7th ICML Workshop on Automated Machine Learning (2020)

Solving Constrained CASH Problems with ADMM

P. Ram P.RAM@GATECH.EDU
S. Liu SIJIA.LIU@IBM.COM
D. Vijaykeerthi DEEPAKVIJ@IN.IBM.COM
D. Wang DAKUO.WANG@IBM.COM
D. Bouneffouf DJALLEL.BOUNEFFOUF1@IBM.COM
G. Bramble GBRAMBLE@US.IBM.COM
H. Samulowitz SAMULOWITZ@US.IBM.COM
A. G. Gray ALEXANDER.GRAY@IBM.COM
IBM Research

Abstract

The CASH problem has been widely studied in the context of automated configurations of machine learning (ML) pipelines and various solvers and toolkits are available. However, CASH solvers do not directly handle black-box constraints such as fairness, robustness or other domain-specific custom constraints. We present our recent approach (Liu et al., 2020) that leverages the ADMM optimization framework to decompose CASH into multiple small problems and demonstrate how ADMM facilitates incorporation of black-box constraints.

1. Automated ML Pipeline Configuration

Hyper-parameter optimization (HPO) for a *single* machine learning (ML) algorithm is widely studied in AutoML (Snoek et al., 2012; Shahriari et al., 2016). HPO was generalized to the **C**ombined **A**lgorithm **S**election and **H**PO (CASH) problem to configure multiple stages of a ML pipeline (transformers, feature selectors, predictive models) automatically (Thornton et al., 2012; Feurer et al., 2015). Since the CASH formulation, various innovative solvers have been proposed (Hutter et al., 2011; Bergstra et al., 2011; Mohr et al., 2018; Rakotoarison et al., 2019). CASH has two main challenges: (i) the tight coupling between the algorithm selection & HPO; and (ii) the black-box nature of optimization objective lacking any explicit gradients – feedback is only available in the form of (often expensive) function evaluations. Furthermore, the CASH formulation does not explicitly handle constraints on the individual ML pipelines such as fairness or domain-specific constraints on individual ML pipelines.

We view CASH as a mixed integer black-box nonlinear program and we recently proposed a novel solution framework leveraging the *alternating direction method of multipliers* (ADMM) framework (Liu et al., 2020). ADMM offers a *two-block* alternating optimization procedure that splits an involved problem (with multiple variables & constraints) into simpler (often unconstrained) sub-problems (Boyd et al., 2011; Parikh and Boyd, 2014).

Contributions. We utilize ADMM to decompose CASH into 3 problems: (i) a black-box optimization with a *small* set of *only continuous* variables, (ii) a closed-form Euclidean projection onto an integer set, and (iii) a black-box integer program. Moreover, the ADMM framework handles *any black-box constraints alongside the black-box objective* – such constraints are seamlessly incorporated while retaining very similar sub-problems.

1.1 Related work

Beyond grid-search and random search (Bergstra and Bengio, 2012) for HPO, sequential model-based optimization (SMBO) is a common technique with different 'surrogate models' such as Gaussian processes (Snoek et al., 2012), random forests (Hutter et al., 2011) and tree-parzen estimators (Bergstra et al., 2011). Successive halving (Jamieson and Talwalkar, 2016; Sabharwal et al., 2016) and HyperBand (Li et al., 2018) utilize computationally cheap *multi-fidelity* approximations of the objective based on some budget (training samples/epochs) with bandit learning to eliminate unpromising candidates early. These schemes essentially perform an efficient random search over discrete or discretized continuous spaces. BOHB (Falkner et al., 2018) combines SMBO (with TPE) and HyperBand. Meta-learning (Vanschoren, 2018; Fusi et al., 2018; Drori et al., 2018) leverages past experiences with search space refinements and promising starting points. SMBO with a large number of variables along with conditional dependencies has been used to solve CASH in the widely-used Auto-WEKA (Thornton et al., 2012; Kotthoff et al., 2017) and Auto-sklearn (Feurer et al., 2015) toolkits. Both apply the general purpose SMAC framework (Hutter et al., 2011) to find optimal ML pipelines of fixed shape. Hyperopt-sklearn (Komer et al., 2014) utilizes TPE as the SMBO.

SMBO-based CASH has been improved by partially splitting CASH and taking advantage of the structure in the algorithm selection problem. ML-Plan (Mohr et al., 2018) uses hierarchical task networks (HTN) planning for algorithm selection and randomized search for HPO, while MOSAIC (Rakotoarison et al., 2019) utilizes Monte-Carlo Tree Search (MCTS) and Bayesian Optimization (BO) respectively. The two sub-problems are coupled with a shared surrogate model. Even though algorithm selection and HPO are solved in independent steps, MOSAIC still requires computations (such as the acquisition function estimation/optimization and the surrogate-model training) over the high-dimensional joint search-space of all algorithms and hyper-parameters (HPs). Our proposed ADMM-based solution performs an explicit primal-dual decomposition, significantly reducing the dimensionality of each of the aforementioned sub-problems. Moreover, our solution presents a general framework that can incorporate HTN-Planning and MCTS based algorithm selection.

ADMMBO (Ariafar et al., 2019) for general BO with black-box constraints maintains a separate surrogate model for the objective and each of the black-box constraints (a total of $M+1$ surrogate models for M constraints) and utilizes the ADMM framework to split the constrained optimization into a sequence of unconstrained problems to minimize the objective and constraint violations. The sub-problem dimensionality remains the same as the original problem. Our proposed ADMM based scheme for CASH with black-box constraints takes advantage of the problem structure to handle M black-box constraints by adding M variables to one of the three sub-problems without needing any additional surrogate models.

2. CASH as a Mixed Integer Nonlinear Program

We focus on CASH for a *fixed pipeline shape* – for N *functional modules* with K_i choices in each module, let $\mathbf{z}_i \in \{0,1\}^{K_i}$ denote the algorithm choice in module i, with the constraint $\mathbf{1}^\top \mathbf{z}_i = 1$ ensuring a single choice per module. Let $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$. Assuming that categorical HPs can be encoded as integers, let $\boldsymbol{\theta}_{ij}$ be the HPs of algorithm j in module i ($\boldsymbol{\theta}^c_{ij} \in \mathcal{C}_{ij} \subset \mathbb{R}^{m^c_{ij}}$ *continuous* and $\boldsymbol{\theta}^d_{ij} \in \mathcal{D}_{ij} \subset \mathbb{Z}^{m^d_{ij}}$ *integer*). With $\boldsymbol{\theta}^c = \left\{\boldsymbol{\theta}^c_{ij}, \forall i \in [N], j \in [K_i]\right\}$, and $\boldsymbol{\theta}^d$ defined

analogously, let $f(\mathbf{z}, \{\theta^c, \theta^d\})$ represent the loss of a ML pipeline configured with algorithm choices $\mathbf{z}$ and the HPs $\{\theta^c, \theta^d\}$. CASH can be stated as[1]:

$$\min_{\mathbf{z}, \theta^c, \theta^d} f(\mathbf{z}, \{\theta^c, \theta^d\}) \text{ subject to } \begin{cases} \mathbf{z}_i \in \{0,1\}^{K_i}, \mathbf{1}^\top \mathbf{z}_i = 1, \forall i \in [N], \\ \theta^c_{ij} \in \mathcal{C}_{ij}, \theta^d_{ij} \in \mathcal{D}_{ij}, \forall i \in [N], j \in [K_i]. \end{cases} \tag{CASH}$$

We may need the ML pipelines to also *explicitly satisfy* application specific *constraints* corresponding to M general *black-box functions* with no analytic form in $(\mathbf{z}, \{\theta^c, \theta^d\})$:

$$g_m(\mathbf{z}, \{\theta^c, \theta^d\}) \leq \epsilon_m, m \in [M]. \tag{1}$$

For example, deployment constraints may require prediction latency below a threshold. Business constraints may require highly accurate pipelines with explicitly bounded false positive rate – false positives may deny loans to eligible applicants, which is lost business and could violate anti-discriminatory requirements. Pursuing *fair* AI, regulators may require that any deployed ML pipeline explicitly satisfies bias constraints (Friedler et al., 2019).

3. Decomposing CASH with ADMM

Introducing a surrogate objective $\widetilde{f}$ over the continuous domain $(\mathcal{C}_{ij} \times \widetilde{\mathcal{D}}_{ij}), i \in [N], j \in [K_i]$ that matches the objective f over $(\mathcal{C}_{ij} \times \mathcal{D}_{ij})$ with $\widetilde{\mathcal{D}}_{ij}$ as the continuous relaxation of the integer space $\mathcal{D}_{ij}$, we follow ADMM, detailed in Appendix C, to decompose (CASH) into the following 3 sub-problems to be solved iteratively, with (t) representing the ADMM iteration index, ρ as the penalty for the augmented Lagrangian term and the Lagrangian multipliers $\boldsymbol{\lambda}$ updated as $\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \rho(\widetilde{\theta}^{d(t+1)} - \boldsymbol{\delta}^{(t+1)})$:

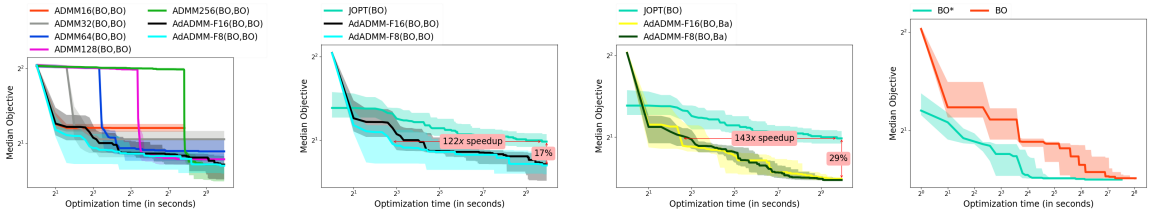
$$\left\{\theta^{c(t+1)}, \widetilde{\theta}^{d(t+1)}\right\} = \underset{\theta^c_{ij}, \widetilde{\theta}^d_{ij}}{\arg\min} \widetilde{f}\left(\mathbf{z}^{(t)}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) + \frac{\rho}{2} \left\|\widetilde{\theta}^d - \mathbf{b}\right\|_2^2, \mathbf{b} := \boldsymbol{\delta}^{(t)} - \frac{1}{\rho}\boldsymbol{\lambda}^{(t)}, \tag{θ-min}$$

$$\boldsymbol{\delta}^{(t+1)} = \underset{\boldsymbol{\delta}_{ij} \in \mathcal{D}_{ij}}{\arg\min} \|\mathbf{a} - \boldsymbol{\delta}\|_2^2, \quad \mathbf{a} := \widetilde{\theta}^{d(t+1)} + (1/\rho)\boldsymbol{\lambda}^{(t)}, \tag{δ-min}$$

$$\mathbf{z}^{(t+1)} = \underset{\mathbf{z}_i \in \{0,1\}^{K_i}, \mathbf{1}^\top \mathbf{z}_i = 1}{\arg\min} \widetilde{f}\left(\mathbf{z}, \left\{\theta^{c(t+1)}, \widetilde{\theta}^{d(t+1)}\right\}\right). \tag{$\mathbf{z}$-min}$$

Solving (θ-min)**.** This is a continuous black-box optimization problem with variables θ^c and $\widetilde{\theta}^d$. Since the algorithms $\mathbf{z}^{(t)}$ are fixed in (θ-min), $\widetilde{f}$ only depends on the HPs of the chosen algorithms – the *active variable set* $S = \{(\theta^c_{ij}, \widetilde{\theta}^d_{ij}) : z_{ij}{}^{(t)} = 1\}$. This splits ($\theta$-min) even further into (i) $\min_{\widetilde{\theta}^d_{ij} \in \widetilde{\mathcal{D}}_{ij}} \|\widetilde{\theta}^d_{ij} - \mathbf{b}_{ij}\|_2^2$ with $z_{ij} = 0$ (the *inactive* set) requiring a Euclidean projection of $\mathbf{b}_{ij}$ onto $\widetilde{\mathcal{D}}_{ij}$, and (ii) a black-box optimization with a *small* active continuous variable set[2] S. Solvers such as BO (Shahriari et al., 2016), direct search (Larson et al., 2019), or trust-region based derivative-free optimization (Conn et al., 2009) work fairly well.

1. The above formulation allows for easily extending the search to more flexible pipelines (Appendix A). To extensively evaluate our proposed CASH solver with multiple repetitions and restarts, we also propose a novel cheap-to-evaluate black-box objective (Appendix B) that possesses the structure of (CASH).
2. For the CASH problems we consider in our empirical evalutations, $|\theta| = |\theta^c_{ij}| + |\widetilde{\theta}^d_{ij}| \approx 100$ while the largest possible active set S is less than 15 and typically less than 10.



(a) Fixed vs. Adaptive (b) Split vs. Joint (c) Custom solvers (d) Cold vs. Warm

Figure 1: Performance of ADMM and enhancements on the artificial black-box objective (Appendix B) with the incumbent objective (median and inter-quartile range over 30 trials) on the vertical axis and optimization time on the horizontal axis. Note the logscale on both axes.

Solving ($\boldsymbol{\delta}$-min)**.** This requires an elementwise minimization $\min_{\boldsymbol{\delta}_{ij}} (\boldsymbol{\delta}_{ij} - \mathbf{a}_{ij})^2$ and solved in closed form by projecting $\mathbf{a}_{ij}$ onto $\widetilde{\mathcal{D}}_{ij}$ and then rounding to the nearest integer in $\mathcal{D}_{ij}$.
Solving z-min. The following black-box discrete optimizers can be utilized: (i) *MCTS* can be used as in Rakotoarison et al. (2019), (ii) *HTN-Planning* as in Mohr et al. (2018) (iii) *Multi-fidelity evaluations* can be used with successive halving (Jamieson and Talwalkar, 2016; Li et al., 2018) or incremental data allocation (Sabharwal et al., 2016), (iv) Interpreting ($\mathbf{z}$-min) as a *combinatorial multi-armed bandits* problem and utilizing Thompson sampling (Durand and Gagné, 2014) as we did in Liu et al. (2020, Appendix 4).

3.1 Empirical Advantages of ADMM

ADMM has been used for both convex (Boyd et al., 2011) and non-convex optimization (Xu et al., 2016) and various ADMM enhancements have been proposed. We demonstrate how these improve ADMM for (CASH). For initial evaluations, we use the artificial objective (Appendix B) for cheap evaluations, which allow us to efficiently generate statistically significant results over multiple trials. A maximum of 100 ADMM iterations is executed with $\rho = 1$ (ADMM for CASH appears to be stable with respect to ρ; see Appendix D) for a maximum runtime of 2^{10} seconds. We consider a search space of 4 modules with 8 scalers, 11 transformers, 7 feature selectors and 11 estimators (total 6776 algorithm combinations with almost 100 HPs). See Liu et al. (2020, Appendix 7) for complete details.
Adaptive precision. ADMM progresses by iteratively solving ($\boldsymbol{\theta}$-min) and ($\mathbf{z}$-min) (($\boldsymbol{\delta}$-min) is solved in closed form). In practice, the sub-problems are often solved to lower precision in the initial ADMM iterations; the precision is progressively increased through the ADMM iterations (*adaptive precision*) instead of maintaining the same precision for all ADMM iterations (*fixed precision*). In Figure 1a, we use Bayesian Optimization (BO) to solve both ($\boldsymbol{\theta}$-min) & ($\mathbf{z}$-min). The precision of the sub-problems are controlled by modifying the number of BO iterations. For fixed precision ADMM, we fix the BO iterations for the sub-problems to $I = \{16, 32, 64, 128, 256\}$ denoted as ADMM16(BO,BO) and so on. For adaptive precision ADMM, we initially solve each sub-problem with 16 BO iterations and progressively increase the number of BO iterations to 256 with an additive factor of $F = \{8, 16\}$ in each ADMM iteration denoted by AdADMM-F8(BO,BO) & AdADMM-F16(BO,BO) respectively. We see the expected behavior – fixed precision ADMM with small I dominate for small time scales but saturate soon; large I require significant start-up time but dominate for larger time scales. Adaptive precision ADMM provides best anytime performance.

Solving smaller sub-problems. In Figure 1b, we demonstrate the advantage of solving multiple smaller sub-problems over solving a single large joint problem. We use BO to solve both the joint problem (JOPT(BO)) and the sub-problems in the adaptive ADMM (AdADMM-F8(BO,BO) & AdADMM-F16(BO,BO)). ADMM demonstrates 120× speedup to reach the best objective obtained by JOPT(BO) within the time budget, and also provides an additional 17% improvement over that objective at the end of the time budget.
Custom sub-problem solvers. BO is designed for problems with (a small number of) continuous variables and hence is well-suited for (θ-min); it is not designed for ($\mathbf{z}$-min). We should instead use schemes customized for ($\mathbf{z}$-min) such as MCTS (Rakotoarison et al., 2019) or Thompson sampling for combinatorial multi-armed bandits (Liu et al., 2020, Appendix 4). In Figure 1c, we consider BO for (θ-min) and bandits for ($\mathbf{z}$-min) – AdADMM-F8(BO,Ba) & AdADMM-F16(BO,Ba) – which further improves over adaptive ADMM, demonstrating 140× speedup over JOPT(BO) with an additional 29% improvement in the objective value.
Warm Start ADMM. It is common in ADMM to *warm-start* the sub-problem minimizations in any ADMM iteration with the solution of the same sub-problem from previous ADMM iterations (if available) to improve empirical convergence. In adaptive precision ADMM, we employ warm-starts for BO in (θ-min) to get BO* and compare it to BO with cold-starts. Figure 1d indicates that while BO and BO* converge to the same objective at the end of the time budget, BO* has significantly better anytime performance.
Evaluation on OpenML data sets. We evaluate the performance of ADMM against JOPT(BO) for (CASH) on 8 OpenML data sets in Appendix E and see over 10× speedup in most cases and over 10% improvement in the final objective in many cases. We refer readers to Liu et al. (2020) for the detailed comparison of ADMM to Auto-sklearn (Feurer et al., 2015) and TPOT (Olson and Moore, 2016) across 30 classification data sets.

4. CASH with Black-box Constraints

We consider (CASH) in the presence of black-box constraints (1). Without loss of generality, let $\epsilon_m \geq 0$ for $m \in [M]$. With scalars $u_m \in [0, \epsilon_m]$, we reformulate the inequality constraint (1) as an equality constraint $g_m(\mathbf{z}, \{\theta^c, \theta^d\}) = \epsilon_m - u_m$ and a box constraint $u_m \in [0, \epsilon_m]$.

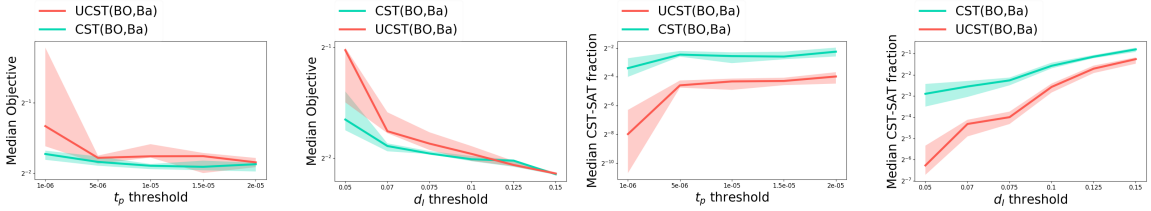
Introducing (i) surrogate $\widetilde{g}_m$ for each black-box function $g_m, m \in [M]$ over the continuous domain in a manner similar to $\widetilde{f}$ in Section 3, (ii) new Lagrangian multipliers $\mu_m, m \in [M]$ for each of the M black-box constraints, and following the ADMM mechanics (detailed in Appendix F), we decompose constrained (CASH) into the following unconstrained problems:

$$\min_{\theta^c_{ij}, \widetilde{\theta}^d_{ij}, u_m} \widetilde{f}(\mathbf{z}^{(t)}, \{\theta^c, \widetilde{\theta}^d\}) + \frac{\rho}{2}[\|\widetilde{\theta}^d - \mathbf{b}\|_2^2 + \sum_{m=1}^{M} [\widetilde{g}_m(\mathbf{z}^{(t)}, \{\theta^c, \widetilde{\theta}^d\}) - \epsilon_m + u_m + \frac{1}{\rho}\mu_m{}^{(t)}]^2], \quad (2)$$

$$\min_{\mathbf{z}} \widetilde{f}(\mathbf{z}, \{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\}) + \frac{\rho}{2} \sum_{i=1}^{M} [\widetilde{g}_m(\mathbf{z}, \{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\}) - \epsilon_m + u_m{}^{(t+1)} + \frac{1}{\rho}\mu_m{}^{(t)}]^2, \quad (3)$$

with $\mathbf{b}$ defined in (θ-min), δ updated as per (δ-min), and $\mu_m, m \in [M]$ updated as $\mu_m{}^{(t+1)} = \mu_m{}^{(t)} + \rho(\widetilde{g}_m(\mathbf{z}^{(t+1)}, \{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\}) - \epsilon_m + u_m{}^{(t+1)})$.

Problem (2) is black-box optimization with continuous variables similar to (θ-min) (further split into active and inactive set of variables as per $\mathbf{z}^{(t)}$). The main difference from



(a) Objective vs. $\mathtt{t_p}$ (b) Objective vs. $\mathtt{d_I}$ (c) CST-SAT vs. $\mathtt{t_p}$ (d) CST-SAT vs. $\mathtt{d_I}$

Figure 2: Performance of unconstrained and constrained ADMM (aggregated over 10 trials) executed for 1 hour with varying thresholds for the 2 constraints. Note the log-scale on the vertical axis.

($\boldsymbol{\theta}$-min) are the M new optimization variables $u_m, m \in [M]$ active in every ADMM iteration. This active set optimization can be solved in the same way as in ($\boldsymbol{\theta}$-min) (namely with BO). Problem (3) remains a black-box integer program in $\mathbf{z}$ with an updated objective incorporating constraint violations and can be solved with methods discussed for ($\mathbf{z}$-min).

4.1 Empirical evaluation

We consider data from the Home Credit Default Risk Kaggle challenge with the objective of (1−AUROC), and 2 black-box constraints: (i) (**deployment**) Prediction latency $\mathtt{t_p}$ enforcing real-time predictions, (ii) (**fairness**) Maximum pairwise disparate impact $\mathtt{d_I}$ (Calders and Verwer, 2010) across all loan applicant age groups enforcing fairness across groups. We perform the following experiments: (i) fixing $\mathtt{d_I} = 0.7$, we vary $\mathtt{t_p} \in [1, 20]$ (in μs), and (ii) fixing $\mathtt{t_p} = 10\mu\mathtt{s}$ and we vary $\mathtt{d_I} \in [0.05, 0.15]$. Note that the constraints get less restrictive as the thresholds increase. We apply ADMM to the unconstrained problem (UCST) and post-hoc filter constraint satisfying pipelines to demonstrate that these constraints are not trivially satisfied. Then we execute ADMM with these constraints (CST). Using BO for ($\boldsymbol{\theta}$-min) (and (2)) & bandits for ($\mathbf{z}$-min) (and (3)), we get UCST(BO,Ba) & CST(BO,Ba).

Figures 2a & 2b present the best objective achieved when limited only to constraint satisfying pipelines as the constraint on $\mathtt{t_p}$ and $\mathtt{d_I}$ are respectively relaxed. As expected, the objective improves as the constraints relax. In both cases, CST outperforms UCST, with UCST approaching CST as the constraints relax. Figures 2c & 2d (for varying $\mathtt{t_p}$ & $\mathtt{d_I}$ respectively) present the constraint satisfying capability of the optimizer by considering the fraction of constraint-satisfying pipelines found. CST again significantly outperforms UCST, indicating that the constraints are non-trivial to satisfy, and that ADMM is able to effectively incorporate the constraints for improved performance.

5. Conclusion

In this paper, we summarized our recent ADMM based CASH solver and demonstrated the utility of ADMM and its multiple enhancements. We also considered CASH with black-box constraints and demonstrated how ADMM seamlessly handles them. While we skip the multi-fidelity aspect of CASH here, ADMM can use multi-fidelity solvers such as Hyperband and BOHB for the ($\boldsymbol{\theta}$-min) and ($\mathbf{z}$-min) sub-problems. Moreover, we can combine ADMM with a context-free grammar for ML pipelines and AI planning to jointly solve CASH and the pipeline shape search problem with black-box constraints (Katz et al., 2020).

References

Setareh Ariafar, Jaume Coll-Font, Dana Brooks, and Jennifer Dy. Admmbo: Bayesian optimization with unknown constraints using admm. *Journal of Machine Learning Research*, 20(123):1–26, 2019.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites and the openml100. *arXiv preprint arXiv:1708.03731*, 2017.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010.

Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.

Iddo Drori, Yamuna Krishnamurthy, Remi Rampin, Raoni Lourenço, J One, Kyunghyun Cho, Claudio Silva, and Juliana Freire. Alphad3m: Machine learning pipeline synthesis. In *AutoML Workshop at ICML*, 2018.

A. Durand and C. Gagné. Thompson sampling for combinatorial bandits and its application to online feature selection. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1437–1446, 2018.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

Sorelle A Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P Hamilton, and Derek Roth. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 329–338. ACM, 2019.

Nicolo Fusi, Rishit Sheth, and Melih Elibol. Probabilistic matrix factorization for automated machine learning. In *Advances in Neural Information Processing Systems*, pages 3348–3357, 2018.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, pages 507–523, Berlin, Heidelberg, 2011. Springer-Verlag.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248, 2016.

Michael Katz, Parikshit Ram, Shirin Sohrabi, and Octavian Udrea. Exploring context-free languages via planning: The case for automating machine learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 403–411, 2020.

Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML*, pages 2825–2830. Citeseer, 2014.

Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *J. Mach. Learn. Res.*, 18(1):826–830, January 2017. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=3122009.3122034`.

Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Sijia Liu, Parikshit Ram, Deepak Vijaykeerthy, Djallel Bouneffouf, Gregory Bramble, Horst Samulowitz, Dakuo Wang, Andrew Conn, and Alexander Gray. An ADMM based framework for automl pipeline configuration. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. URL `https://arxiv.org/abs/1905.00424v5`.

Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8-10):1495–1515, 2018.

Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on Automatic Machine Learning*, pages 66–74, 2016.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1 (3):127–239, 2014.

Herilalaina Rakotoarison, Marc Schoenauer, and Michle Sebag. Automated machine learning with monte-carlo tree search. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3296–3303, 2019.

Ashish Sabharwal, Horst Samulowitz, and Gerald Tesauro. Selecting near-optimal learners via incremental data allocation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2012.

Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *arXiv*, 2012. URL `http://arxiv.org/abs/1208.3719`.

Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

Zheng Xu, Soham De, Mario Figueiredo, Christoph Studer, and Tom Goldstein. An empirical study of admm for nonconvex problems. *arXiv preprint arXiv:1612.03349*, 2016.

Appendix A. Generalization of the CASH Problem (CASH)

Generalization for more flexible pipelines. We can extend the problem formulation (CASH) to enable optimization over the ordering of the functional modules. For example, we can choose between 'preprocessor → transformer → feature selector' OR 'feature selector → preprocessor → transformer'. The ordering of $T \leq N$ modules can be optimized by introducing T^2 Boolean variables $\mathbf{o} = \{o_{ik} : i, k \in [T]\}$, where $o_{ik} = 1$ indicates that module i is placed at position k. The following constraints are then needed:

(i) $\sum_{k\in[T]} o_{ik} = 1, \forall i \in [T]$, indicating that module i is placed at a single position, and

(ii) $\sum_{i\in[T]} o_{ik} = 1 \forall k \in [T]$ enforcing that only one module is placed at position k.

These variables and constraints can be added to $\mathbf{z}$ in problem (CASH) ($\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N, \mathbf{o}\}$). The resulting formulation still obeys the generic form of (CASH), which as we will demonstrate in the following section, can be efficiently solved by an operator splitting framework like ADMM. We can also extend the above formulation to allow the choice of multiple algorithms from the same module. Note that we increase the combinatorial complexity of the problem as we extend the formulation.

Appendix B. Artificial black-box objective

We want to devise an artificial black-box objective to study the behaviour of the proposed schemes and baselines that matches the properties of the AutoML problem (CASH) where

1. The same pipeline (the same algorithm choices $\mathbf{z}$ and the same hyperparameters θ) always gets the same value.
2. The objective is not convex and possibly non-continuous.
3. The objective captures the conditional dependence between $\mathbf{z}_i$ and θ_{ij} – the objective is only dependent on the hyper-parameters θ_{ij} if the corresponding $z_{ij} = 1$.
4. Minor changes in the hyper-parameters θ_{ij} can cause only small changes in the objective.
5. The output of module i is dependent on its input from module $i - 1$.

Novel artificial black-box objective. To this end, we propose the following novel black-box objective that emulates the structure of (CASH):

- For each $(i, j), i \in [N], j \in [K_i]$, we fix a weight vector $\mathbf{w}_{ij}$ (each entry is a sample from $\mathcal{N}(0, 1)$) and a seed s_{ij}.
- We set $f_0 = 0$.
  - For each module i, we generate a value

$$v_i = \sum_j z_{ij} \left| \frac{\mathbf{w}_{ij}^\top \theta_{ij}}{\mathbf{1}^\top \theta_{ij}} \right|$$

which only depends on the θ_{ij} corresponding to the $z_{ij} = 1$, and the denominator ensures that the number (or range) of the hyper-parameters does not bias the objective towards (or away from) any particular algorithm.

- We generate n samples $\{f_{i,1}, \ldots, f_{i,n}\} \sim \mathcal{N}(f_{i-1}, \nu_i)$ with the fixed seed s_{ij}, ensuring that the same value will be produced for the same pipeline.
- $f_i = \max_{m=1,\ldots,n} |f_{i,m}|$.

- Output f_N

The basic idea behind this objective is that, for each operator, we create a random (but fixed) weight vector $\mathbf{w}_{ij}$ and take a weighted normalized sum of the hyper-parameters θ_{ij} and use this sum as the scale to sample from a normal distribution (with a fixed seed s_{ij}) and pick the maximum absolute of n (say 10) samples. For the first module in the pipeline, the mean of the distribution is $f_0 = 0.0$. For the subsequent modules i in the pipeline, the mean f_{i-1} is the output of the previous module $i-1$. This function possesses all the aforementioned properties of the AutoML problem (CASH).

In black-box optimization with this objective, the black-box evaluations are very cheap in contrast to the actual AutoML problem where the black-box evaluation requires a significant computational effort (and hence time). However, we utilize this artificial objective to evaluate the black box optimization schemes when the computational costs are dominated by the actual derivative-free optimization and not by the black-box evaluations.

Appendix C. ADMM Decomposition for (CASH)

Introduction of continuous surrogate loss. We begin by proposing a surrogate loss of problem (CASH), which can be defined over the continuous domain. With $\widetilde{\mathcal{D}}_{ij}$ as the continuous relaxation of the integer space $\mathcal{D}_{ij}$ (if $\mathcal{D}_{ij}$ includes integers ranging from $\{l, \ldots, u\} \subset \mathbb{Z}$, then $\widetilde{\mathcal{D}}_{ij} = [l, u] \subset \mathbb{R}$), and $\widetilde{\theta}^d$ as the continuous surrogates for θ^d with $\widetilde{\theta}_{ij} \in \widetilde{\mathcal{D}}_{ij}$ (corresponding to $\theta_{ij} \in \mathcal{D}_{ij}$), we utilize a surrogate loss function $\widetilde{f}$ for problem (CASH) defined solely over the continuous domain with respect to θ:

$$\widetilde{f}\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) := f\left(\mathbf{z}, \left\{\theta^c, \mathcal{P}_{\mathcal{D}}\left(\widetilde{\theta}^d\right)\right\}\right), \tag{4}$$

where $\mathcal{P}_{\mathcal{D}}(\widetilde{\theta}^d) = \{\mathcal{P}_{\mathcal{D}_{ij}}(\widetilde{\theta}^d_{ij}), \forall i \in [N], j \in [K_i]\}$ is the projection of the continuous surrogates onto the integer set. This projection is **necessary** since the black-box function is defined (hence *can only be evaluated*) on the integer sets $\mathcal{D}_{ij}$s, not the relaxed continuous set $\widetilde{\mathcal{D}}_{ij}$s. Note that this projection and has an efficient closed form. Given the above definitions, we have the following *equivalent* form of the problem (CASH):

$$\min_{\mathbf{z}, \theta^c, \widetilde{\theta}^d, \delta} \widetilde{f}\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) \text{ subject to } \begin{cases} \mathbf{z}_i \in \{0,1\}^{K_i}, \mathbf{1}^\top \mathbf{z}_i = 1, \forall i \in [N] \\ \theta^c_{ij} \in \mathcal{C}_{ij}, \widetilde{\theta}^d_{ij} \in \widetilde{\mathcal{D}}_{ij}, \forall i \in [N], j \in [K_i] \\ \delta_{ij} \in \mathcal{D}_{ij}, \forall i \in [N], j \in [K_i] \\ \widetilde{\theta}^d_{ij} = \delta_{ij}, \forall i \in [N], j \in [K_i], \end{cases} \tag{5}$$

where the equivalence between problems (CASH) & (5) is established by the equality constraint $\widetilde{\theta}^{d}_{ij} = \delta_{ij} \in \mathcal{D}_{ij}$, implying $\mathcal{P}_{\mathcal{D}_{ij}}(\widetilde{\theta}^{d}_{ij}) = \widetilde{\theta}^{d}_{ij} \in \mathcal{D}_{ij}$ and $\widetilde{f}(z, \{\theta^{c}, \widetilde{\theta}^{d}\}) = f(z, \{\theta^{c}, \widetilde{\theta}^{d}\})$, thereby making the objective functions in problems (CASH) and (5) equal. We highlight that the introduction of the continuous surrogate loss (4) is the key to efficiently handling AutoML problems (CASH) by allowing us to perform theoretically grounded operator splitting methods, e.g., ADMM, over mixed continuous/integer hyper-parameters and integer model selection variables.

Operator splitting from ADMM. Using the notation that $I_{\mathcal{X}}(x) = 0$ if $x \in \mathcal{X}$ else $+\infty$, and defining the sets

$$\mathcal{Z} = \left\{ z \colon z = \{ z_i \colon z_i \in \{0,1\}^{K_i}, \mathbf{1}^{\top} z_i = 1, \forall i \in [N] \} \right\}, \tag{6}$$

$$\mathcal{C} = \left\{ \theta^{c} \colon \theta^{c} = \{ \theta^{c}_{ij} \in \mathcal{C}_{ij} \forall i \in [N], j \in [K_i] \} \right\}, \tag{7}$$

$$\mathcal{D} = \{ \delta \colon \delta = \{ \delta \in \mathcal{D}_{ij} \forall i \in [N], j \in [K_i] \} \}, \tag{8}$$

$$\widetilde{\mathcal{D}} = \left\{ \widetilde{\theta}^{d} \colon \widetilde{\theta}^{d} = \{ \widetilde{\theta}^{d}_{ij} \in \widetilde{\mathcal{D}}_{ij} \forall i \in [N], j \in [K_i] \} \right\}, \tag{9}$$

we can re-write problem (5) as

$$\min_{z, \theta^{c}, \widetilde{\theta}^{d}, \delta} \widetilde{f}\left(z, \left\{\theta^{c}, \widetilde{\theta}^{d}\right\}\right) + I_{\mathcal{Z}}(z) + I_{\mathcal{C}}(\theta^{c}) + I_{\widetilde{\mathcal{D}}}(\widetilde{\theta}^{d}) + I_{\mathcal{D}}(\delta) \text{ subject to } \widetilde{\theta}^{d} = \delta, \tag{10}$$

with the corresponding augmented Lagrangian function

$$\begin{aligned} &\mathcal{L}(z, \theta^{c}, \widetilde{\theta}^{d}, \delta, \lambda) \coloneqq \\ &\widetilde{f}\left(z, \left\{\theta^{c}, \widetilde{\theta}^{d}\right\}\right) + I_{\mathcal{Z}}(z) + I_{\mathcal{C}}(\theta^{c}) + I_{\widetilde{\mathcal{D}}}(\widetilde{\theta}^{d}) + I_{\mathcal{D}}(\delta) + \lambda^{\top}\left(\widetilde{\theta}^{d} - \delta\right) + \frac{\rho}{2}\left\|\widetilde{\theta}^{d} - \delta\right\|_2^2, \end{aligned} \tag{11}$$

where λ is the Lagrangian multiplier, and $\rho > 0$ is a penalty parameter for the augmented term.

ADMM (Boyd et al., 2011) alternatively minimizes the augmented Lagrangian function (11) over *two* blocks of variables, leading to an efficient operator splitting framework for nonlinear programs with *nonsmooth* objective function and *equality* constraints. Hence ADMM solves the original problem (CASH) when reformulated as problem (10) with a sequence of easier sub-problems. Specifically, ADMM solves problem (CASH) by alternatively minimizing (11) over variables $\{\theta^{c}, \widetilde{\theta}^{d}\}$, and $\{\delta, z\}$. This can be equivalently converted into 3 sub-problems over variables $\{\theta^{c}, \widetilde{\theta}^{d}\}$, δ and z, respectively. ADMM decomposes the optimization variables into two blocks and alternatively minimizes the augmented Lagrangian function (11) in the following manner at any ADMM iteration t:

$$\left\{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\right\} = \arg\min_{\theta^{c}, \widetilde{\theta}^{d}} \mathcal{L}\left(z^{(t)}, \theta^{c}, \widetilde{\theta}^{d}, \delta^{(t)}, \lambda^{(t)}\right) \tag{12}$$

$$\left\{\delta^{(t+1)}, z^{(t+1)}\right\} = \arg\min_{\delta, z} \mathcal{L}\left(z, \theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}, \delta, \lambda^{(t)}\right) \tag{13}$$

$$\lambda^{(t+1)} = \lambda^{(t)} + \rho\left(\widetilde{\theta}^{d\,(t+1)} - \delta^{(t+1)}\right). \tag{14}$$

Problem (12) can be simplified by removing constant terms to get

$$\begin{aligned}
&\left\{\theta^{c\,(t+1)},\ \widetilde{\theta}^{d\,(t+1)}\right\} \\
&\quad = \underset{\theta^c,\widetilde{\theta}^d}{\arg\min} \quad \widetilde{f}\left(\mathbf{z}^{(t)}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) + I_{\mathcal{C}}(\theta^c) + I_{\widetilde{\mathcal{D}}}(\widetilde{\theta}^d) \qquad (15)\\
&\qquad\qquad + \boldsymbol{\lambda}^{(t)\top}\left(\widetilde{\theta}^d - \boldsymbol{\delta}^{(t)}\right) + \frac{\rho}{2}\left\|\widetilde{\theta}^d - \boldsymbol{\delta}^{(t)}\right\|_2^2, \\
&\quad = \underset{\theta^c,\widetilde{\theta}^d}{\arg\min} \quad \widetilde{f}\left(\mathbf{z}^{(t)}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) + I_{\mathcal{C}}(\theta^c) + I_{\widetilde{\mathcal{D}}}(\widetilde{\theta}^d) + \frac{\rho}{2}\left\|\widetilde{\theta}^d - \mathbf{b}\right\|_2^2 \qquad (16)\\
&\qquad\qquad \text{where } \mathbf{b} = \boldsymbol{\delta}^{(t)} - \frac{1}{\rho}\boldsymbol{\lambda}^{(t)}.
\end{aligned}$$

A similar treatment to problem (13) gives us

$$\begin{aligned}
\left\{\boldsymbol{\delta}^{(t+1)}, \mathbf{z}^{(t+1)}\right\} &= \underset{\boldsymbol{\delta},\mathbf{z}}{\arg\min} \quad \widetilde{f}\left(\mathbf{z}, \left\{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\right\}\right) + I_{\mathcal{Z}}(\mathbf{z}) \qquad (17)\\
&\qquad + I_{\mathcal{D}}(\boldsymbol{\delta}) + \boldsymbol{\lambda}^{(t)\top}\left(\widetilde{\theta}^{d\,(t+1)} - \boldsymbol{\delta}\right) + \frac{\rho}{2}\left\|\widetilde{\theta}^{d\,(t+1)} - \boldsymbol{\delta}\right\|_2^2, \\
&= \underset{\boldsymbol{\delta},\mathbf{z}}{\arg\min} \quad \widetilde{f}\left(\mathbf{z}, \left\{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\right\}\right) + I_{\mathcal{Z}}(\mathbf{z}) \qquad (18)\\
&\qquad + I_{\mathcal{D}}(\boldsymbol{\delta}) + \frac{\rho}{2}\|\mathbf{a} - \boldsymbol{\delta}\|_2^2 \ \text{ where } \mathbf{a} = \widetilde{\theta}^{d\,(t+1)} + \frac{1}{\rho}\boldsymbol{\lambda}^{(t)}.
\end{aligned}$$

This simplification exposes the independence between $\mathbf{z}$ and $\boldsymbol{\delta}$, allowing us to solve problem (13) independently for $\mathbf{z}$ and $\boldsymbol{\delta}$ as:

$$\boldsymbol{\delta}^{(t+1)} = \underset{\boldsymbol{\delta}}{\arg\min} \quad I_{\mathcal{D}}(\boldsymbol{\delta}) + \frac{\rho}{2}\|\mathbf{a} - \boldsymbol{\delta}\|_2^2 \ \text{ where } \mathbf{a} = \widetilde{\theta}^{d\,(t+1)} + \frac{1}{\rho}\boldsymbol{\lambda}^{(t)}, \qquad (19)$$

$$\mathbf{z}^{(t+1)} = \underset{\mathbf{z}}{\arg\min} \quad \widetilde{f}\left(\mathbf{z}, \left\{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\right\}\right) + I_{\mathcal{Z}}(\mathbf{z}). \qquad (20)$$

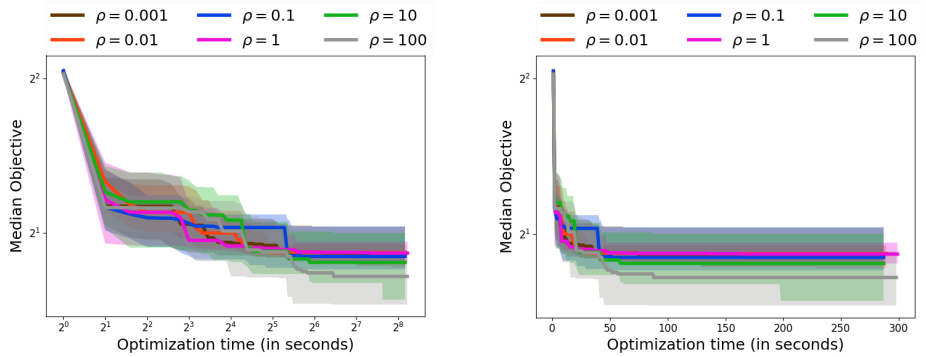
So we are able to decompose problem (3) into problems (16), (19) and (20) which can be solved iteratively along with the $\boldsymbol{\lambda}^{(t)}$ updates (see Table 1).

Appendix D. ADMM parameter sensitivity

The performance of ADMM is dependent on the choice of the penalty parameter $\rho > 0$ for the augmented term in the augmented Lagrangian (11). Figure 3 shows the dependence of adaptive precision ADMM on ρ with the artificial black-box objective (Appendix B). The results indicate that adaptive ADMM for CASH is fairly robust to the choice of ρ.

Appendix E. Empirical Evaluation of ADMM vs. JOPT on OpenML data

We compare the performance of the adaptive precision ADMM with different sub-problem solvers to JOPT(BO) (BO on the joint problem with all the variables) on 8 OpenML (Bischl et al., 2017) binary classification data sets where we optimize for the area under the ROC curve on 10% of the data as the validation set. Both adaptive precision ADMM and



(a) Run time in log scale (b) Run time in linear scale

Figure 3: ADMM **sensitivity to the penalty parameter** ρ with adaptive ADMM. The performance is aggregated over 20 trials. The run time is dominated by the black-box optimization scheme, which, in this case, is a Bayesian optimization via Gaussian Process Regression.

Dataset	$\mathbf{S}_{\text{Ba}}$	$\mathbf{S}_{\text{BO}}$	$\mathbf{I}_{\text{Ba}}$	$\mathbf{I}_{\text{BO}}$
Bank8FM	10×	2×	0%	5%
CPU small	4×	5×	0%	5%
fri-c2	153×	25×	56%	64%
PC4	42×	5×	8%	13%
Pollen	25×	7×	4%	3%
Puma8NH	11×	4×	1%	1%
Sylvine	9×	2×	9%	26%
Wind	40×	5×	0%	5%

Table 1: Comparing ADMM schemes to JOPT(BO), we list the speedup $\mathbf{S}_{\text{Ba}}$ & $\mathbf{S}_{\text{BO}}$ achieved by AdADMM(BO,Ba) & AdADMM(BO,BO) respectively to reach the best objective of JOPT, and the final objective improvement $\mathbf{I}_{\text{Ba}}$ & $\mathbf{I}_{\text{BO}}$ (respectively) over the JOPT objective. These numbers are generated using the aggregate performance of JOPT and AdADMM over 10 trials.

JOPT(BO) are executed for 1 hour with 10 repetitions, and the incumbent objective is averaged over the 10 repetitions. All evaluations were run single-threaded on a 8 core 8GB CentOS virtual machines. For each data set, we note the objective f_J achieved by JOPT(BO) at the end of 1 hour and note the time T_A (in seconds) required by adaptive precision ADMM to reach that objective. Then we also note the best objective achieved by ADMM at the end of 1 hour f_A. The speedup S for ADMM is computed as $S = \frac{3600}{T_A}$ and the improvement (in %) $I = 100 * \frac{f_J - f_A}{f_J}$.

We consider two versions of adaptive precision ADMM – (i) using BO for both the ($\boldsymbol{\theta}$-min) and ($\mathbf{z}$-min) to get AdADMM(BO,BO), and (ii) using BO for the ($\boldsymbol{\theta}$-min) and Combinatorial multi-armed bandits (Ba) for the ($\mathbf{z}$-min) to get AdADMM(BO,Ba). The improvements of ADMM over JOPT(BO) are summarized in Table 1, indicating significant speedup (over 10× in most cases) and further improvement (over 10% in many cases). Table 1 shows that between AdADMM(BO,BO) and AdADMM(BO,Ba), the latter provides *significantly higher speedups*, but the former provides higher additional improvement in the final objective. This demonstrates ADMM's flexibility, for example, allowing choice between faster or more improved solution.

Appendix F. ADMM Decomposition for (CASH) with black-box constraints

Without loss of generality, we assume that $\epsilon_m \geq 0$ for $m \in [M]$. By introducing scalars $u_m \in [0, \epsilon_m]$, we can reformulate the inequality constraint (1) as the equality constraint together with a box constraint

$$g_m\left(\mathbf{z}, \left\{\theta^c, \theta^d\right\}\right) - \epsilon_m + u_m = 0, u_m \in [0, \epsilon_m], \ m \in [M]. \tag{21}$$

We then introduce a continuous surrogate black-box functions $\widetilde{g}_m$ for $g_m, \forall m \in [M]$ in a similar manner to $\widetilde{f}$ given by (4). Following the reformulation of (CASH) that lends itself to the application of ADMM, the version with black-box constraints (21) can be equivalently transformed into

$$\min_{\mathbf{z}, \theta^c, \widetilde{\theta}^d, \delta} \widetilde{f}\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) \text{ subject to } \begin{cases} \mathbf{z}_i \in \{0,1\}^{K_i}, \mathbf{1}^\top \mathbf{z}_i = 1, \forall i \in [N] \\ \theta^c_{ij} \in \mathcal{C}_{ij}, \widetilde{\theta}^d_{ij} \in \widetilde{\mathcal{D}}_{ij}, \forall i \in [N], j \in [K_i] \\ \delta_{ij} \in \mathcal{D}_{ij}, \forall i \in [N], j \in [K_i] \\ \widetilde{\theta}^d_{ij} = \delta_{ij}, \forall i \in [N], j \in [K_i] \\ u_m \in [0, \epsilon_m], \forall m \in [M] \\ \widetilde{g}_m\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) - \epsilon_m + u_m = 0, \forall m \in [M]. \end{cases} \tag{22}$$

Compared to problem (5), the introduction of auxiliary variables $\{u_m\}$ enables ADMM to incorporate *black-box equality* constraints as well as elementary *white-box* constraints.

Defining $\mathcal{U} = \{\mathbf{u} \colon \mathbf{u} = \{u_m \in [0, \epsilon_m] \forall m \in [M]\}\}$, we can go through the mechanics of ADMM to get the augmented Lagrangian with $\boldsymbol{\lambda}$ and $\mu_m \forall m \in [M]$ as the Lagrangian multipliers and $\rho > 0$ as the penalty parameter as follows:

$$\begin{aligned} &\mathcal{L}\left(\mathbf{z}, \theta^c, \widetilde{\theta}^d, \delta, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}\right) = \\ &\widetilde{f}\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) + I_{\mathcal{Z}}(\mathbf{z}) + I_{\mathcal{C}}(\theta^c) + I_{\widetilde{\mathcal{D}}}(\widetilde{\theta}^d) + I_{\mathcal{D}}(\delta) + \boldsymbol{\lambda}^\top\left(\widetilde{\theta}^d - \delta\right) + \frac{\rho}{2}\left\|\widetilde{\theta}^d - \delta\right\|_2^2 \\ &+ I_{\mathcal{U}}(\mathbf{u}) + \sum_{i=1}^{M} \mu_m\left(\widetilde{g}_m\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) - \epsilon_m + u_m\right) + \frac{\rho}{2}\sum_{i=1}^{M}\left(\widetilde{g}_m\left(\mathbf{z}, \left\{\theta^c, \widetilde{\theta}^d\right\}\right) - \epsilon_m + u_m\right)^2. \end{aligned}$$

ADMM decomposes the optimization variables into two blocks for alternate minimization of the augmented Lagrangian in the following manner at any ADMM iteration t

$$\left\{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}, \mathbf{u}^{(t+1)}\right\} = \underset{\theta^c, \widetilde{\theta}^d, \mathbf{u}}{\arg\min}\, \mathcal{L}\left(\mathbf{z}^{(t)}, \theta^c, \widetilde{\theta}^d, \delta^{(t)}, \mathbf{u}, \boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right) \tag{23}$$

$$\left\{\delta^{(t+1)}, \mathbf{z}^{(t+1)}\right\} = \underset{\delta, \mathbf{z}}{\arg\min}\, \mathcal{L}\left(\mathbf{z}, \theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}, \delta, \mathbf{u}^{(t+1)}, \boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right) \tag{24}$$

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \rho\left(\widetilde{\theta}^{d\,(t+1)} - \delta^{(t+1)}\right) \tag{25}$$

$$\forall m \in [M], \quad \mu_m{}^{(t+1)} = \mu_m{}^{(t)} + \rho\left(\widetilde{g}_m(\mathbf{z}^{(t+1)}, \{\theta^{c\,(t+1)}, \widetilde{\theta}^{d\,(t+1)}\}) - \epsilon_m + u_m{}^{(t+1)}\right). \tag{26}$$

Note that, unlike the unconstrained case, the update of the augmented Lagrangian multiplier μ_m requires the evaluation of the black-box function for the constraint g_m. Simplifying

problem (23) gives us

$$\begin{aligned}\min_{\theta^c,\widetilde{\theta}^d,\mathbf{u}} \quad & \widetilde{f}\left(\mathbf{z}^{(t)},\left\{\theta^c,\widetilde{\theta}^d\right\}\right) \\ & + \frac{\rho}{2}\left[\left\|\widetilde{\theta}^d-\mathbf{b}\right\|_2^2+\sum_{i=1}^{M}\left[\widetilde{g}_m\left(\mathbf{z}^{(t)},\left\{\theta^c,\widetilde{\theta}^d\right\}\right)-\epsilon_m+u_m+\frac{1}{\rho}{\mu_m}^{(t)}\right]^2\right] \\ \text{subject to} & \begin{cases}\theta^c_{ij}\in\mathcal{C}_{ij}\forall i\in[N],j\in[K_i],\\ \widetilde{\theta}^d_{ij}\in\widetilde{\mathcal{D}}_{ij}\forall i\in[N],j\in[K_i], \quad \text{where } \mathbf{b}=\boldsymbol{\delta}^{(t)}-\frac{1}{\rho}\boldsymbol{\lambda}^{(t)},\\ u_m\in[0,\epsilon_m],\end{cases}\end{aligned} \tag{27}$$

which can be further split into active and inactive set of continuous variables based on the $\mathbf{z}^{(t)}$ as in the solution of problem (16) (the θ-min problem). The main difference from the unconstrained case in problem (16) (the θ-min problem) to note here is that the black-box optimization with continuous variables now has M new variables u_m (M is the total number of black-box constraints) which are active in every ADMM iteration. This problem (27) can be solved in the same manner as problem (16) (θ-min) using SMBO or TR-DFO techniques.

Simplifying and utilizing the independence of $\mathbf{z}$ and $\boldsymbol{\delta}$, we can split problem (24) into the following problem for $\boldsymbol{\delta}$

$$\min_{\delta} \quad \frac{\rho}{2}\|\boldsymbol{\delta}-\mathbf{a}\|_2^2 \quad \text{subject to } \delta_{ij}\in\mathcal{D}_{ij}\forall i\in[N],j\in[K_i] \text{ where } \mathbf{a}=\widetilde{\theta}^{d\,(t+1)}+\frac{1}{\rho}\boldsymbol{\lambda}^{(t)}, \tag{28}$$

which remains the same as problem (19) (the $\boldsymbol{\delta}$-min problem) in the unconstrained case, while the problem for $\mathbf{z}$ becomes

$$\begin{aligned}\min_{\mathbf{z}} \quad & \widetilde{f}(\mathbf{z},\{\theta^{c\,(t+1)},\widetilde{\theta}^{d\,(t+1)}\}) \\ & + \frac{\rho}{2}\sum_{i=1}^{M}\left[\widetilde{g}_m(\mathbf{z},\{\theta^{c\,(t+1)},\widetilde{\theta}^{d\,(t+1)}\})-\epsilon_m+{u_m}^{(t+1)}+\frac{1}{\rho}{\mu_m}^{(t)}\right]^2 \\ & \text{subject to } \mathbf{z}_i\in\{0,1\}^{K_i},\mathbf{1}^\top\mathbf{z}_i=1,\forall i\in[N].\end{aligned} \tag{29}$$

The problem for $\mathbf{z}$ is still a black-box integer programming problem, but now with an updated black-box function and can be handled with techniques proposed for the combinatorial problem (20) in the absence of black-box constraints (the $\mathbf{z}$-min problem).