Accelerating Cutting-Plane Algorithms via Reinforcement Learning Surrogates

Kyle Mana[1], Fernando Acero[1,2], Stephen Mak[3], Parisa Zehtabi[1], Michael Cashmore[1], Daniele Magazzeni[1], Manuela Veloso[1]

[1]J.P. Morgan AI Research
[2]University College London
[3]University of Cambridge
{kyle.mana, parisa.zehtabi, michael.cashmore, daniele.magazzeni, manuela.veloso}@jpmorgan.com,
fernando.acero@ucl.ac.uk, sm2410@cam.ac.uk

Abstract

Discrete optimization belongs to the set of $\mathcal{NP}$-hard problems, spanning fields such as mixed-integer programming and combinatorial optimization. A current standard approach to solving convex discrete optimization problems is the use of cutting-plane algorithms, which reach optimal solutions by iteratively adding inequalities known as *cuts* to refine a feasible set. Despite the existence of a number of general-purpose cut-generating algorithms, large-scale discrete optimization problems continue to suffer from intractability. In this work, we propose a method for accelerating cutting-plane algorithms via reinforcement learning. Our approach uses learned policies as surrogates for $\mathcal{NP}$-hard elements of the cut generating procedure in a way that (i) accelerates convergence, and (ii) retains guarantees of optimality. We apply our method on two types of problems where cutting-plane algorithms are commonly used: stochastic optimization, and mixed-integer quadratic programming. We observe the benefits of our method when applied to Benders decomposition (stochastic optimization) and iterative loss approximation (quadratic programming), achieving up to 45% faster average convergence when compared to modern alternative algorithms.

Introduction

A large number of problems require discrete decisions. Examples include the decision to purchase an item in whole units, schedule tasks with finite resources, or plan the shortest route through given locations. Even seemingly simple problems can become incredibly challenging to solve when they necessitate discrete decisions (Parker and Rardin 2014). In some cases, discrete optimization problems are provably unsolvable in polynomial time, e.g., it is known that integer programs with quadratic constraints are not solvable at all by Turing machines (Jeroslow 1973). Indeed, problems that would otherwise take fractions of a second to solve can take hours, if not days in discrete space.

When faced with discrete decisions, heuristic methods such as rounding can offer fast solutions, but at an unknown cost of sub-optimality and if not careful, infeasibility. Due to these issues, there is a strong desire to generate naturally integer, and provably optimal solutions. General-purpose solvers typically rely on a mixture of branch-and-bound and cutting-plane methods to achieve this (Bonami et al. 2008). Despite advancements in cutting and branching techniques, large-scale problems remain heavily dependent on domain-specific algorithms that exploit problem structure for more efficient convergence. In this paper, we propose a procedure that leverages reinforcement learning for accelerating cutting-plane algorithms. We focus on the domain-specific cutting-plane algorithms of Benders decomposition (applied to stochastic optimization) and a cutting-plane method for solving regression problems with L_0 regularization. Our proposed method is generalizable to any cutting-plane procedure and not isolated to the two examples given.

The first of the two cutting-plane procedures we consider, Benders decomposition (BD), is a method that aims to exploit a unique *block* structure commonly found in stochastic optimization (SO) problems. Considering each scenario as a unique block, the global problem is decomposed into a master problem (MP) and a collection of scenario-specific sub-problems (SP). Each SP ingests decisions from the MP, and shares loss information back to the MP in the form of constraints. Iteratively, the MP gains a better understanding of global loss exhibited by each SP, and eventually converges (Benders 1962).

The second procedure we enhance is aimed at solving machine learning problems with sparsity enforced via an L_0 regularization term. In settings such as medical imaging (Daducci et al. 2014), economics (Fan, Lv, and Qi 2011), or causal learning (Idé et al. 2021), sparsity via L_0 regularization plays a critical role. Furthermore, it has been shown that regularization techniques such as lasso (L_1), ridge (L_2), and elastic-net (L_1, L_2) can improve out-of-sample performance (Zou and Hastie 2005). Typically, sparse regression is implemented via an L_1 regularization term (i.e. lasso), eliminating the necessity to optimize over support of the coefficient set. Louizos, Welling, and Kingma (2018) note that using L_0 penalties in parametric models is generally intractable due to non-differentiability and the combinatorial nature of cardinality regularization. Despite its complexity, the importance of L_0 regularization is advocated for by Bertsimas, King, and Mazumder (2016), who argue that using L_1 regularization to achieve sparsity

results in undesirably biased coefficient estimates by disproportionately affecting larger coefficients in the set. To solve least-squares regression problems with L_0 regularization, we introduce a cutting-plane algorithm that iteratively optimizes the sum of penalized support, and a proxy variable bound via sub-gradient approximations of the least-squares loss.

Both of the cutting-plane procedures we consider suffer from three easily observed limitations. First, in discrete space they rely on an $\mathcal{NP}$-hard mixed-integer master problem (MIMP). Second, they operate by abstracting the loss function to a proxy variable, bound from below (assuming minimization without loss of generality) by SP constraints. This can lead to solutions being highly sub-optimal until loss is well represented via constraints. Third, with each iteration an SP generates sub-gradients that are passed to the MIMP as constraints (i.e. cuts), a process which linearly scales the complexity of the MIMP.

Related Work

Acknowledging the overall benefit and potential of cutting-plane algorithms, accelerating these methods has become a compelling research problem. Magnanti and Wong (1981) proposed pareto-optimal cut selection for BD. In production routing applications, Adulyasak et al. (2015) implement lower-bound lifting inequalities to tighten initial lower bounds, and exploit scenario grouping to reduce complexity at each iteration. Crainic et al. (2016) aid initial iterations by including an informative subset of scenarios within the MIMP. Lee et al. (2021) offer a machine learning approach to predict constraint importance; retaining only important cuts and limiting MIMP complexity. Each of these proposals has shown computational benefits, but remain solely dependent on the expensive MIMP to generate successive solutions. In contrast, Poojari and Beasley (2009) replace the MIMP of BD with a genetic algorithm to produce faster feasible solutions. Although the heuristic produces fast MP solutions, it is still reliant on SP approximations to obtain scenario loss, and offers feasible as opposed to certifiably optimal solutions. We refer to Rahmaniani et al. (2017) for a review of BD methods.

Machine learning methods have been explored for mixed-integer programming and combinatorial optimization (CO). We refer to Mazyavkina et al. (2021) for a review of RL for CO. Nair et al. (2020) use neural networks trained via imitation learning to improve branch-and-bound methods for solving MIPs. RL has been used to solve large combinatorial problems, achieving performance close to expert implementations, as shown by Delarue et al. (2020) for notoriously challenging capacitated vehicle routing problems. Recent work has explored the use of RL to improve the performance of modern CO solvers, which typically rely on human-designed heuristics tuned with experience or data. In this sense, RL for cut selection in Integer Programs (IPs) was proposed by Tang et al. (2020), and hierarchical RL for cut selection in MILPs was proposed by Wang et al. (2023).

In our work, a surrogate to the MIMP generates fast solutions after learning pseudo-optimal decisions via RL in similar environments. At varying rates, the MIMP is still run to retrieve the certificate of optimality offered. Our contributions are as follows:

- A generalized method of accelerating cutting-plane algorithms that retrieves optimal solutions while drastically reducing run times.
- Three surrogate solution selection methods, including one that uses cuts to inform selection of surrogate MP solutions, offering a further unification of the surrogate MP within the cutting-plane algorithmic framework.
- Empirical evaluation of our approach on two different cutting-plane algorithms. We offer explicit formulations, leverage the learned policy of an RL agent as our surrogate MP in both cases, and provide results showing up to a 45% reduction in run-time against modern alternative methods.

Background

We now discuss relevant background on BD, cutting-planes for L_0 regression, and RL.

Benders Decomposition

A widely used form of stochastic optimization is Sample Average Approximation (SAA). In essence, SAA aims to approximate loss over the distribution of possible scenarios using Monte-Carlo simulation. In SAA, R scenarios are simulated, with each simulation yielding its own deterministic SP with a loss function $f(x, w, D_r)$, where x is a set of global decisions (universal across all scenarios), w is a cost vector, and D_r is a set of scenario-specific parameters. The total loss is computed as the average across scenarios,

$$\ell(x) = \frac{1}{R} \sum_{\forall r \in R} f(x, w, D_r) \tag{1}$$

To combat scalability issues as the number of simulations grow, decomposition methods are commonly employed to solve SAA. Here we introduce the principles of Benders decomposition. Consider an SAA problem of the form:

$$\min_{x,y} c^T x + \frac{1}{R} \sum_{\forall r \in R} w^T y_r \tag{2}$$

s.t.

$$Ax = b \tag{3}$$

$$Bx + D_r y_r = g, \qquad \forall r \in R \tag{4}$$

$$x \in \mathbb{Z}, y_r \in \mathbb{Z}^+, \qquad \forall r \in R \tag{5}$$

where x is our set of global decisions, A, b, and B are parameters that define constraints on x, c is the cost of global decisions, D_r are scenario-specific parameters, y_r is a set of decisions made independently within each scenario, g constrains a combination of global and scenario-specific decisions, and w is the cost of each scenario-specific decision. In this formulation, $w^T y_r$ is equivalent to (1). The first step of BD is to separate global decision variables x and scenario specific decision variables y_r, yielding a MP:

$$\{\min_{x,\theta} c^T x + \frac{1}{R} \sum_{\forall r \in R} \theta_r : Ax = b, x \in \mathbb{Z}^+\} \tag{6}$$

and a collection of R SPs, where $\forall r \in R$:

$$\{\min_{y_r} w^T y_r : D_r y_r = g - Bx^*, y_r \in \mathbb{R}^+\} \quad (7)$$

The SPs ingest a fixed x^* based on the solution to (6), and are solved to obtain optimal SP decisions y_r. Note that BD introduces a set of auxiliary variables $\theta_r, \forall r \in R$ to the MP (6). This auxiliary variable, frequently called the recourse variable, is responsible for tracking an approximation of the SP loss that has been moved to (7). Let us assume the SP is always feasible. This is not a necessary assumption, but simplifies the following description of BD.

Integrality on y_r has been relaxed in the SP. This relaxation is necessary for BD, and is only possible when (i) the SP variables were not discrete to begin with, or (ii) the decomposition results in a totally-unimodular SP structure. Taking the dual of the SP yields:

$$\{\max_{q_r} q_r^T (g - Bx^*) : q_r^T D_r \leq w\} \quad (8)$$

The dual SP has three essential properties. First, through strong duality the optimal value of (8) is equivalent to the optimal value of (7) at x^*. Second, the objective function (8) is linear with respect to the MP decisions x. And lastly, with the optimal dual values of q_r^* we can establish

$$\{\min_{y_r} w^T y_r : D_r y_r = g - Bx\} \geq q_r^{*T}(g - Bx), \forall x \in \mathbb{R}, \forall w \in \mathbb{R} \quad (9)$$

via weak duality. With these traits established, we see that the optimal dual SP objective $q_r^{*T}(g - Bx)$ can be included as a valid constraint on θ_r in the MIMP. These constraints serve as sub-gradient approximations of the SP loss. For each SP solution, we can update the MIMP with the valid constraint of $\theta_r \geq q_r^{*T}(g - Bx)$ and re-solve for a new x. This process is repeated until the SP's do not offer any strengthening constraints on θ_r, indicating convergence and full approximation of SP loss. Figure 1 offers a visual representation of this process, which translates to the L_0 regularization application that we introduce next.

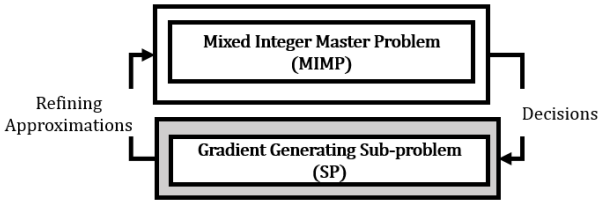


Figure 1: Iterative procedure of Benders decomposition, alternating between a MIMP (6) and SP (8).

Cutting-Planes for L_0 Regularized Regression

In statistical analysis and machine learning, high-dimension datasets may necessitate the use of methods that distinguish a meaningful feature subset. Sparse regression aims to minimize loss while limiting support over the coefficient set, represented by β. Using L_0 regularization encourages this sparsity by penalizing support over β. Consider data given by the general form $y = f(X, \beta) + \epsilon$, where f is a possibly nonlinear function. The problem of L_0 regularized regression

$$\min_{\beta} ||f(X, \beta) - y||_2^2 + \lambda \sum_{\forall i \in p} ||\beta_i||_0 \quad (10)$$

can be cast as a mixed-integer quadratic program MIQP, where P represents the number of features and M is a sufficiently large constant:

$$\{\min_{\beta, z} ||f(X, \beta) - y||_2^2 + \lambda \sum_{\forall i \in p} z_i : |\beta_i| \leq z_i M \ \forall i \in P, \beta \in \mathbb{R}^P, z \in \{0,1\}^P\} \quad (11)$$

MIQPs of the form (11) struggle in high-dimensional settings, inspiring our use of cutting-plane procedures. To do this, we can reframe (11) as a linear program:

$$\{\min_{\beta, \theta, z} \theta + \lambda \sum_{\forall i \in p} z_i : |\beta_i| \leq z_i M \ \forall i \in P, \theta \geq 0, \beta \in \mathbb{R}^P, z \in \{0,1\}^P, \theta \in \mathbb{R}\} \quad (12)$$

where θ serves as a proxy variable for the convex and differentiable loss $||f(X, \beta) - y||_2^2$. For each iteration n of (12) we compute the loss l_n and sub-gradient $\nabla g(\beta^{(n)})$ to constrain θ with a lower bound in the form of a linear constraint. If we denote the polytope defined by (12) as $\mathcal{P}_0$, after n iterations the polytope $\mathcal{P}_n$ is restricted to $\mathcal{P}_{n-1} \cup \{\beta, \theta : \theta \geq l_n + \nabla g(\beta^{(n)})(\beta - \beta^{(n)})\}$. This process terminates when the gap between l_n and the evaluation of the objective from (12) over $\mathcal{P}_n$ is within a tolerance e.

Reinforcement Learning

RL is a framework for solving sequential decision-making problems, formulated as a Markov Decision Process (MDP) (Sutton and Barto 2018). Our proposed framework requires casting the optimization problem at hand as an MDP, i.e., decision variables form the action space and the negative of the cost function is the reward function. Formally, MDPs are defined as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the set of transition probabilities from states s_t to s_{t+1} upon taking action a_t, and $\mathcal{R}$ is the reward function. We note how $\mathcal{T}$ and $\mathcal{R}$ may be non-deterministic, and therefore MDPs may be used to model problems pertaining SO. A discount factor γ is typically introduced to discount rewards. We denote a policy parametrised by ϕ as $\pi_\phi : \mathcal{S} \to \mathcal{A}$.

RL algorithms may be categorized as value-based or policy gradient methods. Value-based methods learn a value function or action-value function, from which optimal actions can be implicitly obtained, whereas policy gradient methods directly optimize an explicit representation of the optimal policy. The value function and action-value functions for episodic MDPs with horizon T are given by:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^k r_{t+k+1} | s = s_t \right] \quad (13)$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=t}^{T} \gamma^{k} r_{t+k+1} | s = s_t, a = a_t\right] \quad (14)$$

where it can be seen that the action-value function Q^{π} is only practically applicable for discrete action spaces (or discretizations of continuous action spaces), and is directly susceptible to the curse of dimensionality. Alternatively, policy gradient methods update policy parameters ϕ via an estimate of the policy gradient, as first introduced by Sutton et al. (1999). Policy gradient methods can be used for discrete or continuous action spaces, and are based on some expression of the policy gradient:

$$\nabla_{\phi}\mathbb{E}\left[\sum_{t=0}^{T} r_t\right] \approx \mathbb{E}\left[\sum_{t=0}^{T} \Psi_t \nabla_{\phi} \log \pi_{\phi}(a_t|s_t)\right] \quad (15)$$

where Ψ_t may be the (discounted) returns of the trajectory, the action-value function, the advantage function, temporal-difference residual, or else, yielding different policy gradient algorithms (Schulman et al. 2016). Proximal Policy Optimization (PPO) is a powerful policy gradient algorithm that avoids detrimentally large policy updates proposed by Schulman et al. (2017), where a surrogate objective to (15) is used based on a probability ratio which is clipped whenever $|\frac{\pi_{\phi}^{\text{new}}(a_t|s_t)}{\pi_{\phi}^{\text{old}}(a_t|s_t)}| > \epsilon$ for some small ϵ, providing a lower bound on the unclipped objective. Consequently, PPO provides stable updates, whilst being on-policy and thus susceptible to low sample efficiency compared to off-policy algorithms such as Q-learning. Compared to value-based methods, using an explicit parametric policy is a more natural choice for solving MDPs for which the optimal policy may be stochastic, as the policy can be a parametric stochastic function, whereas value-based methods require crafting a sampling strategy (e.g. ϵ-greedy) to generate a stochastic policy from a value function. Sutton et al. (1999) discussed advantages of stochastic policies in contrast to policies induced by value functions.

Actor-critic methods combine the benefits of value-based methods and policy gradients. Value estimates can be used as a baseline for advantage estimates (Sutton et al. 1999). Modern actor-critic algorithms frequently use Generalized Advantage Estimation (GAE), an exponentially-weighted advantage estimator that addresses the bias-variance tradeoff (Schulman et al. 2016). We use actor-critic PPO with GAE.

Accelerating Cutting-Plane Algorithms

We now introduce our proposed acceleration method. First, we offer specifics on how a surrogate is used in place of the MIMP. Then, we introduce three possible mechanisms for leveraging the surrogate. Lastly, we offer a more thorough coverage of the theoretical benefits that may be provided by a surrogate, and known deficiencies of cutting-plane methods that it addresses. We focus on surrogates learned via RL, but we note that our proposed method is agnostic to the nature of the method used to learn the surrogate policy.

Surrogate-MP

Recall the iterative procedure outlined in Figure 1. As is the case in our two examples, we assume the sub-gradient can be computed efficiently (as a linear problem in the case of BD, and in closed form in the case of L_0 regularization). However, each case calls back to an $\mathcal{NP}$-hard MIMP, with complexity that scales linearly with the number of iterations. Given these dynamics, there is a strong desire to (i) increase the speed of each MP iteration and (ii) decrease the total number of calls to the MIMP required. We achieve both results by periodically introducing a faster surrogate in place of the MIMP (Figure 2). This surrogate can be any policy that has learned to map the input space to the discrete decision space with the objective of minimizing the problem loss or cost.

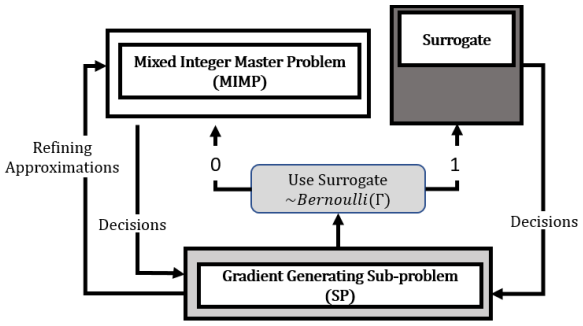


Figure 2: Iterative procedure of Surrogate-MP.

Note in this modified schema that with each iteration, the decision to use the surrogate in place of the MIMP is drawn from a Bernoulli distribution with a control parameter Γ. If a value of 1 is returned from the Bernoulli distribution, the surrogate is used to generate global decisions. Otherwise, the standard MIMP is run and the optimality gap can be confirmed. Regardless of whether the MIMP or surrogate are used, global decisions are passed to the SP and loss approximating cuts are added to the iterative process.

Leveraging Surrogate Solutions

The solutions produced by a surrogate can be used in a variety of ways, and we propose three selection mechanisms. These variants are aimed at answering: (i) How can we use the surrogate to improve convergence? (ii) If surrogate actions are non-deterministic, how can we decide which actions are best to use? The three methods we propose are greedy selection, weighted selection, and informed selection. Each of these methods assume the surrogate has generated a batch of stochastic trajectories of actions for B episodes, i.e. a batch of B distinct solutions, each with loss ℓ_b equal to the negative returns of the trajectory in the MDP.

Greedy Selection This method selects the best performing solution within a batch, i.e. $\arg\min_b(\ell_b)$. In the case of BD we evaluate the solution against an expected outcome, as performance cannot be deterministically evaluated.

Weighted Selection Rather than selecting actions based on expected performance, we can perform weighted random sampling. We use the loss of each solution ℓ_b to define a probability mass $p(b) = \frac{\frac{1}{\ell_b}}{\sum_{\forall b \in B} \frac{1}{\ell_b}}$ for random sampling.

Informed Selection The final proposal incorporates feedback from the constraint matrix on θ at a given iteration. The benefit of utilizing the constraint matrix to select surrogate solutions is that these constraints inherently motivate exploration to either (i) minimal or (ii) poorly approximated regions of the convex loss. Given final convergence is defined by a binding subset of these constraints, it is desirable to explore these regions.

We introduce the constraint matrix $A_r \in \mathbb{R}^{I \times N}$ which contains the sub-gradient approximations imposed on θ_r, and a row vector of constant values $c_r \in \mathbb{R}^B$ that is added to each sub-gradient approximation. Recall $r \in R$, and R is the number of scenarios for BD, while $R = \{1\}$ for L_0 regularization. I refers to the iteration number of the cut-generating algorithm, and N refers to the number of MP decision variables. Each iteration generates a new set of sub-gradient approximations which are added to A_r. These are the same sub-gradients that are applied to θ_r in the MP, and are generated using the SP. On a given iteration, we have a batch of B solutions that have been generated by the surrogate. Decisions for this batch are represented by matrix $D \in \mathbb{Z}^{N \times B}$. We begin by computing the loss approximations of each gradient, for each of the B solutions. This is given by $\mathcal{L}_r \in \mathbb{R}^{I \times B}$, which we define as:

$$\mathcal{L}_r = A_r \cdot D + (\mathbf{c_r} \cdot 1^{1 \times I})^T \tag{16}$$

The $\mathcal{L}_r$ matrix contains approximations of the SP loss for each of the B solutions, generated by each of the I constraints currently placed on θ_r. We can now take the maximum value for each column B as the approximated cost of solution b. In LP terms, this maximum value relates to the binding constraint on θ_r in the MIMP, and is thus our best approximation of SP cost at that point. We represent this approximation ($\hat{\ell}_{b,r}$) as:

$$\hat{\ell}_{b,r} = \max_{\forall i \in I} (\mathcal{L}_r)_{i,b} \tag{17}$$

Now we fully approximate the expected loss for each of the M solutions by taking an average across all $r \in R$, and adding the fixed loss of that decision (denoted f_b):

$$\hat{\ell}_b = \frac{1}{R} \sum_{\forall r \in R} \hat{\ell}_{b,r} + f_b \tag{18}$$

The informed selection then solves the problem $\arg\min_b \hat{\ell}_b$, which is taken as our MP solution, and passed to the SP for constraint generation.

Benefits of Surrogate-MP

The benefits of using a learned surrogate in place of the MIMP are based on two key observations:

1. The time required to generate solutions from a learned surrogate (e.g. inference on a neural net) is negligible compared to the time required to solve a large-scale MIP.
2. The surrogate has learned to produce actions from past experience. As a result, SP loss is expressed in surrogate solutions regardless of how well θ_r approximates SP loss. This means that even at early iterations, the surrogate solutions will be highly reflective of SP loss.

The first benefit is fairly self-explanatory; we desire faster MP solutions, and the surrogate provides them. The second benefit is more nuanced and worth expanding. We recall the general form MIMP (6), where θ_r offers an approximation of SP loss that is refined through linear constraints generated by (8). It is well observed that this approximation can converge quickly if global decisions are localized to the optimal region, but it can also be very slow if global decisions are far from the optimal region or if cuts poorly approximate the loss (Crainic et al. 2016; Baena, Castro, and Frangioni 2020). At initialization, θ_r has not received any feedback from the SP, and is instead bound by some heuristic or known lower bound (commonly $\theta_r \geq 0$ for non-negative loss). Given the lack of information initially imparted on θ_r, the MP generates global solutions that lack consideration of SP loss and can be very distant from the optimal region. Similar to a gradient based algorithm with a miss-specified learning rate, this can lead cutting-plane methods such as BD to oscillate around the minimal region or converge slowly, wasting compute and adding complexity with minimal benefit to the final solution (Baena, Castro, and Frangioni 2020).

The surrogate mitigates this major issue by generating global decisions that reflect an understanding of their associated SP loss without requiring the MP to have strong loss approximations on θ_r. As a result, initial global decisions generated by the surrogate are localized to the minimal region and cuts can quickly approximate the minimum of the convex loss. These two fundamental benefits are the basis for a 30%-45% reduction in run-times, observed in experiments within the two domains that follow.

Experiments

We evaluate our proposed acceleration method on two distinct cutting-plane algorithms in separate domains. The first application displays an acceleration of Benders decomposition, using a stochastic inventory management problem consisting of a basic two-stage decision process, reflecting the MP/SP structure displayed in Figure 1. The second algorithm is an L_0 regularized regression problem as described in the Background section.

Inventory Management Problem (IMP). In the proposed IMP, we assume the required solutions must (i) choose a delivery schedule from a finite set, (ii) decide an order-up-to amount (where order equals order-up-to minus current inventory) for each scheduled day, and (iii) place costly emergency orders if demand cannot be met with current inventory at any time. We assume there is a requirement to satisfy all demand using either *planned schedules*, or more costly *just-in-time emergency orders*. Demand estimates are generated using a forecast model with an error term from an unknown probability distribution.

To model the IMP as a SO mixed-integer problem we introduce the following notation: let T be the set of days t, R the set of scenarios r, S the set of schedules s, holding cost of an item (per unit-of-measure, per day) h, cost of emergency services (per unit) e, penalty applied to over-stocking (per unit over-stocked) q, and fixed cost of a schedule f_s . The decision space is defined by seven sets

of variables, some of which are: the units of holding space required to stock the inventory $p_{tr} \in \mathbb{Z}^+$, the required emergency order quantity $o_{tr} \in \mathbb{Z}^+$, and the number of units that inventory is over-filled by $v_{tr} \in \mathbb{Z}^+$ (all defined $\forall t \in T, \forall r \in R$). The general formulation of our IMP is:

$$\min \sum_{\forall s \in S} (u_s f_s) + \frac{1}{R} \sum_{\forall r \in R} \sum_{\forall t \in T} (p_{tr} h + o_{tr} e + v_{tr} q) \quad (19)$$

subject to several constraints. Of primary importance is a set of constraints that link the volume of inventory on hand (in each SP) to the scheduling and order-up-to decisions (MP). The resultant decomposition consists of a master problem objective

$$\min \sum_{\forall s \in S} (u_s f_s) + \sum_{\forall r \in R} \theta_r \quad (20)$$

and sub-problem objective

$$\min \sum_{\forall t \in T} (p_{tr} h + o_{tr} e + v_{tr} q), \forall r \in R \quad (21)$$

The SP decisions are constrained by MP scheduling and order-up-to decisions, and we take its dual to generate cuts on θ_r in the MP. See Mana et al. (2024) for full details.

As previously mentioned, we leverage policies learned via RL as our Surrogate MPs for both problems. In order to do so, we cast the problems into MDP formulations, and train neural networks (multi-layer perceptrons) as our policy and value functions. The policy networks return log-odds for discrete actions from which stochastic actions are sampled. We now discuss the formulation for the IMP problem, see Mana et al. (2024) for further details. The MDP transitions by selecting a schedule first, and subsequently setting order quantities as required by the schedule, reaching termination at the end of the temporal horizon. The state space includes balance, capacity, current day, cost parameters, forecasted demand (mean and standard deviation) per day, residuals (between forecasted and actual demand) per day, schedule selected, previous order quantities, and which day the agent is ordering for. The action space includes the schedule decision, and the order quantity decision. Note that there is a hierarchical structure to our IMP problem, in that a schedule decision must be made at the beginning of the horizon, and subsequently only order quantity decisions are to be made for each day within the horizon. The reward function is given by the negative of the cost defined in (19). We use action masking on the policy network outputs in order to enforce constraints on scheduling and order quantity decisions as required during MDP transitions.

We perform experiments on 153 independent cases of our IMP using real-world data. Each experiment was performed with the following parameters: 500 scenarios ($R = 500$), 28 day horizon ($T = 28$), and 169 possible schedules ($S = 169$). The baseline implementation of BD includes accelerations such as scenario group cuts (Adulyasak et al. (2015)) and partial decomposition (Crainic et al. (2016)). We do not compare against a generic implementation of BD due to tractability limitations.

Regularized Regression (RR). In this problem, the objective is to minimize the sum of a convex loss function and the penalized support of a feature set. We follow the cutting-plane procedure for L_0 regularized regression outlined in the Background section, computing $\nabla g(\beta)$ in closed form for the selected sparse feature set. In our experiments, we focus on linear regression problems, i.e. $f(X, \beta) = X^T \beta$ in equation 10.

The MDP is formulated to allow for selecting one feature at a time. The state space includes coefficients β for each feature assuming all features were to be used for regression and their corresponding p-values (which do not change as the MDP transitions), as well as coefficients for the currently chosen sparse feature set $\beta|_z$ and the corresponding multi-hot encoding z for the currently chosen features (which do change as the MDP transitions). The action space consists of the categorical distribution for all features. The reward function is given by the change in the MSE of the residuals ($||f(X, \beta|_z) - y||_2^2$) at each step as z grows. We use action masking to mask previously chosen features within an episode. The episode terminates when the increase in explained variance when adding a feature is less than the increase in L_0 penalty for the added feature.

We perform experiments on 250 regularized regression problems using synthetic data $y = X^T \beta + \epsilon$ with Gaussian noise ϵ (data generation process is outlined in Mana et al. (2024)).

Results

We evaluate performance against a baseline for each problem. A modern accelerated version of BD is used as a benchmark for the IMP, and the cutting-plane algorithm we describe in the Background section is used for RR. All MIPs and LPs are solved using the CPLEX commercial solver. Experiments were run on a 36 CPU, 72 GB RAM Linux machine. For every implementation of Surrogate-MP, we deactivate the surrogate after the optimality gap is less than 5%, to focus on retrieval of optimality using the MIMP.

IMP. All three selection methods, when ran with $\Gamma = 0.75$, produced faster convergence than the baseline model: weighted selection performed 14.96% faster than the baseline, greedy selection achieved 19.43% faster performance, and informed selection performed 30.45% faster. Furthermore, with informed selection Surrogate-MP performed faster on over 88% of instances (convergence rates in Figure 4, instances of faster convergence in Table 1). To further investigate the strong performance of informed selection, we experimented with $\Gamma = 0.25, 0.5,$ and 0.75. $\Gamma = 0.75$ was fastest at 30.45% acceleration, $\Gamma = 0.5$ converged 21.24% faster, and $\Gamma = 0.25$ realized 11.84% faster convergence (Figure 3, Figure 5). Empirically, RL solutions produced results with 14.73% higher cost than the optimal results produced by Benders decomposition at convergence (average across all 153 instances). This indicates the RL heuristic solutions are pseudo-optimal, whilst being significantly faster to obtain.

RR. For $\Gamma = 0.75$, greedy selection resulted in 45.31% acceleration, weighted selection in 44.41%, and informed selection in 37.97%. It is worth noting that RR is a less partially observable MDP than the IMP is. This stronger observability may explain why the RL surrogate benefits

	Surrogate-MP	No Surrogate
Inventory Management	**135 (88.24%)**	18 (11.76%)
Regularized Regression	**214 (85.60 %)**	36 (14.40%)

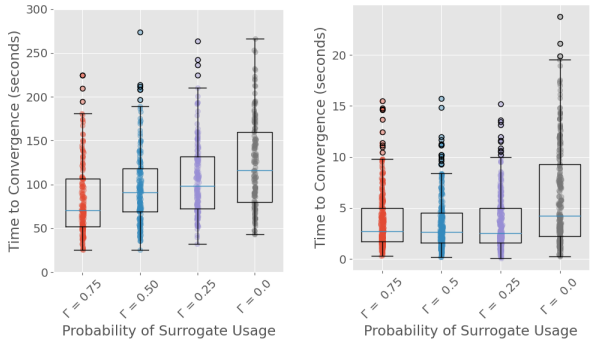
Table 1: Instances of faster convergence when using RL-surrogate with the best performing configuration (Γ and selection method), compared to baseline of no surrogate.

less from information sharing via the constraint matrix A_r. The convergence rates for each selection method are shown in Figure 4. For consistency, we vary the Γ parameter with informed selection and inspect its impact on convergence. In contrast with the IMP, we observe $\Gamma = 0.5$ results in fastest convergence, accelerating RR by 42.94% (Figure 3). For RR, varying Γ produced much less variance in convergence than in the IMP. Similar to the IMP experiments, we observe Surrogate-MP has outperformed the baseline algorithm across most RR instances: Surrogate-MP achieved better convergence rates on up to 85.60% of instances (Table 1). Lastly, we compare L_0 vs L_1 regularization with an L_1 penalty (λ) of 0.1 and 0.5. The two L_1 parameter settings are intended to achieve accurate parameter estimation, and effective coefficient recovery (respectively). With $\lambda = 0.1$ for both L_1 and L_0, L_0 yields significantly improved coefficient recovery (by 81%), and parameter estimation (by 34%), while L_1 yields marginally better MSE (by 2%). With $\lambda = 0.5$, L_1's coefficient recovery improves, but both MSE and parameter estimation degrade considerably. These results display the benefits of L_0 regularization with respect to unbiased feature reduction. Metric definitions and a table with full results can be found in Mana et al. (2024).

Conclusion & Future Work

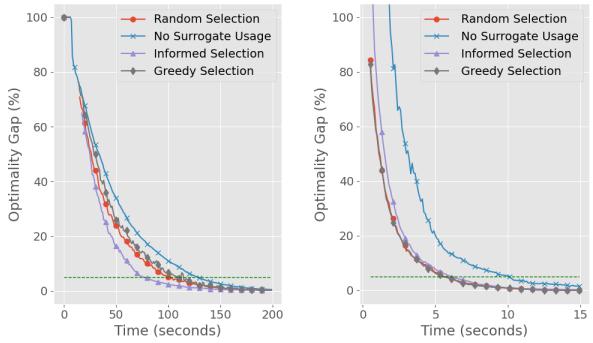
We use RL to learn surrogates in place of MIMPs for accelerating cutting-plane algorithms, achieving drastic reduction in convergence times. Our proposed method is application agnostic, retrieves certificates of optimality, and can utilize any surrogate capable of generating MP solutions. We provide formulations in two different domains – Benders decomposition applied to inventory management and a cutting-plane algorithm for L_0 regularized regression – and provide results showing superiority of our approach in 88.24% and 85.60% of instances with a 30.45% and 45.31% reduction in average run-time respectively.

A promising direction for future work would be to design stronger integration between the surrogate, SP, and MP. Our informed selection method is a first step in this direction, and realized promising results. Some additional opportunities we leave unexplored would be to directly inform the surrogate on the strength of past solutions, offer sub-gradient information as a feature, or redesign the surrogate's objective function to reward the strength of subsequent cuts as opposed to mirroring the MP objective directly. We are additionally eager to observe the performance of Surrogate-MP on other cutting-plane algorithms and optimization problems.



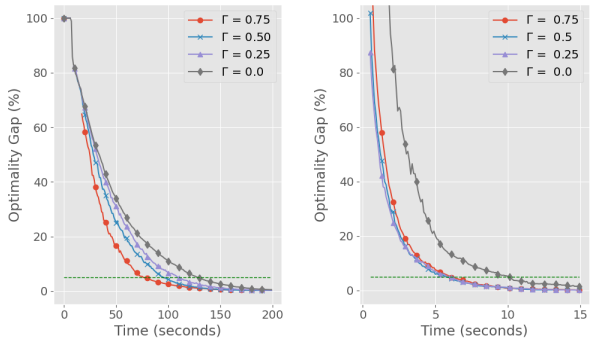
(a) Inventory Management (b) Regularized Regression

Figure 3: Convergence instances of BD accelerated by an informed Surrogate-MP, with different surrogate usages.



(a) Inventory Management (b) Regularized Regression

Figure 4: Convergence rates of a baseline BD, and Surrogate-MP (greedy, weighted, informed) and $\Gamma = 0.75$.



(a) Inventory Management (b) Regularized Regression

Figure 5: Convergence rates of BD accelerated by an informed Surrogate-MP with different surrogate usages. Surrogate-MP deactivated at 5% as indicated by dotted line.

Acknowledgments

We thank Alec Koppel for helpful comments on this manuscript, and anonymous reviewers of this submission.

References

Adulyasak, Y.; Cordeau, J.-F.; and Jans, R. 2015. Benders Decomposition for Production Routing Under Demand Uncertainty. *Operations Research*, 63(4): 851–867.

Baena, D.; Castro, J.; and Frangioni, A. 2020. Stabilized Benders Methods for Large-Scale Combinatorial Optimization, with Application to Data Privacy. *Management Science*, 66(7): 3051–3068.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4: 238–252.

Bertsimas, D.; King, A.; and Mazumder, R. 2016. Best subset selection via a modern optimization lens.

Bonami, P.; Biegler, L. T.; Conn, A. R.; Cornuéjols, G.; Grossmann, I. E.; Laird, C. D.; Lee, J.; Lodi, A.; Margot, F.; Sawaya, N.; et al. 2008. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete optimization*, 5(2): 186–204.

Crainic, T. G.; Rei, W.; Hewitt, M.; and Maggioni, F. 2016. *Partial Benders decomposition strategies for two-stage stochastic integer programs*, volume 37. CIRRELT.

Daducci, A.; Van De Ville, D.; Thiran, J.-P.; and Wiaux, Y. 2014. Sparse regularization for fiber ODF reconstruction: From the suboptimality of $\ell 2$ and $\ell 1$ priors to $\ell 0$. *Medical Image Analysis*, 18(6): 820–833.

Delarue, A.; Anderson, R.; and Tjandraatmadja, C. 2020. Reinforcement learning with combinatorial actions: An application to vehicle routing. *Advances in Neural Information Processing Systems*, 33: 609–620.

Fan, J.; Lv, J.; and Qi, L. 2011. Sparse high-dimensional models in economics. *Annu. Rev. Econ.*, 3(1): 291–317.

Idé, T.; Kollias, G.; Phan, D.; and Abe, N. 2021. Cardinality-regularized hawkes-granger model. *Advances in Neural Information Processing Systems*, 34: 2682–2694.

Jeroslow, R. C. 1973. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21(1): 221–224.

Lee, M.; Ma, N.; Yu, G.; and Dai, H. 2021. Accelerating Generalized Benders Decomposition for Wireless Resource Allocation. *IEEE Transactions on Wireless Communications*, 20(2): 1233–1247.

Louizos, C.; Welling, M.; and Kingma, D. P. 2018. Learning Sparse Neural Networks through L_0 Regularization. In *International Conference on Learning Representations*.

Magnanti, T. L.; and Wong, R. T. 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3): 464–484.

Mana, K.; Acero, F.; Mak, S.; Zehtabi, P.; Cashmore, M.; Magazzeni, D.; and Veloso, M. 2024. Accelerating Cutting-Plane Algorithms via Reinforcement Learning Surrogates (Extended Version). arXiv:2307.08816.

Mazyavkina, N.; Sviridov, S.; Ivanov, S.; and Burnaev, E. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134: 105400.

Nair, V.; Bartunov, S.; Gimeno, F.; Von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2020. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.

Parker, R. G.; and Rardin, R. L. 2014. *Discrete optimization*. Elsevier.

Poojari, C.; and Beasley, J. 2009. Improving benders decomposition using a genetic algorithm. *European Journal of Operational Research*, 199(1): 89–97.

Rahmaniani, R.; Crainic, T. G.; Gendreau, M.; and Rei, W. 2017. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3): 801–817.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Solla, S.; Leen, T.; and Müller, K., eds., *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Tang, Y.; Agrawal, S.; and Faenza, Y. 2020. Reinforcement learning for integer programming: Learning to cut. In *International conference on machine learning*, 9367–9376. PMLR.

Wang, Z.; Li, X.; Wang, J.; Kuang, Y.; Yuan, M.; Zeng, J.; Zhang, Y.; and Wu, F. 2023. Learning Cut Selection for Mixed-Integer Linear Programming via Hierarchical Sequence Model. In *The Eleventh International Conference on Learning Representations*.

Zou, H.; and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2): 301–320.