (2009). Approximately 5,000 observations: measurements of 12 chemical properties and associated quality index between 3 and 9. The objective here is to determine the "magical" recipe that would produce a repeatedly high-quality wine.

The workflow explained in Section 3 has been applied with 100 centroids and 99th percentile cut-off, 300 trees and 4 terms per tree, 100 points pre-optimization search, 25 Monte-Carlo iterations per point. As a result, 99.27% of the data was included in $\Omega(X)$, the random forest R^2 was 56% with a RMSE of 0.345. In Figure 5, performance of the three derivative-free algorithms is compared.

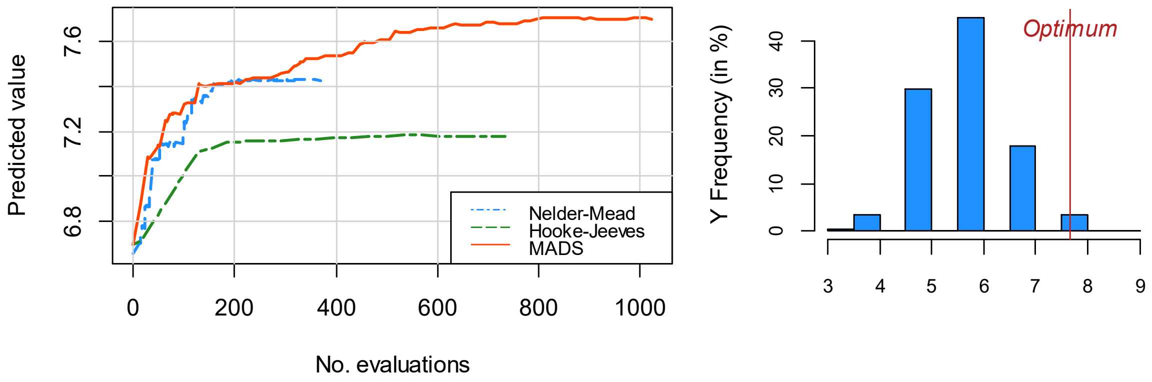


Figure 5: Optimizers performance on the white wine quality dataset with optimum compared to histogram of white wine quality observed values.

The MADS algorithm performed similarly to the Nelder-Mead algorithm for the first 300 evaluations, but it was able to converge to more attractive wine quality levels. In Figure 6, more details are provided concerning: relative distance to nearest centroid (how far from $\Omega(X)$ limit?), values of $m(x)$ (described in Section 3.4, how robust is x?), and on the predicted wine quality $f(x)$. In the first portion of the optimization process, MADS quickly reduced the variability of $f(x)$, and then maintained a certain robustness level while increasing $f(x)$ values.

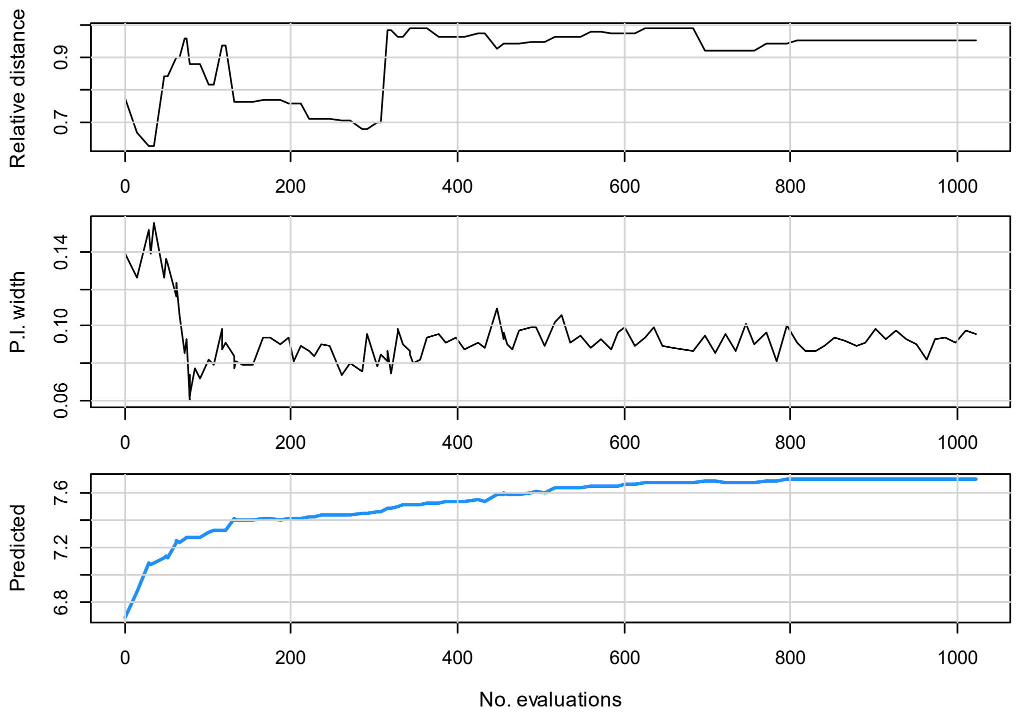


Figure 6: Details on MADS evolution on the white wine quality dataset.

4.2 Optimizing Concrete Strength

This is an open-source machine learning dataset by Yeh (1998). Mixing high resistance concrete with homogeneous (low variability) properties has always been a challenge. The dataset has 1,030 observations of 9 metrics and associated measurements of compressive strength. The objective is to find a x^* such that the compressive strength is as high as possible with the least amount of variability.

The workflow explained in Section 3 has been applied with 50 centroids and 99th percentile cut-off, 300 trees and 5 terms per tree, 500 points pre-optimization search, 25 Monte-Carlo iterations per point. As a result, 96.5% of the data was included in $\Omega(X)$, the random forest R^2 was 92.5% with a RMSE of 20.9 (best reported R^2 in Yeh (2009) is 92.2%, random forests were not considered in that paper). Algorithms performance is reported in Figure 7.

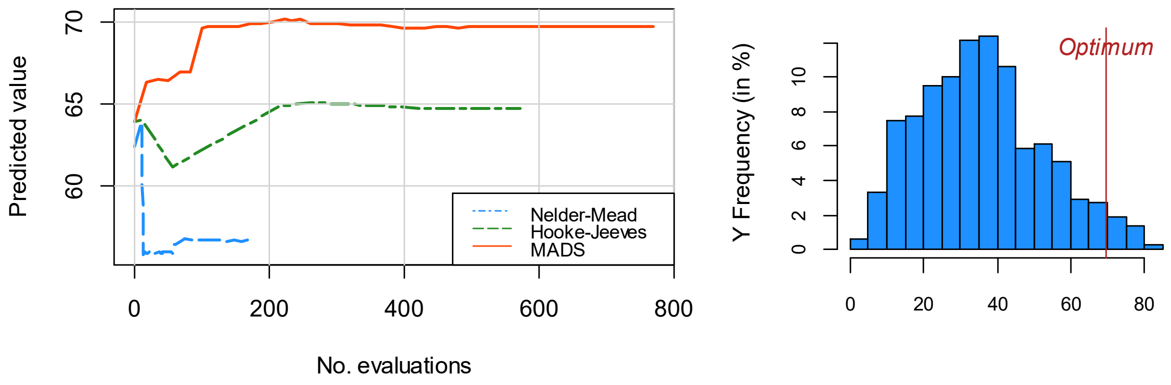


Figure 7: Optimizers performance on the concrete compressive strength dataset with optimum compared to histogram of compressive strength observed values.

The MADS algorithm performed best during the entire process. The Hooke-Jeeves algorithm made improvements to $m(x)$ before unsuccessfully attempting to improve $f(x)$. The Nelder-Mead algorithm reduced $m(x)$ and was not able to move forward (this phenomenon is known as simplex degenerescence, Wright 1997). In Figure 8, more details are provided to understand MADS evolution during the process.

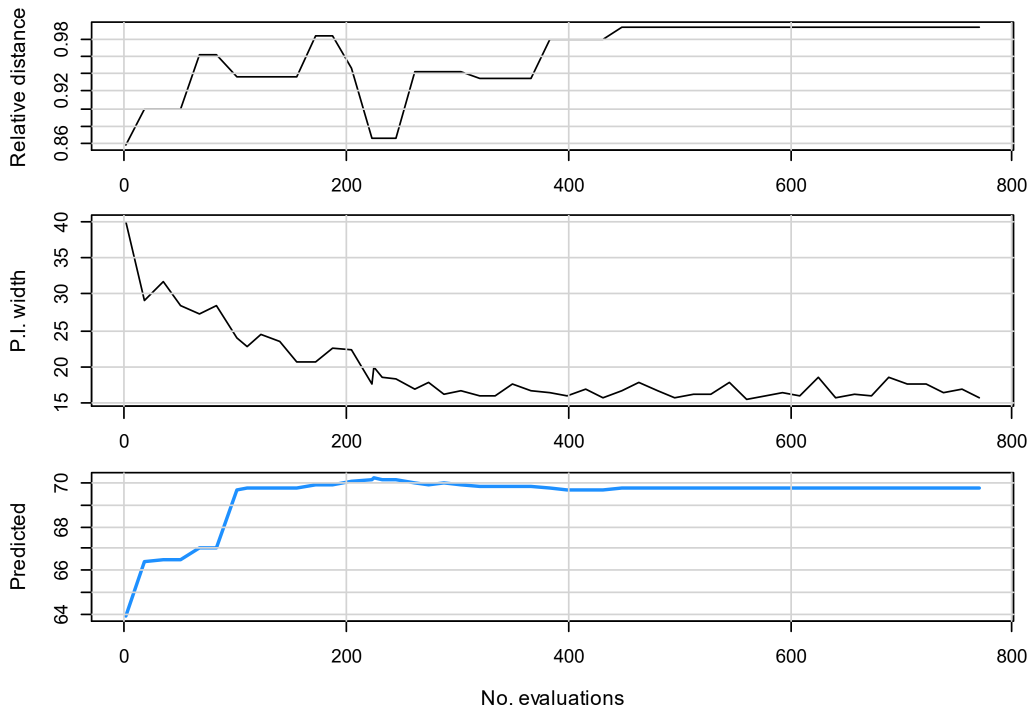


Figure 8: Details on MADS evolution on the concrete compressive strength dataset.

The top chart in Figure 8 shows that the optimum lies on the hull of $\Omega(X)$. Searching in the vicinity of the domain boundary is in itself a source of complexity: the optimizer faces frequent infeasible trial solutions. The middle chart in Figure 8 reveals the noisy nature of this dataset: the variability of $f(x)$ had to be decreased by a factor of 3. Finally, the bottom chart shows a non-intuitive behavior: value of $f(x)$ is slightly decreasing in the second half. This is the trade-off made by the optimizer in order to truly maximize $(f(x) + m(x)) \times d(x)$ (as described in Section 3.5), not only to increase $f(x)$.

5 CONCLUSION AND FUTURE WORKS

In this paper, we proposed a framework to conduct robust optimization task directly using data. The rationale behind this work is to exploit existing data using a fast optimization process to provide relevant insights to experts. This task can be conducted to avoid deploying complex time-consuming specialized simulations. Hence, the framework was designed to extract as much useful information as possible from the data. The framework has also considered using a single global predictive model to be searched by global optimizers to ensure consistency and statistical soundness of the results.

One of the major challenges with today's large datasets is the volume: computation time of iterative procedures can become important to the point it is not of a practical usefulness. Therefore, two compression mechanisms were included: mapping the data domain $\Omega(X)$ using k-means centroids (and identifying each centroid influence zone) and estimating the $f(x)$ relationship using random forests, a fast and powerful machine learning technique. No matter what is the size of the original dataset, the optimizer will always iterate on a fixed number of centroids and fixed number of trees in the random forest.

Another practical challenge is robustness. In order to identify optimal solution that is as invariant as possible, a Monte-Carlo simulation step was included into the framework. This simulation provides an estimate of the variability of $f(x)$ around x. A corrected prediction interval $m(x)$ is derived and added to the optimization objective function.

Combining all these ingredients gives an optimization problem that is potentially: noisy, non-smooth, non-linear, non-convex, non-differentiable, over a disjoint and nonconvex domain. The problem to be solved is the simultaneous $f(x)$ and $m(x)$ minimization while remaining within $\Omega(X)$, inner domain interpolation is permitted. At this stage, the problem was formulated as unconstrained minimization. Several derivative-free algorithms have been tried and MADS (Mesh Adaptive Direct Searches) appeared to be the most successful.

There are important facts concerning executing time: preparing the data (obtaining centroids and training a random forest) is indeed dependent of the dataset size, but the iterative optimization is not. Optimization time is proportional to user-determined quantities: number of centroids, number of trees in the forest and number of Monte-Carlo iterations.

Several studies could be undertaken to improve the framework. A thorough literature review could potentially help finding a more efficient or accurate technique to find the hull of $\Omega(X)$. There exist techniques to use correlated data in random forests; this could possibly enhance the machine learning component since correlated data are common situations. A design of experiment on the framework hyper-parameters could help understanding better their interactions and provide wiser default values.

The optimization portion can be improved, since the constrained bi-objective black-box problem was turned into unconstrained single objective for practical reasons. Techniques to solve the real constrained bi-objective do exist, however some work might be required to adapt them to the framework and to implement them in R. A possible path forward could be to develop a R interface to use NOMAD (the software developed by the MADS authors).

REFERENCES

Amaran, S., N. V. Sahinidis, B. Sharda, and S. J. Bury. 2014. "Simulation Optimization: A Review of Algorithms and Applications". *4OR Quarterly Journal of Operations Research* 12(4):301-333.

Audet, C., V. Béchard, and S. Le Digabel. 2008. "Nonsmooth Optimization Through Mesh Adaptive Direct Search and Variable Neighborhood Search". *Journal. of Global. Optimization* 41(2):299-318.

Béchard, V., C. Audet, and J. Chaouki. 2008. "Spent Potliner Treatment Process Optimization Using a MADS Algorithm". *Optimization and Engineering* 9(2):143-160.

Biegler, L.T. and I. E. Grossmann. 2004. "Retrospective on Optimization". *Computers and Chemical Engineering* 28(2004):1169-1192.

Chepenko, D. 2018. *A Density-Based Algorithm for Outlier Detection*, towardsdatascience.com/density-based-algorithm-for-outlier-detection-8f278d2f7983, accessed 26th August 2019.

Cortez, P., A. Cerdeira, F. Almeida, T. Matos, and J. Reis. 2009. "Modeling Wine Preferences by Data Mining from Physicochemical Properties". *Decision Support Systems* 47(4):547-553.

Fan, H., O. R. Zaïane, A. Foss, and J. Wu. 2006. "A Nonparametric Outlier Detection for Effectively Discovering Top-N Outliers from Engineering Data". In *PAKDD 2006: Advances in Knowledge Discovery and Data Mining*, edited by W. K. Ng, M. Kitsuregawa, J. Li and K. Chang, 557-566. Berlin, Germany: Springer.

Gröger, C., F. Niedermann, and B. Mitshchang. 2012. "Data Mining-driven Manufacturing Process Optimization". In *Proceedings of the World Congress on Engineering 2012*, edited by S. I. Ao, L. Gelman, D. W. L. Hukins, A. Hunter and A. M. Korsunsky, 1475-1481. London, U K: Newswood Limited.

Han, T., D. Jiang, Q. Zhao, L. Wang, and K. Yin. 2017. "Comparison of Random Forest, Artificial Neural Networks and Support Vector Machine for Intelligent Diagnosis of Rotating Machinery". *Transactions of the Institute of Measurement and Control* 40(8): 2681-2693.

Hertog, D. D. and K. Postek. 2016. *Bridging the Gap Between Predictive and Prescriptive Analytics - New Optimization Methodology Needed*, www.optimization-online.org/DB_FILE/2016/12/5779.pdf, accessed 26th August 2019.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal on Global Optimization*13(4):455-492.

Kondal raj, C. 2017. "Comparison of k Means, k Medoids, Dbscan Algorithms Using DNA Microarray Dataset". *International Journal of Computers and Applied Mathematics* 12(1):344-355.

Le Digabel, S., C. Audet, V. Rochon-Montplaisir, and C. Tribes. 2018. "Blackbox Optimization With the MADS Algorithm and the NOMAD Software". Presentation at IREQ, 2018-06-04, Montréal (Canada), 1-47

Leisch, F. and B. Gruen. 2019. *CRAN Task View: Cluster Analysis & Finite Mixture Models*, https://cran.r-project.org/view=Cluster, accessed 26th August 2019.

Lewis, R. M., V. Torczon, and M. W. Trosset. 2000. "Direct Search Methods: Then and Now". *Journal of Computational and Applied Mathematics* 124(1-2):191-207.

Liu, C., G. Pedrielli, and S. H. Ng. 2014. "eTSSO: Adaptive Search Method for Stochastic Global Optimization Under Finite Budget". *Journal on Global Optimization*13(4):455-492.

Malakar, P., P. Balaprakash, V. Vishwanath, V. Morozov, and K. Kumaran. 2018. "Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications". *The 9th International Workshop on Performance Modeling, Benchmarking, and Simulation of High-Performance Computer Systems*, 12th November 2018, Dallas, TX, USA, 33-44.

Nawar, S. and A. M. Mouazen. 2017. "Comparison Between Random Forests, Artificial Neural Networks and Gradient Boosted Machines Methods of On-Line Vis-NIR Spectroscopy Measurements". *Sensors* 17(10):2428.

Schubert, E., and P. J. Rousseeuw. 2019. "Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms". *arXiv:1810.05691*.

Shi, Y. 2005. "Optimization-Based Data Mining Techniques with Applications". Technical Report Number: 2005-05 November, Department of Mathematics and Computing Science, Saint Mary's University, Canada.

Shmueli, G. 2010. "To Explain or to Predict?". *Statistical Science* 55(3):289-310.

Theussl, S., F. Schwendinger, and H. W. Borchers. 2019. *CRAN Task View: Optimization and Mathematical Programming*, https://cran.r-project.org/view=Optimization, accessed 26th August 2019.

Varadhan, R., and H. W. Borchers. 2018. *Package dfoptim: Derivative-Free Optimization*. https://cran.r-project.org/package=dfoptim, accessed 26th August 2019.

Wright, M H. 1996. "Direct Search Methods: Once Scorned, Now Respectable". In *Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis*, Harlow, United Kingdom: Addison-Wesley, 27th-30th June, 191-208.

Wright, M.N., and A. Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". *Journal of Statistical Software* 77(1):1-17.

Yeh, I.-C. 1998. "Modeling of Strength of High-Performance Concrete Using Artificial Neural Networks". *Cement and Concrete Research* 28(12):1797-1808.

AUTHOR BIOGRAPHY

VINCENT BÉCHARD has been working since 2004 as a consultant in the fields of: statistical analysis and modelling, predictive and prescriptive analytics, machine learning, discrete event and Monte-Carlo simulation, and robust optimization. His data science and applied mathematics skills are highly valuable to solve industrial problems such as: debottlenecking of operations, increasing productivity, estimating potential throughput, assessing risk, testing and optimizing asset utilization. He is familiar with parts and materials handling, operations and traffic, reliability and maintenance, logistics and scheduling, and supply chains. His diversified experience includes surface and underground mining (aluminum, iron, copper, chrome, gold, nickel, potash, phosphate), port operations, smelting facilities, manufacturing of food and beverages, baggage handling, greenhouses, pharmaceutical and pulp and paper industries. Mr. Bechard also acts as an independent reviewer and lecturer. He holds a Bachelor's degree in Chemical Engineering and a Master's degree in Applied Mathematics from École Polytechnique de Montréal (Canada), in addition to being an independent graduated student in data science and machine learning at HEC Montréal (Canada). LinkedIn profile is www.linkedin.com/in/vincentbechard and email address is vbechard@difference-gcs.com .