IJCAI 2025
MONTREAL

Balans: Bandit-based Adaptive LArge Neighborhood Search for Mixed-Integer Programming Problems

International Joint Conference on Artificial Intelligence, IJCAI, Canada, 2025

Junyang Cai[1], Serdar Kadıoğlu[2,3], Bistra Dilkina[1]

[1]Dept. of Computer Science, Univ. of Southern California

[2] Dept. of Computer Science, Brown University

[3] AI Center of Excellence, Fidelity Investments

skadio.github.io

USC

AI Institute for Advances in Optimization

BROWN

Combinatorial Optimization

Mixed-Integer Programming (MIP) is applicable to important combinatorial optimization problems, and hence, improving the efficiency of MIP solving is of great practical and theoretical interest.

$$f(x) = \min c^T x \mid Ax \leq b, x \in \mathrm{R}^n, x_j \in \mathrm{Z}, \forall j \in I$$

Exact Methods

Branch-and-Bound and its extensions, are at the core of solving MIPs to optimality.

Meta-Heuristic Methods

When proving optimality is beyond reach, metaheuristics, such as **Large Neighborhood Search (LNS)**, offer an attractive alternative

Large-Neighborhood Search

Initial Solution

Generate an initial feasible solution $\mathbf{x}^0$ typically by running BnB for a short time.

Destroy Operation

Given solution $\mathbf{x}^t$ create a sub-MIP by fixing or constraining some variables.

Repair Operation

Re-optimize the sub-MIP to obtain a new solution $\mathbf{x}^{t+1}$

Acceptance Decision

Decide whether to accept $\mathbf{x}^{t+1}$ as the next state based on acceptance criteria.

Adaptive LNS (ALNS)

Extends LNS by providing **multiple destroy heuristics** to choose from at each iteration.

The key challenge: how to **select the most effective neighborhood** for each iteration?

Learning-based methods in MIP solving

Learning to Branch

Bengio et al. 2021; Khalil et al. 2016
Liberto et al., 2016; Cai et al.2024a

Node Selection

He et al. 2014

Solution Prediction

Kadioglu et al., 2017;
Ding et al., 2020; Cai et al., 2024b

ML-Enhanced LNS

LNS(MIP) Song et al. 2020
IL-LNS Sonnerat 2021, RL-LNS Wu 2021
CL-LNS Huang et al. 2023b

Heuristic Scheduling

Khalil et al., 2017; Chmiela et al., 2021;
Hendel, 2022

Algorithm Configuration

Kadioglu et al., 2009

Limitations of Learning-Based Approaches

Significant drawback of learning-based methods is their **heavy dependency on offline training.**

Computational Cost

Training is costly and requires carefully curating training data with desired properties and distributions.

Limited Generalization

Offline methods have limited generalization to unseen larger instances.

Circular Dependency

Training often depends on using exact solvers in the first place to create the supervised datasets.
This defeats the purpose of solving for hard instances.

Domain Adaptation

Adapting offline learning-based methods to new distributions and domains remains a challenge.

Our Focus: Online Adaptive Methods

On-the-fly, online learning approaches to MIP solving that **do not depend on any offline training.**

Online Adaptive Methods

Our focus is on approaches that **eliminate the dependency** on **offline training**, enabling dynamic adaptation during the solving process.

Leveraging LNS(MIP) Success

Recent success of embedding **MIP solver within LNS** outperform default solver, with specific neighborhoods like local branching relaxation.

Introducing ALNS(MIP)

A meta-solver utilizing **Adaptive LNS** with diverse neighborhood definitions, dynamically managed by a **online learning** policy.

The Challenge of Online Learning

Online learning is non-trivial, but given the significant drawbacks of offline learning, it must be tackled on the fly, **adaptively for the specific instance** at hand.

Multi-Armed Bandit Approach

We show how to cast this as a multi-armed bandit problem that treats adaptive neighborhoods as different arm choices with unknown reward distributions to be estimated during the search.

Multi-Armed Bandits (MAB)

Bandit algorithms solve online sequential decision-making problems:

- Each **arm** represents a decision that generates a **reward**
- The agent faces the exploration-exploitation dilemma:
  - **Exploit:** Use arm with highest expected reward
  - **Explore:** Try new arms to learn more
- The goal is to maximize cumulative reward over time
- Arm rewards are estimated from past decisions using a **learning policy**

Ideal for our setting as it can learn effective strategies on-the-fly without requiring offline training.

Common MAB learning policies:

- **ε-Greedy:** Select best arm with probability 1-ε, random arm with probability ε
- **Softmax:** Select arms with probability proportional to their estimated values
- **Thompson Sampling:** Sample from posterior distributions of estimated rewards

BALANS

Bandits-based Adaptive Large Neighborhood Search

A novel online meta-solver for MIPs that combines:

Mixed-Integer Programming

Powerful modeling paradigm for combinatorial optimization problems

Adaptive Large Neighborhood Search

Meta-heuristic that iteratively destroys and repairs parts of a solution

Multi-Armed Bandits

Online learning algorithm that balances exploration and exploitation

github.com/skadio/balans

pip install balans

Main Contributions

1

Significant Performance Improvements

We show that the performance of our bandit-based ALNS(MIP), carefully implemented in our Balans solver, significantly improves the default MIP solver SCIP, outperforms single LNS(MIP) and improves over the state-of-the-art LNS(MIP) on hard instances.

2

Adaptive Neighborhood Exploration

We show that our bandit-based ALNS(MIP) rarely depends on the single best neighborhood and instead improves the performance by exploring and sequencing other weaker neighborhoods.

3

Ablation Studies

Balans as a meta-solver is highly different from scheduling heuristics within the BnB tree

Balans is solver agnostic by performing the same set of experiments on a different MIP solver Gurobi.

4

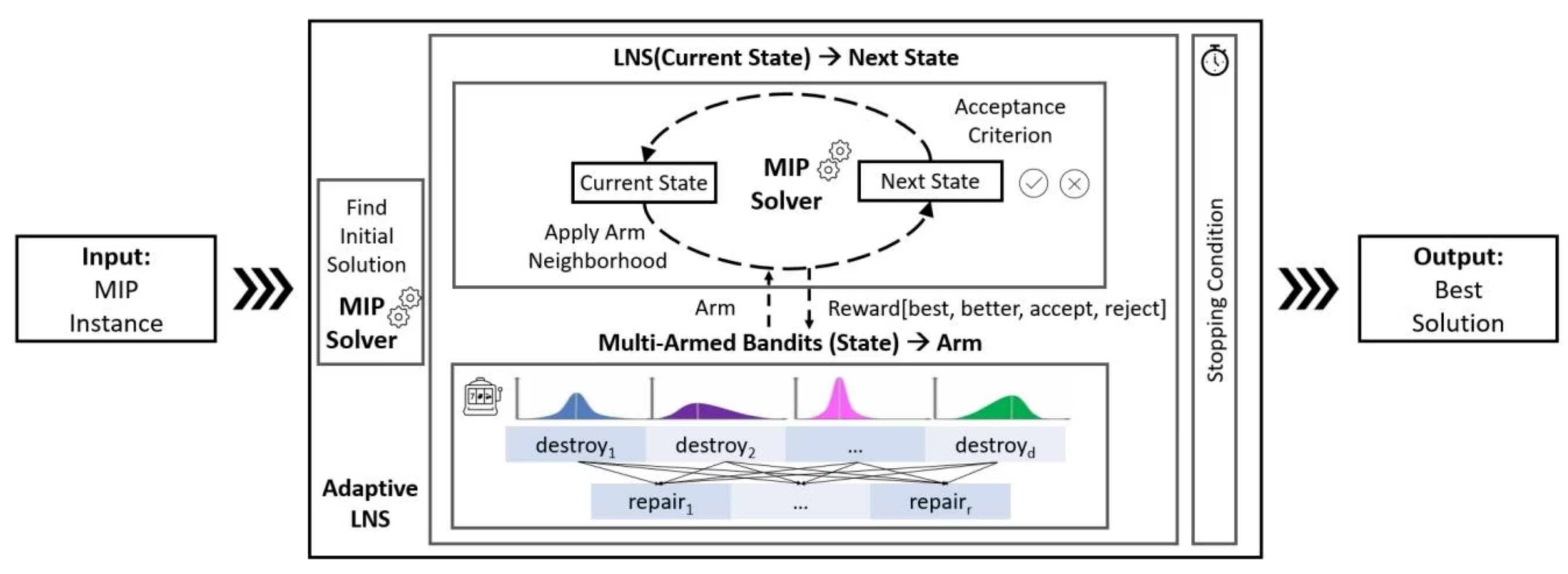
Open-Source Library

We release Balans as an open-source meta-solver for conducting ALNS(MIP) available to others with a one-liner from PyPI.

github.com/skadio/balans

pip install balans

Balans: Online Meta-Solver for MIPs



Initial Solution

Given an MIP instance as input, we first find an initial solution by running an MIP solver with a time limit to find the first feasible solution.

Model Instantiation

Instantiate a *single* MIP model is maintain ed throughout the search. The LP relaxation at the root node is saved with other information.

ALNS Loop

The initial solution yields the current state to start operating ALNS. ALNS is a combination of LNS guided by MAB to adapt to diverse operators.

Destroy Operators

Balans implements **eight built-in destroy operators** with diverse characteristics:

Crossover [Rothberg, 2007]

Generates a random feasible solution and compares with previous state. If discrete vars have the same value, fix them.

DINS [Ghosh, 2007]

Uses LP relaxation and previous solution to bound variables with significant differences.

Local Branching [Fischetti and Lodi, 2003]

Allows only a limited number of binary variables to flip by adding a constraint to the original MIP.

Mutation [Rothberg, 2007]

Fixes a subset of discrete variables to their values from the previous state.

Proximity Search [Fischetti and Monaci, 2014]

Finds a feasible solution with better objective that is as close as possible to the previous solution.

Random Objective

Explores the feasible region randomly by replacing the objective function with random coefficients.

RENS [Berthold, 2014]

Fixes variables with integer LP relaxation values and restricts fractional variables to rounded values.

RINS [Danna et al., 2005]

Compares LP relaxation with previous solution and fixes variables with matching values.

Operator Characteristics

Our operators create a **diverse portfolio** with **complementary characteristics**:

Problem Type Coverage

Some operators work on specific MIP subfamilies (e.g., binary or integer only), while others are general. When combined together, we cover all subfamilies of MIP problems.

No Tuning Required

Unlike the previous work, we do not need to tune destroy size parameters. We simply introduce the same operator multiple times with varying destroy sizes as different options in portfolio.

Distinct Approaches

Each operator has a unique focus, rather than mixing different flavors. Our online learning sequences these distinct operators to obtain effective hybrid behavior for each instance.

Comparison with the state-of-the-art LNS (MIP)

Local Branching Relaxation

- The state-of-the-art LNS(MIP) approach from [Huang et al., 2023a] has a **hyper-parameter** to control the **destroy size** that must be chosen carefully **for each problem domain**.
- In addition, the initial destroy size is then dynamically adjusted during the search according to a **fixed schedule**.
- The hybrid nature of lb-relax combined with a tuned destroy size and its dynamic adjustment is key to its **state-of-the-art** LNS(MIP) performance.

Balans Approach

- Our destroy operators are not mixing different flavors together and are designed to be **distinct** to constitute a diverse portfolio.
- Effective online learning algorithm would be able to **sequence these distinct operators** in a way to obtain the desired hybrid behavior for the instance at hand.
- We do **not need to tune for the destroy size**. We simply introduce the same operator multiple times in our portfolio with varying destroy sizes, serving as different options to choose from during the search.

Online Learning

State Exploration

- Decides whether the search should continue with the next state or discard the move. We consider two complementary acceptance criteria:

  - **Hill Climbing (HC):** Mostly exploits yet allows the search to progress to next state when the objective value is the same.
  - **Simulated Annealing (SA):** Offers more exploration capacity and allows the search to move to worsening next states.

Neighborhood Exploration

- Decides the destroy operator to apply at each state. We employ multi-armed bandits for choosing among different neighborhoods with three important design decisions:

  - **Arms:** Every pair of destroy and repair operators as a single arm
  - **Reward mechanism:** Four distinct rewards aligned to possible outcomes of the accept criterion [best, better, accept, reject]
  - **Learning policy:** MAB learns from historical arm choices associated with observed rewards

Distinguishing these two exploration needs

and addressing them separately is a **key novelty** of our approach.

Quick Start Example

```
from balans import Balans, DestroyOperators, RepairOperators
from alns import HillClimbing, MaxRuntime
from mabwiser import MAB, LearningPolicy

# Balans meta-solver for MIPs
balans = Balans(destroy_ops = [DestroyOperators.LocalBranching_10, … ],
                repair_ops = [RepairOperators.Repair],
                selector = MAB([best, better, accept, reject], LearningPolicy.EpsGreedy(eps=0.5)),
                accept = HillClimbing(),
                stop = MaxRuntime(100),
                mip_solver="scip") # "gurobi"

# Solve MIP instance
result = balans.solve(“miplib/routing.mps")

# Best solution and objective
print("Best solution:", result.best_state.solution())
print("Best solution objective:", result.best_state.objective())
```

github.com/skadio/balans

pip install balans

Experiments

Q1: Performance Comparison

- What is the performance comparison between the default MIP, LNS(MIP) that commits to a single neighborhood, the state-of-the-art LNS(MIP), and our ALNS(MIP) using BALANS?
- Can BALANS achieve good performance without any offline training and explore states and neighborhoods simultaneously by adapting to the instance at hand on the fly using bandits?

Q2: Arm Selection Distribution

- How is arm selection among the portfolio of neighborhoods distributed in our bandit strategy?
- Does BALANS depend on the single best neighborhood, or can it improve over the single best by applying weaker operators sequentially in an adaptive fashion?

Datasets

D-MIPLIB [Huang *et al.* 2024]

We select 10 random instances with a total of 50 instances

- Multiple Knapsack
- Set Cover
- Maximum Independent Set
- Minimum Vertex Cover
- Generalized Independent Set Problem

H-MIPLIB [Gleixner *et al.*, 2021]

We consider a subset that permits a feasible solution within 20 seconds, yielding 43 instances.

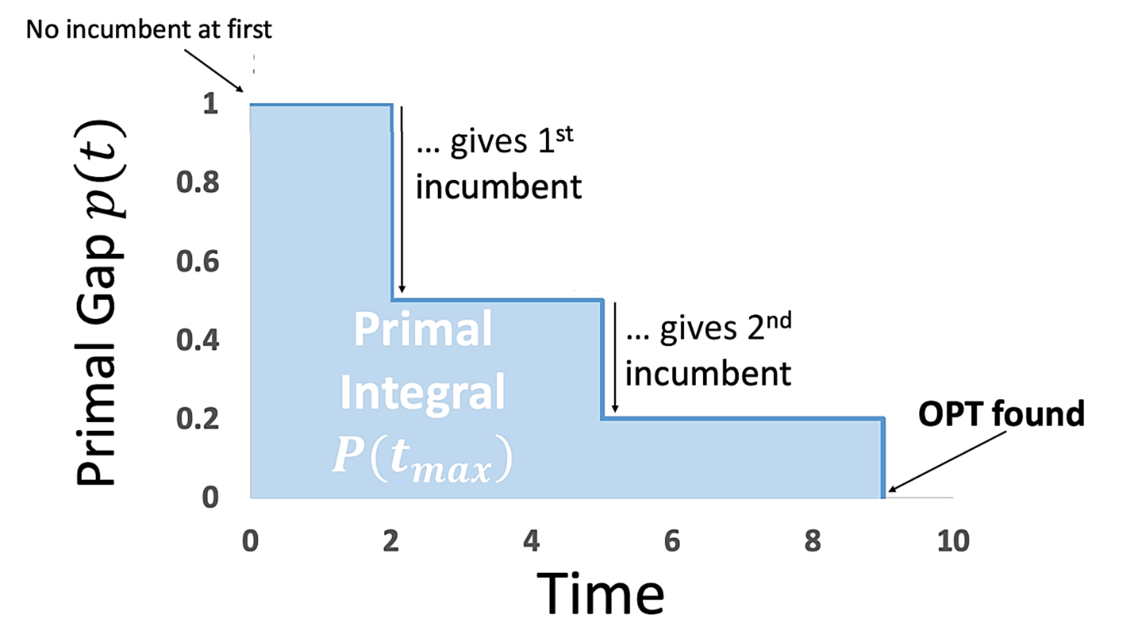
SCIP and Gurobi cannot solve any of these instances to optimality within 1 hour, ensuring the hardness of our benchmarks.

https://huggingface.co/skadio/datasets/balans

Evaluation Metrics & Setup

Primal Gap (PG) [Berthold, 2006] is the normalized difference between primal bound v and precomputed best known obj value v* and is defined as |v−v*|/max(|v*|,ϵ) if v exists and vv∗ ≥ 0.

Primal Integral (PI) [Achterberg *et al.*, 2012] at time q is the integral on [0, q] of the primal gap as a function of runtime. PI captures the quality of and the speed at solutions are found.



Time Limits For initial solution, we run the solver for **20 sec.** Each LNS iteration is limited to **1 minute**, except for Local Branching to **2.5 minutes**, which solves larger sub-problems than other operators. The time limit to solve each instance is set to **1 hour**.

We conduct experiments on AWS EC2 Trn1 with 128 vCPUs and 512GB memory. Balans solver integrates:

- ALNS library [Wouda and Lan, 2023]
- MABWiser library [Strong *et al.*, 2019]
- SCIP (v9.0.0) [Bolusani *et al.*, 2024]
- GUROBI (v11.0.0) [Gurobi, 2024]

Comparisons

Default MIP Solver

We use **SCIP** and **Gurobi**, the state-of-the-art opensource and commercial MIP solvers with default settings running single thread [Bolusani et al., 2024; Gurobi, 2024].

State-of-the-art LNS(MIP)

We use **lb-relax** thanks to the original implementation from [Huang *et al. 2023a*]. This algorithm selects the neighborhood with the local branching relaxation heuristic.

Single Neighborhood LNS(MIP)

All eight operators are implemented and readily available in BALANS to serve in LNS(MIP). By varying the parameters of these operators, we obtain **16 different destroy operators** from **6 unique neighborhoods**.

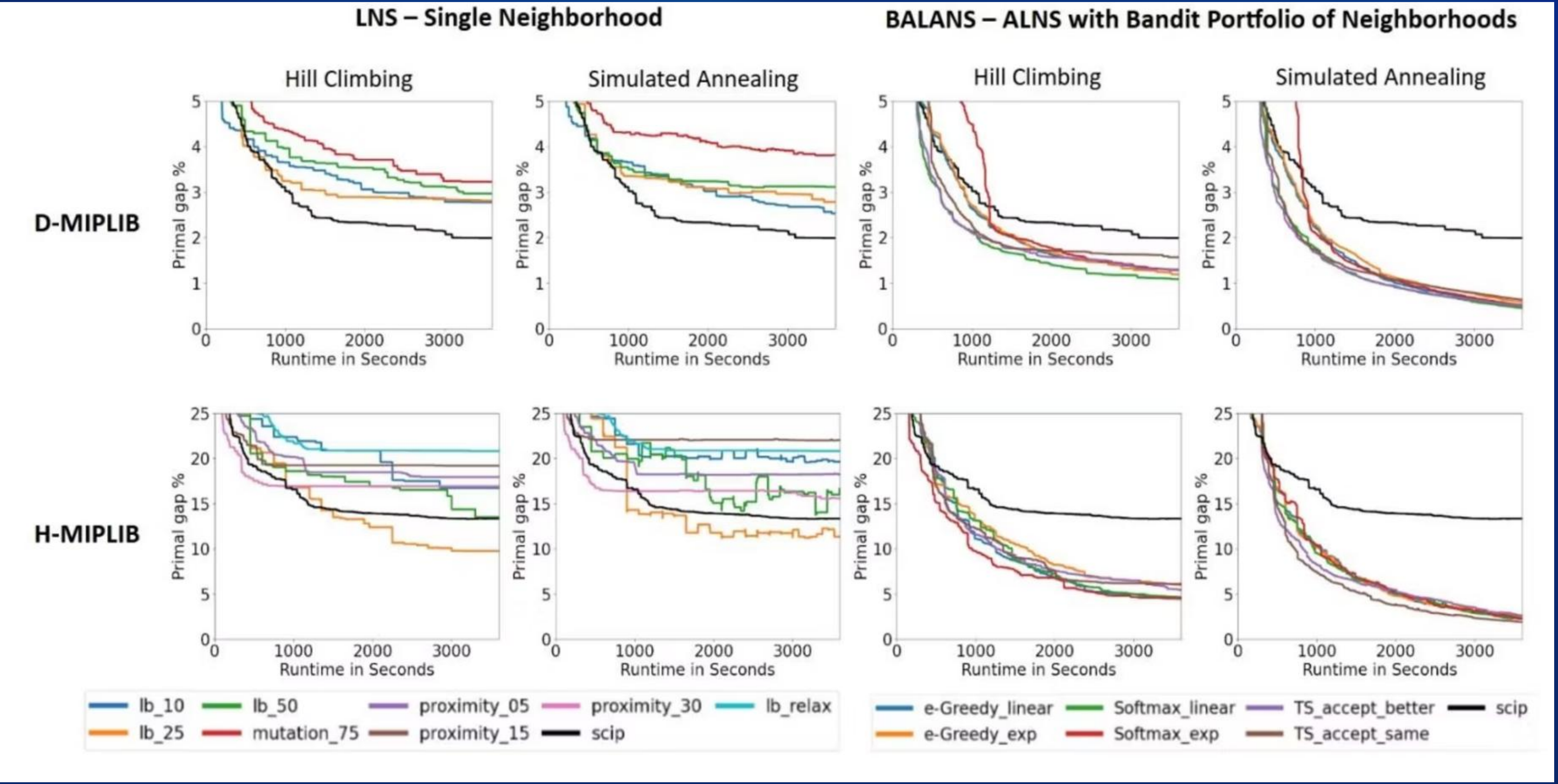
For the accept criterion, we use **HC** and **SA** with an initial temperature set to 20 and an end temperature set to 1 with a step size of 0.1.

Balans ALNS(MIP)

Given the 16 different single destroy operators used in LNS(MIP), we build Balans for ALNS(MIP) with a portfolio that **includes all of the 16 operators**.

For the accept criterion, we again use HC and SA. For the learning policy, we use **e-Greedy** and **Softmax** with numeric rewards and **Thompson Sampling (TS)** with binary rewards.

Q1: Default MIP vs. LNS(MIP) vs. BALANS



Overall Performance

75%+

Primal Gap Reduction

Overall, we reduce the primal gap of SCIP by 75+% across datasets.

50%+

Primal Integral Reduction

We reduce the primal integral of SCIP by 50+% across datasets.

Any Balans configuration is better than SCIP and single LNS, revealing its robust out-of-the-box performance.

Improves SOTA lb-relax method which requires offline training and **solver-agnostic**, similar results with Gurobi.

Q2: Distribution of Arm Selection

	Crossover	Local Branching	Mutation	Proximity	RENS	RINS
D-MIPLIB (Softmax linear SA)	6.4%	11%	25%	17.9%	19.8%	19.9%
H-MIPLIB (TS accept same SA)	9.4%	0.6%	21%	7.3%	31%	30.8%

The single best operator (Local Branching) is **not a popular arm at all** - only 0.6% usage in H-MIPLIB.

RENS and RINS, which perform poorly alone, account for **~40% and ~60%** of usage in D-MIPLIB and H-MIPLIB respectively.

BALANS outperforms MIP and any single best LNS by **using weaker operators sequentially** in an intelligent order.

This demonstrates the **power of adaptive operator selection** through **online learning.**

Our Contributions

No Offline Training

Balans eliminates the need for computationally costly offline training and carefully curated datasets.

Modular Architecture

Leverages best-in-class open-source software for bandits, ALNS, and MIP solving in a highly configurable framework.

Solver Agnostic

Functions as a meta-solver that can be applied on top of any MIP solver, significantly improving their performance.

Adaptive Search

Learns effective strategies on-the-fly for the specific instance at hand, adapting to diverse problem structures.

Balans achieves significant performance improvements over state-of-the-art methods with zero tuning.

Broader Impact

Balans solver subsumes the previous literature on LNS(MIP) when run with a single neighborhood while serving as a highly configurable, modular, and extensible integration technology at the intersection of adaptive search, meta-heuristics, multi-armed bandits, and mixed-integer programming.

Integration Technology

Balans brings together multiple state-of-the-art approaches:

- Adaptive search algorithms
- Meta-heuristics for optimization
- Multi-armed bandits for online learning
- Mixed-integer programming solvers

Key Advantages

This modular design provides several benefits:

- Highly configurable framework
- Extensible design for new neighborhoods
- Solver-agnostic implementation
- No offline training required

MAB Success Stories

Other successful MAB applications for optimization and beyond

Multi-Agent Pathfinding

[Phan *et al.*, 2024] uses MAB with ALNS for multi-agent pathfinding

Multi-Objective Flow Shop

[Almeida *et al.*, 2020] uses MAB with hyper-heuristics for multi-objective flow shop problems

Maximum Satisfiability

[Zheng *et al.*, 2022] uses MAB for maximum satisfiability (MaxSAT)

Personalization & Agents

MAB is heavily used in recommender systems [Kadioglu and Kleynhans, 2024] and game-playing agents [Schaul 2019].

github.com/skadio/mabwiser

pip install mabwiser

Conclusions

Novel Online Meta-Solver
Balans combines multi-armed bandits with adaptive large neighborhood search for effective online learning for MIPs.

Superior Performance
Significant improvements over default MIP solvers, LNS, and SOTA approaches on hard optimization instances.

Effective Online Sequencing
Significant performance from sequencing weaker operators rather than relying on a single best neighborhood.

Open-Source Software
Released Balans as an open-source meta-solver with high-level interface, modular architecture, and configurable design.

github.com/skadio/balans

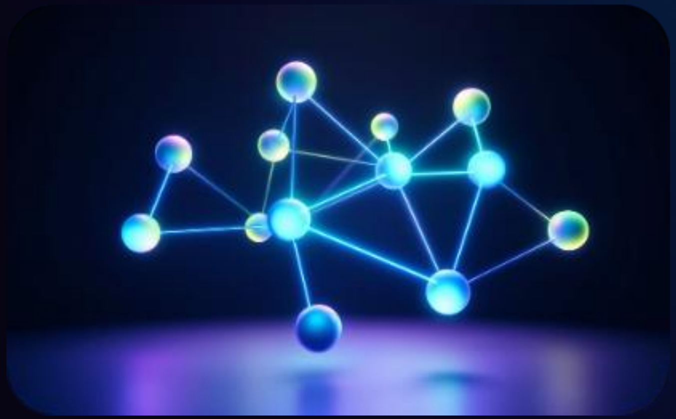
pip install balans

Future Directions

- Boost performance through careful algorithm configuration and portfolio construction
- Explore hybrid ALNS(MIP) approaches that incorporate existing offline training methods as additional arms
- Develop specialized reward and repair mechanisms for different problem domains
- ParBalans: Parallel Multi-Armed Bandits-based Adaptive Large Neighborhood Search

skadio.github.io

AI Center of Excellence @ Fidelity



Optimization & Decision Systems

[AAAI'25, Constraints'24, CP'23, CPAIOR'23, NeurIPS'22] **Text2Zinc** & **Ner4Opt** LLM optimization copilots github.com/skadio/ner4opt

[IJCAI'25, ArXiv'24] **Balans**: Meta optimization solver with online learning github.com/skadio/balans

[ArXiv'24] **iCBS**: Pruning LLVMs using combinatorial optimization github.com/amazon-science/icbs

Explainable & Responsible AI

[AAAI'25, MAKE'23] **BoolXAI** Explainable AI with Boolean formulas github.com/fidelity/boolxai

[ACM'24, LION'23, ICMLA'21] **Jurity** Fairness & bias mitigation github.com/fidelity/jurity

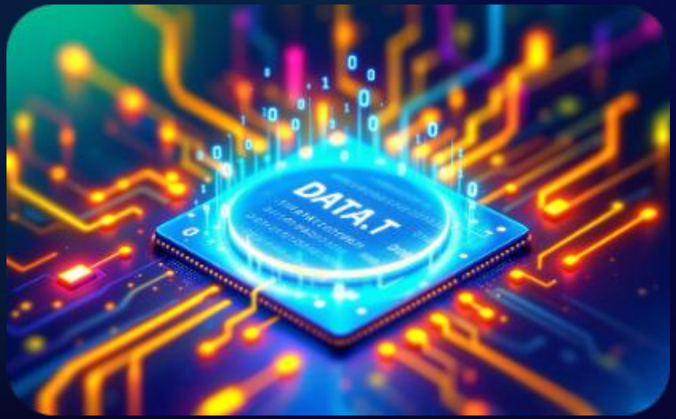
[JDSA'21] **Uncertainty** prediction using Bayesian deep learning

Machine Learning & Recommendations

[AAAI'24, AMAI'24, CIKM'22] **Mab2Rec** Multi-armed bandit recommender systems github.com/fidelity/mab2rec

[TMLR'22, IJAIT'21, ICTAI'19] **MABWiser** Contextual bandits github.com/fidelity/mabwiser

[ACM'23] **Read-Write-Learn** Self-learning for handwriting recognition



Embeddings & Data Processing at Scale

[AI Magazine'23, AAAI'22] **Seq2Pat** Sequential pattern mining github.com/fidelity/seq2pat

[AAAI'21] **TextWiser** NLP/text featurization github.com/fidelity/textwiser

[CPAIOR'22] **Selective** Tabular feature selection github.com/fidelity/selective