Multi-target tree regression approach for surrogate-based optimisation

Artemis Tsochatzidi[1], Georgios I. Liapis[1], Francesca Cenci[2], Magdalini Aroniada[2], and Lazaros G. Papageorgiou[1]

[1] The Sargent Centre for Process Systems Engineering, Department of Chemical Engineering, UCL (University College London), Torrington Place, London WC1E 7JE, UK
[2] GlaxoSmithKline (GSK), Park Road, Ware SG12 0DP, UK
l.papageorgiou@ucl.ac.uk

Abstract. Industries are increasingly reliant on advanced process modeling techniques to improve development and operational efficiency. Utilising these models for optimisation holds the potential to significantly enhance performance, reduce costs, and ensure the highest standards of quality. However, when the underlying models become too complex or computationally expensive, surrogate-based optimisation offers a viable solution. In this work, we introduce a multi-target tree regression approach designed to address the complexities of multi-objective optimisation. The proposed methodology simultaneously handles multiple outputs, effectively captures nonlinear relationships, and enhances interpretability, making it a powerful tool for process optimisation. Additionally, we propose a novel methodology to mitigate the challenges of high dimensionality which is inherent in large datasets, enabling more efficient use of mathematical programming surrogates. By leveraging the developed methodologies, we aim to implement multi-objective optimisation to optimise key performance metrics like yield and purity in a real-world Active Pharmaceutical Ingredient Manufacturing Case Study, while deriving a Pareto curve to effectively illustrate the trade-off between competing objectives.

Keywords: Multi-target · Tree Regression · Process Optimisation · Mathematical Programming · Surrogate-based Optimisation.

1 Introduction

Advanced process modeling has transformed multiple industries, especially pharmaceuticals, by offering insights into complex systems, supporting informed decision-making, and improving process efficiency. Optimisation further enhances this by minimising resource use, reducing costs, and meeting sustainability goals. Traditional optimisation techniques, such as gradient-based methods, are limited by the complexity and computational demands of contemporary systems. Surrogate-based optimisation overcomes these limitations by employing simplified models to approximate complex behaviors, thereby reducing computational costs and time while improving decision-making [6].

At the same time, mathematical programming-based techniques have proven to be practical tools for developing various machine learning methods for small and medium-sized datasets [11]. Various tree-based surrogate modeling techniques have been explored in the literature, demonstrating effectiveness in addressing complex problems. A heuristic approach, the Classification and Regression Tree (CART), applies recursive binary partitioning to each node until no further splitting is possible or a specified termination criterion is met [3]. Regarding mathematical programming, Yang et al. [21] introduced MPtree, a regression tree algorithm that utilises OPLRA [20] to optimise binary node splitting. An enhanced version of this algorithm, known as StatTree, was later developed [7]. StatTree employs an optimisation model for data splitting and uses Chow statistical test to manage the structure of the tree. As a non-recursive approach, Optimal Regression Trees (ORT) are proposed, formulating the construction of the optimal decision tree as a discrete optimisation problem, enabling the entire tree to be built in a single step and identifying the tree that minimises the training error most effectively [2].

The concept of applying surrogate models in optimisation has been introduced in the literature [13], while early foundational work explored how individual surrogates can fit into large decision-making problems [9]. One of the most widely adopted techniques for optimising expensive simulations is Bayesian Optimisation, which typically employs Gaussian Processes as surrogate models. A key strength of this approach is its ability to quantify uncertainty, which allows for the use of an acquisition function to guide the sampling process [19].

Among the early strategies for optimising neural networks are big-M mixed-integer programming formulations specifically designed for ReLU activation functions [12, 5]. In terms of software implementations, mathematical programming formulations of gradient-boosted regression trees are available in the black-box optimiser ENTMOOT [18] and the Optimisation and Machine Learning Toolkit (OMLT), which integrates with the algebraic modeling language Pyomo [4].

Motivated by the need to optimise complex flowsheets while working with high-dimensional datasets, this study introduces a multi-target tree regression methodology to predict multiple performance metrics simultaneously, effectively capturing nonlinear relationships while maintaining interpretability. Moreover, a tailored optimisation approach is presented to handle the challenges of high-dimensional data. The mathematical model of the reverse optimisation is derived while the combined framework supports the generation of Pareto fronts to visualise trade-offs between competing objectives, facilitating informed decision-making in optimisation tasks. This approach enhances interpretability, computational efficiency, and practical applicabil ity, offering a powerful tool for decision-making.

2 Methodology

In this section, a novel approach of surrogate-based optimisation is presented. First, a Simultaneous Multi-tARget Tree regression ($SMART$) mathematical

model is presented, along with a methodology to mitigate the challenges of high dimensionality inherent in large datasets. This surrogate model builds upon concepts from existing regression tree formulations, such as those used in Optimal Regression Trees (ORT) [2, 1], extending them to handle multiple targets, along with additional enhancements to refine the leaf predictions. The complete Mixed-Integer Linear Programming (MILP) mathematical model is presented below.

Indices

m	Attribute ($m = m_1, m_2, ..., M$)
s	Sample ($s = s_1, s_2, ..., S$)
n, n'	Node ($n = n_1, n_2, ..., N$)
r	Response-target ($r = r_1, r_2, ..., R$)

Sets

NB	Branch nodes
NL	Leaf nodes
P_n	Parent node of node n
L_n	Set of ancestors of n whose left child has been encountered along the path from the root node to n
R_n	Set of ancestors of n whose right child has been encountered along the path from the root node to n

Parameters

A_{sm}	Value of sample s on attribute m
Y_{sr}	Real output value of sample s for response r
N_{min}	Minimum number of samples at each active leaf node
ϵ_m	Smallest non-zero difference between two adjacent values on attribute m
U, U'	Suitably large numbers

Binary Variables

d_n	1 if a split is applied at node n
W_{mn}	1 if a split is applied at node n using attribute m
E_{sn}	1 if sample s falls into leaf node n
Y_n	1 if leaf node n is active

Continuous Variables

b_n	Split point of node n
C_{mnr}	Regression coefficient for feature m in leaf node n for response r
$\hat{C}_{mnr}$	Regression coefficient of the quadratic term for feature m in leaf node n for response r
I_{nr}	Intercept of regression function in leaf node n for response r
Pd_{snr}	Predicted output for sample s in leaf node n for response r
D_{sr}	Absolute deviation between predicted output and real output for sample s for response r

Tree Structure Constraints

$$\sum_m W_{mn} = d_n \quad \forall n \in NB \tag{1}$$

$$0 \leq b_n \leq d_n \quad \forall n \in NB \tag{2}$$

$$d_n \leq d_{n'} \quad \forall n \in NB,\, n' \in P_n \tag{3}$$

$$\sum_m W_{mn'} \cdot A_{sm} \geq B_{n'} - U \cdot (1 - E_{sn}) \quad \forall s, n \in NL, n' \in R_n \tag{4}$$

$$\sum_m (W_{mn'} \cdot A_{sm} + \epsilon_m) \leq B_{n'} + U \cdot (1 - E_{sn}) \quad \forall s, n \in NL, n' \in L_n \tag{5}$$

Prediction Definition

$$Pd_{snr} = \sum_m C_{mnr} \cdot A_{sm} + \sum_m C_{1mnr} \cdot A_{sm}^2 + I_{nr} \quad \forall s, n, r \in NL \tag{6}$$

$$D_{sr} \geq Y_{sr} - Pd_{snr} - U' \cdot (1 - E_{sn}) \quad \forall s, n, r \in NL \tag{7}$$

$$D_{sr} \geq Pd_{snr} - Y_{sr} - U' \cdot (1 - E_{sn}) \quad \forall s, n, r \in NL \tag{8}$$

Logical Conditions

$$E_{sn} \leq Y_n \quad \forall s, n \in NL \tag{9}$$

$$\sum_s E_{sn} \geq N_{min} \cdot Y_n \quad \forall n \in NL \tag{10}$$

$$\sum_n E_{sn} = 1 \quad \forall s \tag{11}$$

Objective Function

$$\text{Min} \sum_{rs} D_{sr} \tag{12}$$

The initial constraints define the splits implemented at branch nodes. Binary variables W_{mn} are introduced to represent whether attribute m is utilised for a binary split at branch node n. Constraints (1) ensure that at most one attribute can be selected for splitting at each branch node, thereby enforcing univariate splits. If no attribute is selected for splitting at a branch node, the node becomes inactive and constraints (1) and (2) set variables d_n and b_n to zero. Constraints (3) prevent a branch node from applying a split if its parent node is not active for splitting [10].

Constraints (4) and (5) are introduced to enforce the splits applied by the branch nodes and ensure that samples follow the appropriate path to the leaf nodes. Specifically, constraints (4) model the path from branch node n' to leaf node n, where n' is an ancestor of n and its right child has been encountered along the path from the root node to n. Similarly, constraints (5) model the path

from branch node n' to leaf node n, where n' is an ancestor of n and its left child has been encountered along the path from the root node to n. A small constant ϵ_m representing the smallest non-zero difference between two adjacent values on the attribute m, is added to the left-hand-side. If $E_{sn} = 0$, the constraints become redundant, as the sufficiently large constant U ensures they are satisfied.

For any sample s and every response r, the training error is equal to the absolute deviation between the real output and the predicted output of the leaf node n where the sample belongs to (i.e $E_{sn} = 1$). It is expressed by equations (7) and (8). Through equations (6), for each leaf node n, a polynomial function of order 2 is employed to predict the value of samples for each response (P_{snr}). It is noted that although equations (6) are closed-form functions of the features, coefficients C_{mnr} and C_{1mnr} can take the value of zero, allowing for a flexibility in the predictions. Furthermore, constraints (9) state that if $Y_n = 0$, then a leaf node n cannot contain samples. Constraints (10) set a minimum number of samples N_{min} to fall in a leaf node, to avoid overfitting, while every sample can fall into only one leaf node, which is ensured by constraints (11). The objective function minimises the sum of absolute training errors and it's expressed by Equation (12). Figure 1 depicts how a sample s can end up in a leaf node by following either the right or the left branch of its ancestors, respecting the corresponding split.

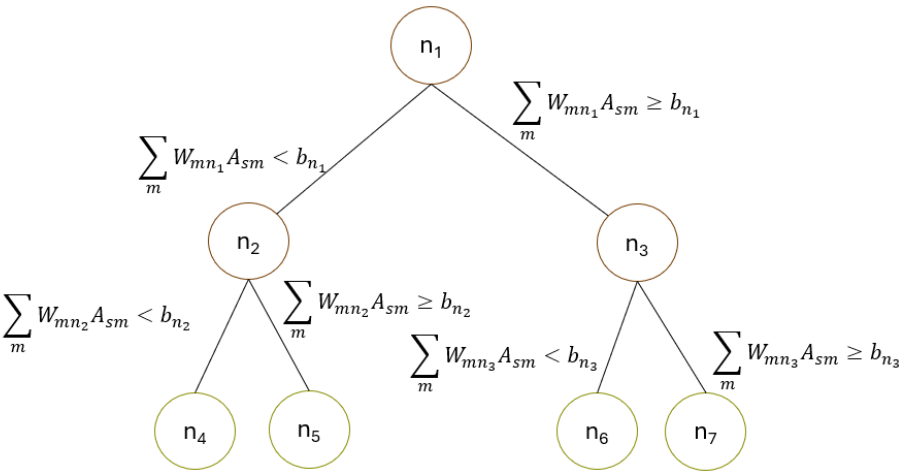

Fig. 1. Tree Visualisation

2.1 Proposed Methodology: Tailored Optimisation

In this approach, a single regression tree can effectively predict multiple outputs by leveraging both linear and quadratic terms of the predictors. This struc-

ture enables the tree to capture complex relationships while maintaining interpretability. However, as the complexity and the number of samples increase, mathematical programming approaches often struggle. To tackle the above in the case of $SMART$, the following methodology is developed.

Algorithm 1 Proposed Methodology

Require: Initial dataset D
1: Apply K-Medoids clustering.
2: Solve $SMART$ using the clustered dataset.
3: Fix W_{mn} and set b_n for initialisation.
4: Solve $SMART$ using all samples to minimise MAE.

As shown in Algorithm 1, the process begins with KMedoids clustering [15] applied to the dataset using the feature variables, reducing it to a representative subset in which only the cluster centers are kept [16], preserving the most essential patterns and variability. $SMART$ is then trained on this reduced dataset from which the optimal values of the key binary variables W_{mn} are obtained, along with the optimal values of the continuous variables B_n. The binary variables W_{mn} are then fixed, and the continuous variables B_n are set as initial values for the subsequent fitting of the full dataset to the $SMART$ model, ensuring a refined and computationally efficient fit. This second fitting is able to produce a feasible and good quality solution within the CPU limit applied.

The proposed tailored optimisation implies that the optimal binary decisions W_{mn} are inferred from a reduced representation of the dataset rather than the full information. In scenarios where computational resources are sufficient to handle the full dataset directly, bypassing clustering may allow for more precise and globally optimal decisions regarding W_{mn}, however, the methodology provides a pragmatic and effective alternative that unlocks the interpretability and representational capacity of $SMART$ under realistic constraints.

2.2 Reverse Optimisation

After the training, the fitted model information is extracted. It can then be used in the reverse optimisation step, where the optimum sample that optimises the objectives needs to be found. In this model, the variables of the previously presented mathematical model switch to parameters. More specifically, the continuous variables b_n, C_{mnr}, $C1_{mnr}$, and I_{nr} as well as the binary W_{mn} and Y_n become parameters, while the continuous variables Pd_{nr} for the predicted output per leaf node n and X_m for the optimal value of feature m are introduced. Binary variable E_n is added to decide if optimal sample falls into leaf node n.

The cardinality of the s index is equal to 1 as the goal is to derive the sample that optimises a key target while restricting other(s) (epsilon constraint [22]). For the case where r_1 is the key target and r_2 is the one restricted, the complete

Mixed-Integer Quadratically Constrained Programming (MIQCP) mathematical model follows.

$$\text{Max/Min} \sum_{n} Pd_{nr_1} \cdot E_n \tag{13}$$

$$\text{st.} \sum_{n} Pd_{nr_2} \cdot E_n \leq \epsilon \tag{14}$$

Equations (4),(5),(8),(9), and (11)

3 Computational Results

The developed methodology is designed to optimise real-world processes, with its effectiveness best demonstrated through application to a practical Case Study. The Case Study utilised features a system model of a drug substance manufacturing process stage, which includes a multi-phase batch reactor, a liquid-liquid extractor, and a crystallisation step. The model encompasses various process phenomena, mass and energy balances, and chemical reactions. Operating conditions such as temperatures or solution volumes may vary, resulting in 22 critical factors that can be optimised with respect to 2 target Quality Attributes, namely Yield and Impurity D. An uncertainty analysis was conducted for this flowsheet in gPROMS software [17], where the Monte Carlo method was applied with quasi-random Sobol sampling over 5000 uncertainty scenarios [14]. This generated a dataset of 5000 samples, referred to as the "Initial dataset D" in the proposed methodology.

To evaluate the developed methodology, a scenario was considered for the Case Study described. The goal is to maximise Yield while restricting Impurity D levels to maintain high purity. The dataset of 5,000 samples was analysed and the mean value of Impurity D was chosen as a benchmark, as it was observed that no samples achieved a Yield $>$ 96 % weight/weight (w/w) with Impurity D below the mean which is equal to the value of 0.0215 % w/w. The above illustrates the complexity of the optimisation problem, as improving Yield while constraining Impurity D to desired levels presents a significant challenge, with no solutions in the initial dataset.

Table 1 describes the optimally trained tree of depth 2 for the Case Study Scenario. As depicted in Figure 1, a sample progresses through the regression tree to arrive at a specific leaf node based on the splitting rules. Thus, the normalised splitting thresholds corresponding to the solution are summarised in Table 1. At each leaf node, both responses are predicted by distinct equations, enhancing the interpretability of the methodology. This structure allows for clear insight into the relationship between input features and outputs. The reverse optimisation process identifies the leaf node that best optimises the objectives and determines the corresponding optimal feature values.

Furthermore, adjusting the threshold for Impurity D, allows the exploration of a spectrum of possible outcomes (points in Figure 2) and the generation of a Pareto curve. This curve represents the trade-off frontier between competing objectives—in this case, maximising Yield while minimising Impurity D. By

Table 1. Rules of leaf nodes

Leaf node	Conditions
Leaf Node 4	$m_2 < 0.241$, $m_4 < 0.718$
Leaf Node 5	$m_2 < 0.241$, $m_4 \geq 0.718$
Leaf Node 6	$m_2 \geq 0.241$, $m_{22} < 0.119$
Leaf Node 7	$m_2 \geq 0.241$, $m_{22} \geq 0.119$

systematically varying the threshold for Impurity D and applying the developed methodology, we can obtain a series of optimisation results that reveal the maximum Yield that can be achieved at different impurity levels. This approach highlights how much one objective needs to be compromised to improve the other. Figure 2 presents the Pareto Curve generated through the multi-target tree regression approach for multi-objective optimisation applied to the Active Pharmaceutical Ingredient (API) Case Study. The points on the curve represent validated results obtained from the mechanistic model for various thresholds of Impurity D.

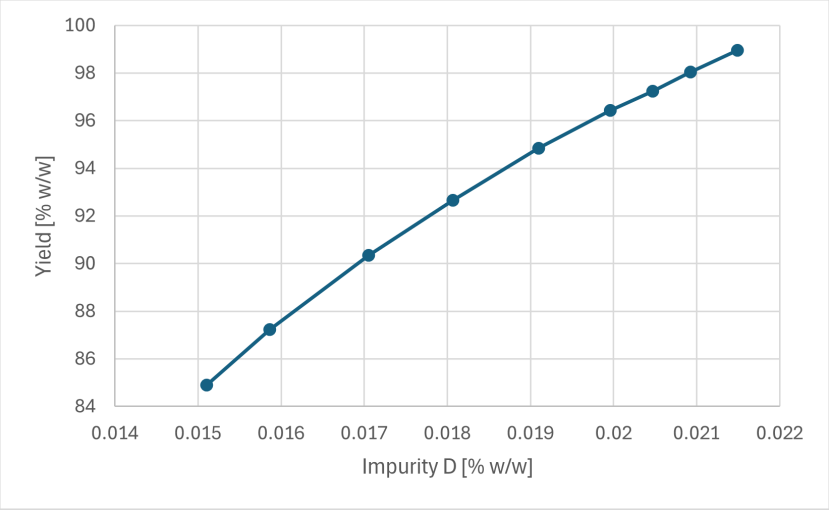

Fig. 2. Pareto Curve

Regarding computational details, the implementation was done in GAMSPy, while the mathematical programming optimisation problems presented are solved through Gurobi solver [8]. Notably, the initial MILP training, performed using the clustered dataset, is solved to optimality under 10 seconds, as is the MIQCP reverse optimisation step. As for the subsequent training using the entire dataset, a computational time limit of 600 seconds is imposed to balance computational efficiency with solution quality.

4 Concluding Remarks

This work introduced a multi-target tree regression methodology tailored for surrogate-based optimisation. The proposed mathematical programming techniques effectively capture non-linear relationships, handle datasets of high dimensionality, and provide interpretable predictions. The methodology has demonstrated its potential in optimising key performance metrics in the context of Active Pharmaceutical Ingredient manufacturing, while the Pareto curve generated serves as a valuable tool for understanding the limitations imposed by the process and allows decision-makers to select an optimal balance between trade-offs of competing objectives. In future work, we aim to explore the impact of deeper trees, and investigate the balance between error and complexity in order to avoid overffiting in this case.

Acknowledgment. The authors gratefully acknowledge the funding support from the industrial partner GlaxoSmithKline (GSK).

References

1. Ammari, B.L., Johnson, E.S., Stinchfield, G., Kim, T., Bynum, M., Hart, W.E., Pulsipher, J., Laird, C.D.: Linear model decision trees as surrogates in optimization of engineering applications. Computers & Chemical Engineering **178**, 108347 (2023). https://doi.org/https://doi.org/10.1016/j.compchemeng.2023.108347
2. Bertsimas, D., Dunn, J.: Machine learning under a modern optimization lens. Dynamic Ideas LLC (2019)
3. Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman and Hall/CRC, 1st edn. (1984), https://doi.org/10.1201/9781315139470
4. Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C.D., Misener, R.: Omlt: Optimization & machine learning toolkit. Journal of Machine Learning Research **23**(349), 1–8 (2022)
5. Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**, 296–309 (2018). https://doi.org/https://doi.org/10.1007/s10601-018-9285-6
6. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1), 50–79 (2009). https://doi.org/https://doi.org/10.1016/j.paerosci.2008.11.001
7. Gkioulekas, I., Papageorgiou, L.G.: Tree regression models using statistical testing and mixed integer programming. Computers & Industrial Engineering **153**, 107059 (2021). https://doi.org/https://doi.org/10.1016/j.cie.2020.107059
8. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023)
9. Henao, C.A., Maravelias, C.T.: Surrogate-based superstructure optimization framework. AIChE Journal **57**(5), 1216–1232 (2011). https://doi.org/10.1002/aic.12341
10. Liapis, G.I., Papageorgiou, L.G.: Optimisation-based classification tree: A game theoretic approach to group fairness. In: Dorronsoro, B., Zagar, M., Talbi, E.G. (eds.) Optimization and Learning. pp. 28–40. Springer Nature Switzerland, Cham (2025)

11. Liapis, G.I., Tsoka, S., Papageorgiou, L.G.: Interpretable optimisation-based approach for hyper-box classification. Machine Learning **114**, 51 (2025). https://doi.org/10.1007/s10994-024-06643-7
12. Lomuscio, A., Maganti, L.: An approach to reachability analysis for feed-forward relu neural networks (2017), arXiv preprint, arXiv:1706.07351 [cs.AI]
13. McBride, K., Sundmacher, K.: Overview of surrogate modeling in chemical process engineering. Chemie Ingenieur Technik **91**(3), 228–239 (2019). https://doi.org/10.1002/cite.201800091
14. Ning, J., and, H.T.: Randomized quasi-random sampling/importance resampling. Communications in Statistics - Simulation and Computation **49**(12), 3367–3379 (2020). https://doi.org/10.1080/03610918.2018.1547398
15. Park, H.S., Jun, C.H.: A simple and fast algorithm for k-medoids clustering. Expert Systems with Applications **36**(2, Part 2), 3336–3341 (2009). https://doi.org/https://doi.org/10.1016/j.eswa.2008.01.039
16. Sheng, W., Liu, X.: A genetic k-medoids clustering algorithm. Journal of Heuristics **12** (2006)
17. Siemens Process Systems Engineering: gproms | products (2022)
18. Thebelt, A., Kronqvist, J., Mistry, M., Lee, R.M., Sudermann-Merx, N., Misener, R.: Entmoot: A framework for optimization over ensemble tree models. Computers & Chemical Engineering **151**, 107343 (2021). https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107343
19. Triantafyllou, N., Lyons, B., Bernardi, A., Chachuat, B., Kontoravdi, C., Papathanasiou, M.M.: Comparative assessment of simulation-based and surrogate-based approaches to flowsheet optimization using dimensionality reduction. Computers & Chemical Engineering **189**, 108807 (2024). https://doi.org/10.1016/j.compchemeng.2024.108807
20. Yang, L., Liu, S., Tsoka, S., Papageorgiou, L.G.: Mathematical programming for piecewise linear regression analysis. Expert Systems with Applications **44**, 156–167 (2016). https://doi.org/https://doi.org/10.1016/j.eswa.2015.08.034
21. Yang, L., Liu, S., Tsoka, S., Papageorgiou, L.G.: A regression tree approach using mathematical programming. Expert Systems with Applications **78**, 347–357 (2017). https://doi.org/https://doi.org/10.1016/j.eswa.2017.02.013
22. Yang, Z., Cai, X., Fan, Z.: Epsilon constrained method for constrained multiobjective optimization problems: some preliminary results. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. p. 1181–1186. GECCO Comp '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2598394.2610012