Progressive Hedging for Optimization of Tree Ensembles as Objective Functions

Bashar L. Ammari,a Sergio Bugosen,a Jean-Paul Watson,b Emma S. Johnson,c Carl D. Laird,a

a Carnegie Mellon University

b Lawrence Livermore National Laboratory

c Sandia National Laboratories

bammari@andrew.cmu.edu

Abstract

Ensembles such as random forests and gradient boosted decision trees have become popular as surrogates in optimization problems. Large ensembles, however, may result in computationally impractical optimization problems. In this work, we adapt the parallelizable progressive hedging algorithm to accommodate ensembles of trees, leveraging mathematical equivalence in block-angular structures of stochastic programming and machine learning ensemble models. We study this algorithm on a large-scale example and show that there are computational benefits to utilizing progressive hedging to find high-quality heuristic solutions.

**Keywords**: Machine Learning, Optimization, Decomposition

* 1. Introduction

The operations research and engineering communities have increasingly adopted integrated machine learning (ML) and optimization in recent years. These communities have developed an extensive body of literature for ML models as surrogates within optimization problems ranging from security constrained alternating current optimal power flow (AC-OPF) for reliable electricity operation (Kilwein et al., 2021) to designing combination chemotherapy regimens to treat cancer (Bertsimas et al., 2016). Much of this adoption is not only driven by recent advancements in open-source ML modeling tools such as Pytorch, Tensorflow, and Scikit-learn, but by additional development of Python libraries such as the Optimization & Machine Learning Toolkit (OMLT) (Ceccon et al., 2022) that enable embedding trained ML models within optimization problems.

Tools such as OMLT that perform this embedding rely on the ability to formulate trained ML models as constraints in mathematical programs. That is, these tools take trained ML models, generate the algebraic expressions required to represent such models, and embed them within broader optimization problems coded within an algebraic modeling language such as Pyomo (Bynum et al., 2021). For example, several authors have proposed formulations for decision trees and ensembles of decision trees. Biggs et al. (2018), Mišić (2020), and Mistry et al. (2021) present mixed-integer linear programming formulations of ensembles of standard decision trees such as random forests (RFs) and gradient-boosted decision trees (GBDTs). Recently, Ammari et al. (2023) explored extensions of standard trees called linear model decision trees and showed how to formulate them as mixed-integer linear programs and mixed-integer quadratic programs.

Although the ML community has shown ensembles such as RFs and GBDTs to be effective models (Caruana and Niculescu-Mizil, 2006), embedding large ensembles in broader decision-making problems may lead to computationally intractable optimization problems. We note that, like stochastic programming problems, these problems often contain a block-angular structure. Therefore, decomposing these large problems into their individual trees may be computationally advantageous. In this work, we use progressive hedging (Rockafellar and Wets, 1991) as a decomposition strategy for optimization over large GBDTs and RFs embedded as objective functions to provide good heuristic solutions with faster computational performance compared with the direct solution of the full-space model. This algorithm is implemented in mpi-sppy, a Python package for parallel decomposition of stochastic programs (Knueven et al., 2023). Although this work does not consider stochastic programming problems, we are interested in solving problems of the form Eq. (1) – Eq. (3).

|  |  |
| --- | --- |
| min | (1) |
| s.t. | (2)  (3) |

To state the problem explicitly, we are optimizing the output, , of a tree ensemble, , subject to constraints on the inputs to the tree ensemble, . Specifically, we are interested in the case where is a large tree ensemble. Biggs et al. (2017) previously studied this problem for RFs as objective functions and proposed Bender’s decomposition to improve solution time over the direct approach. We extend this work by applying progressive hedging (PH) to get good heuristic solutions to Eq. (1) – Eq. (3). While this methodology is heuristic and does not come with optimality guarantees, several algorithmic advances have resulted in successful application of progressive hedging to solve mixed-integer programming problems (Watson and Woodruff, 2011).

The remainder of this paper is structured as follows. In Section 2, we review the relevant generalized disjunctive programming (GDP) formulation for individual trees used in this work. Furthermore, we draw analogies to the extensive form (EF) of a stochastic program and present an EF of the problem described by Eq. (1) – Eq. (3). In addition, we introduce the PH algorithm to accommodate GBDTs and RFs and present the computational environment and example problem used to compare performance. In Section 3, we compare the computational time and solutions using both PH and direct solution of the EF. We discuss concluding remarks and future avenues of research in Section 4.

* 1. Overview of Methods

*2.1 Formulation for Embedding Individual Trees*

When decomposing an ensemble into its individual trees, we can apply the GDP formulation presented in Ammari et al. (2023) for a single linear model decision tree, by using constant values at the leaf nodes. This formulation is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ˅ |  |  |  |  |
|  |  | (4) |
|  |  |  |
| exactly\_one{ | |  |  | (5) |
|  | |  |  | (6) |
|  | |  |  | (7) |
|  | |  |  | (8) |

The Boolean variable indicates which leaf is selected, implying the bound constraints and the constant value that are enforced. The bound vectors and are calculated by traversing the tree and finding the tightest lower and upper bounds taken from the tree's splitting thresholds. Constraint Eq. (5) and the use of the “or” operator, ˅, enforces selection of only one disjunct per disjunction (i.e., exactly one leaf is returned by the tree). To acquire a mixed-integer programming representation of this disjunctive formulation, we apply a Big-M transformation on the disjuncts as is common in GDP literature.

*2.2 Progressive Hedging*

Progressive Hedging (PH) is an effective algorithm for solving stochastic optimization problems. In this work, we apply PH to find good heuristic solutions for large tree ensembles that are otherwise prohibitively computationally expensive or intractable. In this section, we review the PH algorithm, and show how this algorithm can accommodate ensembles of trees.

Many stochastic optimization problems are formulated as two-stage problems. There are “here-and-now” decisions in the first stage (variables ) and “wait-and-see” decisions in the second stage (variables ). That is, a modeler may identify a finite set of realizations of the second stage (called scenarios), , and their probability of occurrence, . The objective, Eq. (9), is to minimize the cost associated with making those first stage decisions and the expected value of the cost over all scenario realizations, while ensuring that any first and second stage decision will satisfy constraints in Eq. (10).

|  |  |  |
| --- | --- | --- |
| min |  | (9) |
| s.t. |  | (10) |

To extend this idea to ensembles of trees, we treat each tree, . as a scenario. The input variables to the tree, will be analogous to our “first stage variables,” and the output of each tree as well as the binary variables associated with correct selection of the leaves (this variable is introduced when Eq. (4) – Eq. (8) is transformed), will be our analog to the “second stage variables.” If we reconsider Eq. (1) – Eq. (3), we can rewrite that optimization problem equivalently as follows.

|  |  |  |
| --- | --- | --- |
| min |  | (11) |
| s.t. |  | (12) |
|  |  | (13) |

The output of an RF is the average of the outputs of the individual trees and therefore our objective is given in Eq. (11). The constraints on the input given by Eq. (12) remain the same, however we embed each tree, using the GDP formulation in Eq. (13). We refer to this problem as the extensive form (EF) representation due to its analog to the extensive form of a stochastic programming problem. Note that the output of a GBDT is the sum of the individual trees, and the objective function is easily modified to accommodate this.

Given the equivalent formulation of Eq. (1) – Eq. (3) described by Eq. (11) – Eq. (13), the PH algorithm for tree ensembles as objective functions can be written as follows.

|  |
| --- |
| **Algorithm 1: Progressive Hedging for Tree Ensembles as Objective Functions** |
| 1. := 0  2. For all  {  3. :=  4. For all    5.  6. For all  }  7.  8. For all    9. Terminate if criterion is met, otherwise go to step 5. |

As explained in Knueven et al. (2023), there are many possible termination criteria in Step 9 of Algorithm 1 including iteration limits, primal convergence, primal-dual convergence, or convergence of . Selection of this criterion is often left for the user and can be problem dependent. Note that steps 2, 4, 6, and 8 are straightforwardly parallelizable which is a key advantage of utilizing PH. In this work, we show that with a fixed number of PH iterations, and the addition of an *incumbent finder* (Knueven et al., 2023), we can acquire good heuristic solutions with faster solution times when compared to the direct solution of the EF. Solution quality and computational benefit also depend on proper selection of values, and automated update schemes are available in mpi-sppy. See Watson and Woodruff (2011) for additional information on calculations.

*2.3 Case Study and Computational Setup*

We use the *ex2\_1\_5* instance from MINLPLib.org. The objective is a nonlinear function, all the constraints are linear, and all the variables are continuous and bounded between 0 and 1. We replace the objective function by sampling the input variables randomly between their bounds, evaluating the objective function, and training a random forest on the resulting data (i.e. the features are the sampled input variables and the label is the value of the nonlinear objective expression evaluated at each sampled point). We fix the maximum depth of the individual trees in the RF to 10 layers and increase the size of the ensemble by adding more trees. This enables comparison of the different solution approaches at increasing scales. To train the RFs, we use Scikit-learn 1.2.2 and Python 3.11.3. We embed each trained RF using Big-M transformations of the individual tree formulation from Eq. (4) – Eq. (8). We use Pyomo 6.6.2 and its extension, Pyomo.GDP.

To utilize the PH algorithm, we use mpi-sppy. In addition, mpi-sppy can generate the EF given by Eq. (11) – Eq. (13) automatically using the same workflow required for utilizing the PH algorithm. For each trained RF, we fix the number of PH iterations to 20. In addition, we utilize the *xhatshuffle* incumbent finder, as well the dynamic update capabilities of the *norm-rho-updater,* both of which are available in the mpi-sppy library. We utilize 30 processors, openmpi 4.0.5, and mpi4py 3.1.4 to take advantage of the parallelizability of the PH algorithm. The computational studies were performed on a Linux server running Ubuntu with 1TB of RAM and 4 Intel(R) Xeon(R) Gold 6234 CPUs (3.30GHz) with 8 cores each. The EF solution and the PH algorithm both use Gurobi 10.0.4 as the optimization solver with 8 solver threads.

* 1. Results

Results for the computational performance are shown in Table 1. With 20 iterations of PH, the best incumbents found are variable, although all are within 2% of the true solution. In certain cases, these PH solutions are almost exactly the true optima. It is again important to note that convergence to the true optimum is not guaranteed due to limitations of PH on second stage binary variables. However, we can see that the PH algorithm scales better than the EF method and can provide high-quality solutions. This improved performance is more evident with larger ensembles. Although not shown here, there is an additional computational benefit to using PH since model generation is also parallelized and the Gurobi-Persistent interface in Pyomo eliminates the need to generate new Pyomo models at each iteration of the PH algorithm.

**Table 1.** Computational performance results on MINLPLib.org example ex2\_1\_5. *Num Trees* indicates the number of trees in the RF. *EF Time* is the solution time using the extensive form in mpi-sppy, and *PH Time* using progressive hedging. *True Obj* is the objective value from the optimal solution of EF. *PH Obj* is the best incumbent after 20 iterations of PH. *% Difference* compares these two objective values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Num Trees** | **EF Time**  **[sec]** | **PH Time**  **[sec]** | **True Obj** | **PH Obj** | **% Difference** |
| 30 | 26.82 | 21.87 | -8.917 | -8.756 | 1.81 |
| 50 | 69.93 | 40.87 | -5.323 | -5.276 | 0.88 |
| 70 | 81.62 | 48.95 | -3.814 | -3.764 | 1.31 |
| 90 | 128.5 | 59.38 | -2.942 | -2.928 | 0.48 |
| 110 | 203.4 | 74.92 | -2.408 | -2.379 | 1.20 |
| 130 | 210.6 | 89.18 | -2.044 | -2.042 | 0.01 |
| 150 | 294.0 | 98.55 | -1.772 | -1.758 | 0.79 |

* 1. Conclusions

Tree ensembles such as random forests (RFs) have become popular machine learning models. In the framework of mathematical optimization, these RFs are mixed-integer programming representable, which has driven increased adoption of these models as surrogates in the engineering community. However, large ensembles may result in computationally intractable optimization problems. Our contributions include using progressive hedging (PH) as an approach to acquire good heuristic solutions with improved solution times. We introduced how to apply the PH algorithm to accommodate ensembles, and results showed that PH scales better than the extensive form solution as problem size increases. Future work remains to determine whether we can acquire better solutions or optimal solutions in comparable computational time. Additionally, other decomposition strategies such as Lagrangian relaxation may be future research directions.

Disclaimer

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the Department of Energy’s Office of Electricity’s Advanced Grid Modelling (AGM) program. This work was also funded by Sandia National Laboratories Laboratory Directed Research and Development (LDRD) program. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.

References

Ammari, B. L., Johnson, E. S., Stinchfield, G., Kim, T., Bynum, M., Hart, W. E., Pulsipher, J., and Laird, C. D. (2023). “Linear model decision trees as surrogates in optimization of engineering applications.” Computers & Chemical Engineering, 178:108347.

Bertsimas, D., O’Hair, A., Relyea, S., and Silberholz, J. (2016). “An analytics approach to designing combination chemotherapy regimens for cancer.” Management Science, 62:1511–1531.

Biggs, M., Hariss, R., and Perakis, G. (2017). “Optimizing objective functions determined from random forests.” SSRN Electronic Journal.

Birge, J. R. and Louveaux, F. (2011). “Introduction to Stochastic Programming.” Springer New York.

Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P., and Woodruff, D. L. (2021). “Pyomo — Optimization Modeling in Python”, volume 67. Springer International Publishing, 3 edition

Caruana, R. and Niculescu-Mizil, A. (2006). “An empirical comparison of supervised learning algorithms.” pages 161–168. ACM Press.

Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C. D., and Misener, R. (2022). “OMLT: Optimization & machine learning toolkit.” Journal of Machine Learning Research, 23:1–8.

Kilwein, Z., Boukouvala, F., Laird, C., Castillo, A., Blakely, L., Eydenberg, M., Jalving, J., and Batsch-Smith, L. (2021). “Ac-optimal power flow solutions with security constraints from deep neural network models.” Computer Aided Chemical Engineering, 50:919–925

Knueven, B., Mildebrath, D., Muir, C., Siirola, J.D., Watson, J.-P., and Woodruff, D. L. (2023), “A parallel hub-and-spoke system for large-scale scenario-based optimization under uncertainty,” Mathematical Programming Computation, pp. 1–29.

Mistry, M., Letsios, D., Krennrich, G., Lee, R. M., and Misener, R. (2021). “Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded.” INFORMS Journal on Computing, 33:1103–1119.

Mišić, V. V. (2020). Optimization of tree ensembles. Operations Research, 68:1605–1624.

Rockafellar, R. and Wets, R. J.-B. (1991). “Scenarios and policy aggregation in optimization under uncertainty.” Mathematics of Operations Research, 16:119–147.

Watson, J.-P. and Woodruff, D. L. (2011). “Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems.” Computational Management Science, 8:355–370.