ParamILS: An Automatic Algorithm Configuration Framework

Hutter, Hoos, Leyton-Brown, Stützle

JAIR'09

Reading

Comprehensive journal paper on a seminal work

Check website for specific sections to read 👀

Feel free to read more at your own interest 😊

Integer programing and SAT

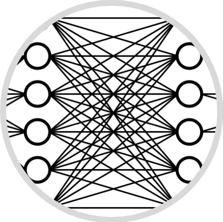
Integer program (IP)

$$\begin{aligned} \max \quad & \boldsymbol{c} \cdot \boldsymbol{z} \\ \text{s.t.} \quad & A\boldsymbol{z} \leq \boldsymbol{b} \\ & \boldsymbol{z} \in \mathbb{Z}^n \end{aligned}$$

SAT

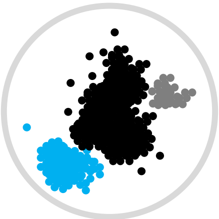
$$(x_1 \vee x_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_8)$$
$$\wedge (x_1 \vee x_8 \vee x_{12}) \wedge (x_2 \vee x_{11})$$

Tons of applications:



Robust ML

MAP estimation



Clustering

Routing

Scheduling

Algorithm configuration

Solvers come with tons of tunable parameters

Tuning by hand is notoriously **slow**, **tedious**, and **error-prone**

Can we **automatically** optimize parameters?

Algorithm configuration pipeline

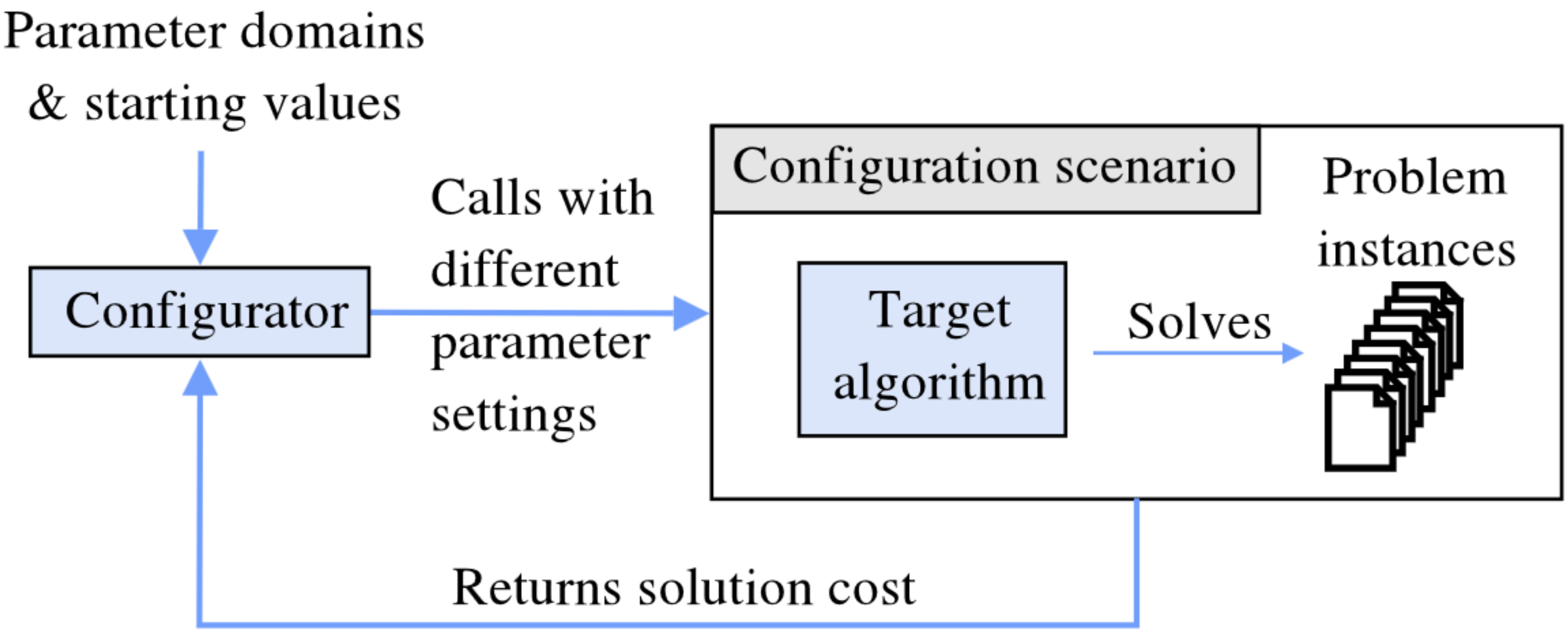
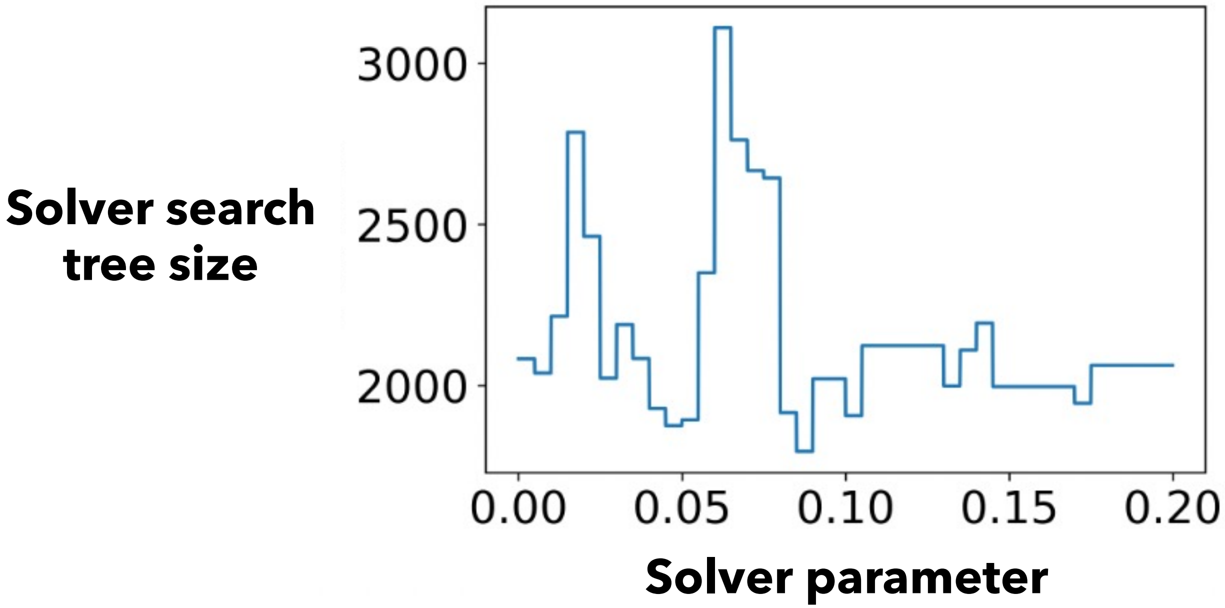


Figure by Hutter et al. [JAIR'09]

Key challenge



- Solver performance is **extremely volatile**
- Gradients are often **uninformative**

Figure by Balcan et al. [NeurIPS’21]

Outline

1. Introduction
2. **Setup**
3. ParamILS
4. Experiments
5. Spectrum auctions

Setup: Parameterized algorithm

Algorithm $\mathcal{A}$ with k parameters

i^{th} parameter setting from a set Θ_i

Assume $|\Theta_i|$ is finite (e.g., by discretizing continuous parameters)

$\boldsymbol{\Theta} \subseteq \Theta_1 \times \cdots \times \Theta_k$ is the set of all feasible configurations

In experiments, $|\boldsymbol{\Theta}|$ as large as $1.38 \cdot 10^{37}$

$\mathcal{A}(\boldsymbol{\theta})$ is the algorithm with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$

Setup: Modeling the application domain

Set of problem instances Π

- E.g., $\pi \in \Pi$ is a routing integer program

Application-specific distribution $\mathcal{D}$ over problem instances, e.g.:

- Distribution over Bay Area routing problems
- Uniform distribution over benchmark dataset

Setup: Measuring performance

$o(\boldsymbol{\theta}, \pi, \kappa)$: runtime of $\mathcal{A}(\boldsymbol{\theta})$ on instance π with runtime cap κ

$\kappa_{\max}$: maximum runtime after which any run will be terminated

$c(\boldsymbol{\theta})$: overall *cost* of running algorithm with parameters $\boldsymbol{\theta}$

- E.g., expected runtime $c(\boldsymbol{\theta}) = \mathbb{E}_{\pi \sim \mathcal{D}}[o(\boldsymbol{\theta}, \pi, \kappa_{\max})]$
- If $\mathcal{D}$ is the uniform distribution over a benchmark set Π, equivalent to

$$c(\boldsymbol{\theta}) = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} o(\boldsymbol{\theta}, \pi, \kappa_{\max})$$

Algorithm configuration goal

Goal: Find parameter setting $\boldsymbol{\theta}$ with low cost $c(\boldsymbol{\theta})$

Challenges:

- Distribution $\mathcal{D}$ may be unknown
- Do not know analytical form of c
- c may be nonconvex, non-Lipschitz, …

Empirical cost

Sequence of runs $\mathbf{R} = \left((\boldsymbol{\theta}_1, \pi_1, s_1, \kappa_1, o_1), \ldots, (\boldsymbol{\theta}_n, \pi_n, s_n, \kappa_n, o_n)\right)$

Parameter setting

Problem instance

Random seed

Runtime cap

Observed runtime

Empirical cost

Sequence of runs $\mathbf{R} = \big((\boldsymbol{\theta}_1, \pi_1, s_1, \kappa_1, o_1), \dots, (\boldsymbol{\theta}_n, \pi_n, s_n, \kappa_n, o_n)\big)$

Empirical cost $\hat{c}(\boldsymbol{\theta}, \mathbf{R}) = \underline{\text{average}}(\{o_i : \boldsymbol{\theta} = \boldsymbol{\theta}_i\})$

Could be replaced by another statistic, e.g., median

Goal: Find $\boldsymbol{\theta}$ with low empirical (training) cost $\hat{c}(\boldsymbol{\theta}, \mathbf{R})$

- Ideally, this should lead to low **actual** cost $c(\boldsymbol{\theta})$
- Later this quarter: statistical guarantees for bounding $|c(\boldsymbol{\theta}) - \hat{c}(\boldsymbol{\theta}, \mathbf{R})|$

Key questions

1. Which parameter configurations $\boldsymbol{\Theta}' \subseteq \boldsymbol{\Theta}$ should we evaluate?

2. Which instances $\Pi_{\boldsymbol{\theta}'} \subseteq \Pi$ should we use to evaluate $\boldsymbol{\theta}' \in \boldsymbol{\Theta}'$?

3. Which cutoff times κ_i should we use for each run?

Outline

1. Introduction
2. Setup
3. **ParamILS**
4. Experiments
5. Spectrum auctions

ParamILS

Iterated local search

ParamILS:
A seminal general-purpose algorithm configuration procedure

- Begins with a **default** parameter configuration
- Performs **local search** in configuration space
  - Changes the setting of **one parameter at a time**
  - Keeps those changes resulting in performance improvements
- After finding a **local minimum**:
  Randomly changes some parameters in order to escape

BasicILS

Step 1: Randomly search for good initial configuration

i. $\boldsymbol{\theta}_0$: initial configuration

ii. For $i = 1, \ldots, r$:

   a. $\boldsymbol{\theta} \leftarrow \text{random}(\boldsymbol{\Theta})$

   b. if BETTER$(\boldsymbol{\theta}, \boldsymbol{\theta}_{\mathbf{0}})$, $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}$

Essentially, run $\mathcal{A}(\boldsymbol{\theta})$ on some random instances and compare $\hat{c}(\boldsymbol{\theta}, \mathbf{R})$ and $\hat{c}(\boldsymbol{\theta}_0, \mathbf{R})$

BasicILS

Step 2: Search for better configuration in neighborhood of $\boldsymbol{\theta}_0$

$\boldsymbol{\theta}_{\text{ils}} \leftarrow \textsc{IterativeFirstImprovement}(\boldsymbol{\theta}_0)$

$\textsc{IterativeFirstImprovement}(\boldsymbol{\theta})$:

Repeat:

i. $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$

ii. Find best parameter setting $\underbrace{\boldsymbol{\theta}'' \text{ in the neighborhood of } \boldsymbol{\theta}'}_{\boldsymbol{\theta}'' \text{ differs from } \boldsymbol{\theta}' \text{ in one component}}$ according to $\hat{c}$

BasicILS

Step 2: Search for better configuration in neighborhood of $\boldsymbol{\theta}_0$

$\boldsymbol{\theta}_{\text{ils}} \leftarrow$ ITERATIVEFIRSTIMPROVEMENT$(\boldsymbol{\theta}_0)$

ITERATIVEFIRSTIMPROVEMENT$(\boldsymbol{\theta})$:

Repeat:

i. $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$

ii. Find best parameter setting $\boldsymbol{\theta}''$ in the neighborhood of $\boldsymbol{\theta}'$ according to $\hat{c}$

iii. Set $\boldsymbol{\theta} = \boldsymbol{\theta}''$

Until $\boldsymbol{\theta}' = \boldsymbol{\theta}$ $\Rightarrow$ found a local minimum

BasicILS

Step 3 (repeat as many times as you can)**:**

1. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\text{ils}}$
2. for s rounds do random exploration:

   $\boldsymbol{\theta} \leftarrow$ random $\boldsymbol{\theta}'$ in the neighborhood of $\boldsymbol{\theta}$
3. $\boldsymbol{\theta} \leftarrow$ IterativeFirstImprovement($\boldsymbol{\theta}$)
4. if BETTER($\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{ils}}$), set $\boldsymbol{\theta}_{\text{ils}} \leftarrow \boldsymbol{\theta}$
5. With some small probability, restart: $\boldsymbol{\theta} \leftarrow$ random($\boldsymbol{\Theta}$)

Return best $\boldsymbol{\theta}$ the algorithm ever found according to $\hat{c}$

Adaptive capping

Solving IPs can take forever…

- We want to give up as early as we can
- But still correctly evaluate BETTER($\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$)

Without a good way to **cap** runs, will waste time on bad $\boldsymbol{\theta}$'s

Adaptive capping

Illustrative example:

- $\boldsymbol{\theta}_1$ takes 10 seconds total to solve 100 instances
$$\hat{c}(\boldsymbol{\theta}_1, \mathbf{R}) = \frac{1}{100} \cdot 10 = 0.1$$
- $\boldsymbol{\theta}_2$ takes at least 11 seconds to solve the first instance
$$\hat{c}(\boldsymbol{\theta}_2, \mathbf{R}) \geq \frac{1}{100} \cdot (11 + \underbrace{0 + 0 + \cdots 0}_{\textbf{99 zeros}}) = 0.11$$

- Can stop evaluating $\boldsymbol{\theta}_2$ 11 seconds into the first run

Simple adaptive capping

BETTER($\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$):

1. Evaluate $\boldsymbol{\theta}_1$ on N random instances to compute $\hat{c}(\boldsymbol{\theta}_1, \mathbf{R})$
2. Evaluate $\boldsymbol{\theta}_2$ on N random instances to compute $\hat{c}(\boldsymbol{\theta}_2, \mathbf{R})$
   But **give up** after $N \cdot \hat{c}(\boldsymbol{\theta}_1, \mathbf{R})$ seconds and return $\boldsymbol{\theta}_1$
3. If didn't give up, return

$$\begin{cases} \boldsymbol{\theta}_1 & \text{if } \hat{c}(\boldsymbol{\theta}_1, \mathbf{R}) < \hat{c}(\boldsymbol{\theta}_2, \mathbf{R}) \\ \boldsymbol{\theta}_2 & \text{else} \end{cases}$$

Many ways **cap more aggressively** and improve this further

Outline

1. Introduction
2. Setup
3. ParamILS
4. **Experiments**
5. Spectrum auctions

Experiments: example

Average runtime

Scenario	SimpleLS(100) Performance	BasicILS(100) Performance	BasicILS(100) Avg. # ILS iterations	p-value
SAPS-SWGCP	0.5 ± 0.39	$\mathbf{0.38 \pm 0.19}$	2.6	$\mathbf{9.8 \cdot 10^{-4}}$
SAPS-QCP	3.60 ± 1.39	$\mathbf{3.19 \pm 1.19}$	5.6	$\mathbf{4.4 \cdot 10^{-4}}$
SPEAR-QCP	0.4 ± 0.39	$\mathbf{0.36 \pm 0.39}$	1.64	$\mathbf{0.008}$

- SAPS and SPEAR: SAT solvers
- SWGCP: graph coloring problems
- QCP: quasi-group completion problem
- SimpleLS: local search w/o randomization
  - Should get stuck in local minima

Outline

1. Introduction
2. Setup
3. ParamILS
4. Experiments
5. **Spectrum auctions**

Spectrum auctions

Later work by the same UBC lab (and others):

- In 2016–17, FCC held an auction to repurpose radio spectrum
  - Broadcast television → wireless internet
  - In total, the auction yielded $19.8 billion

Kevin-Leyton Brown
UBC

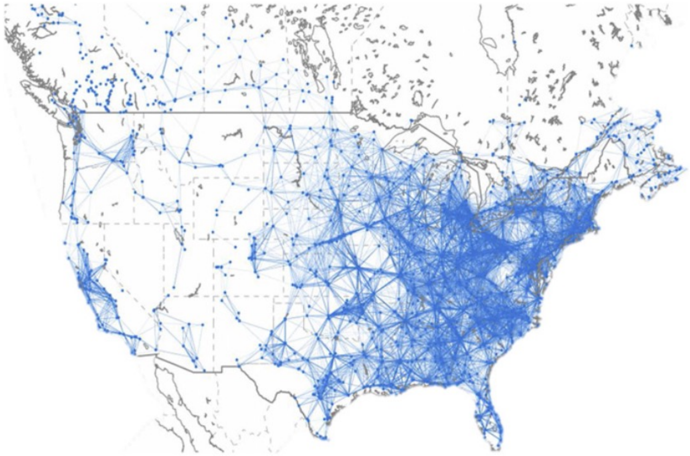
Ilya Segal
Stanford

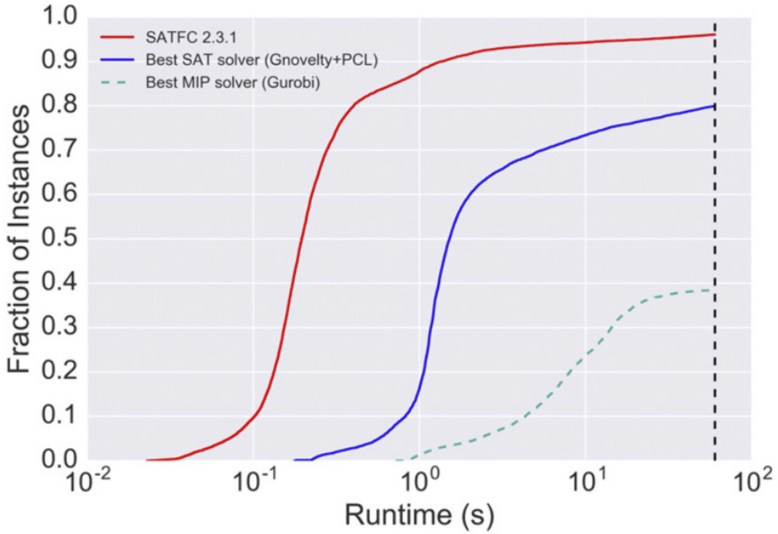
Paul Milgrom
Stanford

And many others!

Spectrum auctions

- The auction involves solving huge graph coloring problems





- SATFC uses algorithm configuration + portfolio selection
- Simulations indicate SATFC saved the government billions

Leyton-Brown et al., PNAS'17; Leyton-Brown and Hutter, ICML'19 tutorial

Overview

ParamILS: a seminal general-purpose configuration procedure

Combines **local search** with **random exploration**

Speedups for **IP** and **SAT** solvers

Inspired later breakthroughs for **spectrum auctions**