Empirical Model Learning

Merging knowledge-based and data-driven decision models through machine learning

Michela Milano

Dipartimento di Informatica – Scienza e Ingegneria

Università di Bologna

Cesena – 12 Luglio 2019

Joint work with…..

Michele Lombardi 

Alessio Bonfietti 

Andrea Bartolini 

Tias Guns 

Luca Benini 

Pascal van Hentenryck 

Machine learning and optimization

- A lot of research activity going on in two directions:

- Helping ML algorithms through optimization techniques
  - survey in [Bessiere et al. , 2016]

- Boosting optimization via ML
  - this is what this talk is about

Boosting optimization via ML

- Boosting optimization via ML:
- ML for modeling

  ConAq [Bessiere et at 2017], Model Seeker [Beldiceanu and Simonis, 2012], using intermediate language [Lallouet et al. 2010], [Lallouet and Legtchenko, 2007], [Bessiere et al. , 2017b]

- ML for search

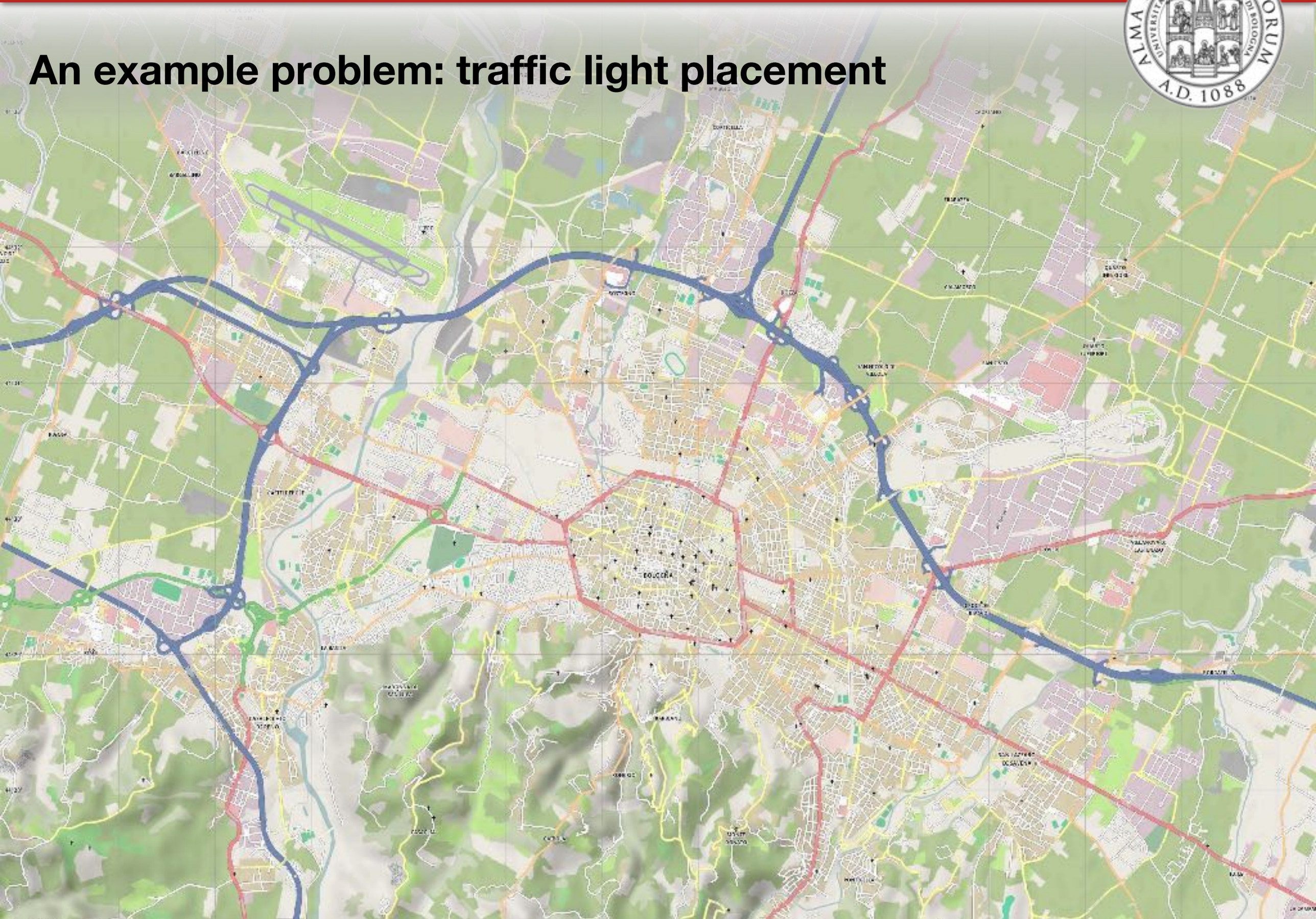
  See the tutorial of Bistra Dilkina at the Master Class
  see also [Mairy, 2011], [Bello et al. , 2016], [Hottung et al. , 2017]

- ML for portfolio selection and algorithm configuration

  [Hutter et al. , 2011], [Hutter et al. , 2014], [Xu et al. , 2008], [Malitsky and Sellmann, 2012]

Motivating Examples

An example problem: traffic light placement



An example problem: traffic light placement

- Add/remove traffic light over a whole city
- Traffic lights can be connected (green wave)
- Every operation has a cost (and there is a budget)
- Probably other constraints (not considered)
- Improve traffic figures

HP: we want to solve it! What is the toughest obstacle?

An example problem: traffic light placement

- Add/remove traffic light over a whole city
- Traffic lights can be connected (green wave)
- Every operation has a cost (and there is a budget)
- Probably other constraints (not considered)
- Improve traffic figures

HP: we want to solve it! What is the toughest obstacle?

Assessing the effect of traffic light placement on the traffic levels:

Impossible via expert-designed declarative models

Human behaviour – complex system

Assessing the effect of traffic light placement on the traffic levels

Traffic light placement: decidables

Traffic levels: observables

WE WANT TO LEARN THE IMPACT OF DECIDABLES ON OBSERVABLES

and

CAST THIS INTO AN OPTIMIZATION MODEL

A second example: policies for PV adoption

A second example: policies for PV adoption

- Given a target of PV adoption to be achieved
- Decide which incentives to give
- Every incentive has a cost (and there is a budget)
- Probably other constraints (not considered)

HP: we want to solve it! What is the toughest obstacle?

A second example: policies for PV adoption

- Given a target of PV adoption to be achieved
- Decide which incentives to give
- Every incentive has a cost (and there is a budget)
- Probably other constraints (not considered)

HP: we want to solve it! What is the toughest obstacle?

Assessing the effect of incentives on PV adoption:

Impossible via expert-designed declarative models

Human behaviour – Social Dynamics

Assessing the effect of incentives on PV adoption

Incentives: decidables

PV adoption levels: observables

WE WANT TO LEARN THE IMPACT OF DECIDABLES ON OBSERVABLES

and

CAST THIS INTO AN OPTIMIZATION MODEL

A third example: thermal-aware workload dispatching

A third example: thermal-aware workload dispatching

- Given a target heterogeneous platform and a workload
- Decide workload (tasks) dispatching
- Every task has a thermal dynamics (and there is a thermal controller)
- Load balancing constraints
- **With thermal limits**

A third example: thermal-aware workload dispatching

- Given a target heterogeneous embedded platform and a workload
- Decide task allocation and scheduing
- Every task has a thermal dynamics (and there is a thermal controller)
- Probably other constraints (not considered)

HP: we want to solve it! What is the toughest obstacle?

A third example: thermal-aware workload dispatching

Given a target heterogeneous embedded platform and a workload

Decide task allocation and scheduing

Every task has a thermal dynamics (and there is a thermal controller)

Probably other constraints (not considered)

HP: we want to solve it! What is the toughest obstacle?

Assessing the effect of workload allocation on thermal dynamics:

Possible with simulation based on differential equations of the thermal dynamics BUT not manageable by combinatorial optimization techniques

Assessing the effect of workload allocation on thermal dynamics:

Workload allocations: decidables

Core temperatures: observables

WE WANT TO LEARN THE IMPACT OF DECIDABLES ON OBSERVABLES

and

CAST THIS INTO AN OPTIMIZATION MODEL

What do these problems have in common?

A strong combinatorial structure

Part of the problem is not easily described through a manageable declarative model

Decisions influence predictions and vice versa

We have access to problem data (historical, simulated, real) to create a training set

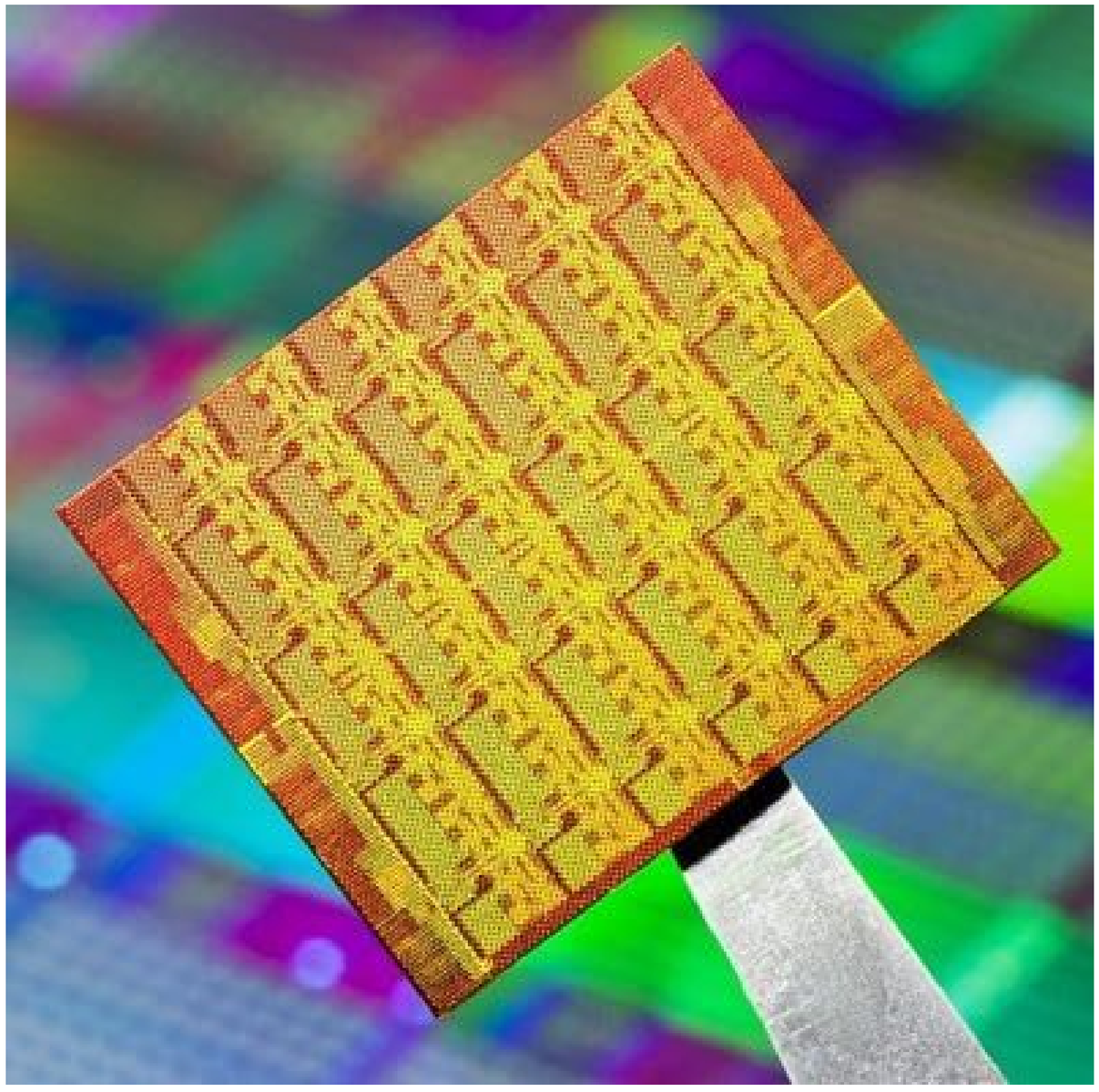
How can we learn the relation between decidables and observables?

A third example: thermal-aware workload dispatching

- Given a target heterogeneous HPC platform and a workload
- Decide task allocation and scheduing
- Every task has a thermal dynamics (and there is a thermal controller)
- Probably other constraints (not considered)

Assessing the effect of workload allocation on thermal dynamics:
We have access to the monitoring data of the real system

Thermal Management for Multicore Systems

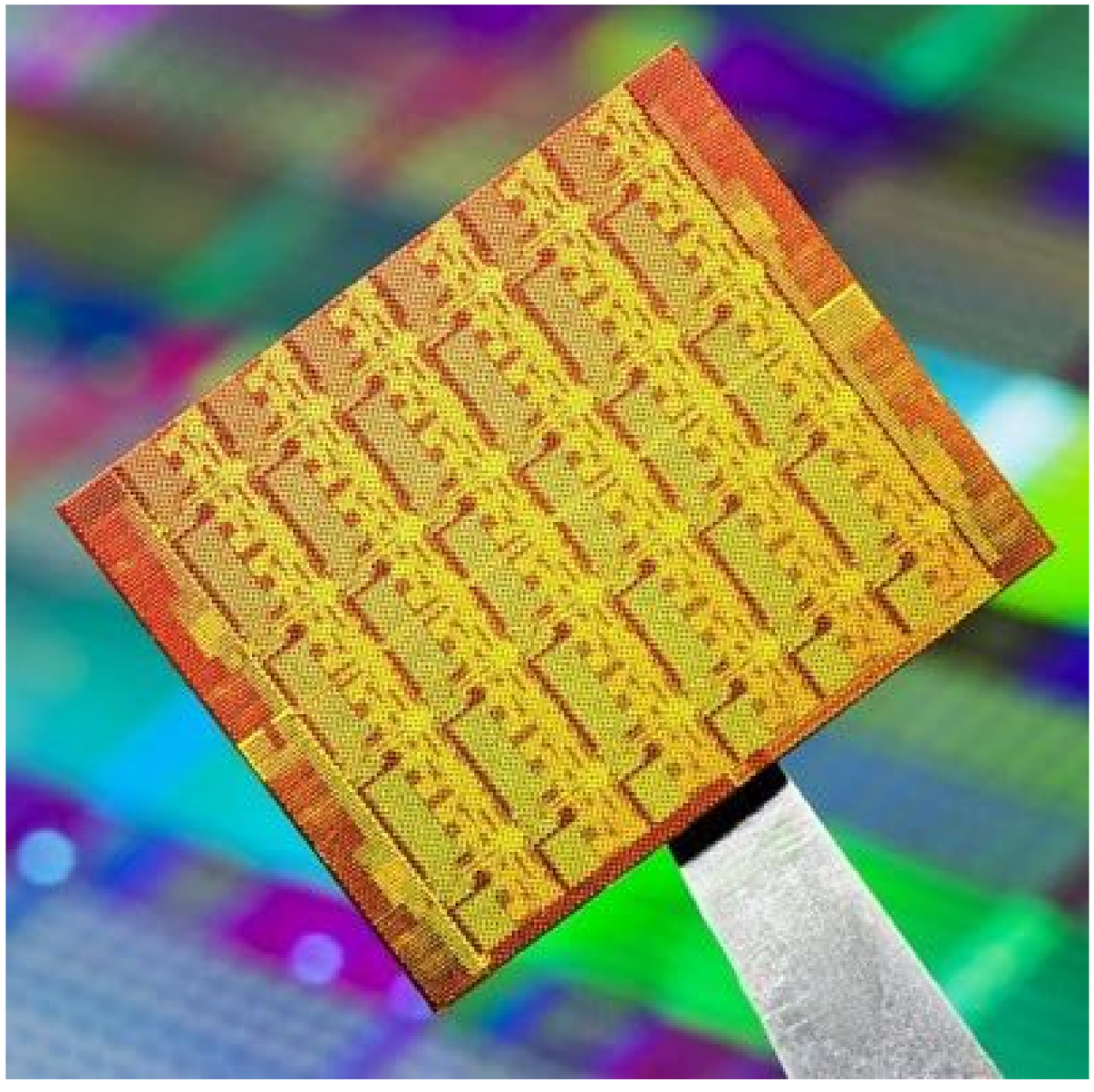


Intel SCC

Single-chip Cloud Computer

- 24 dual core CPUs (**48 cores**)
- **Network on Chip** (a full-fledged LAN in silicon)
- **4 Memory Controllers**

Thermal Management for Multicore Systems



Overheating is bad:

- Risk of chip damage
- Reduced performance
- Reduced lifetime

Traditionally:

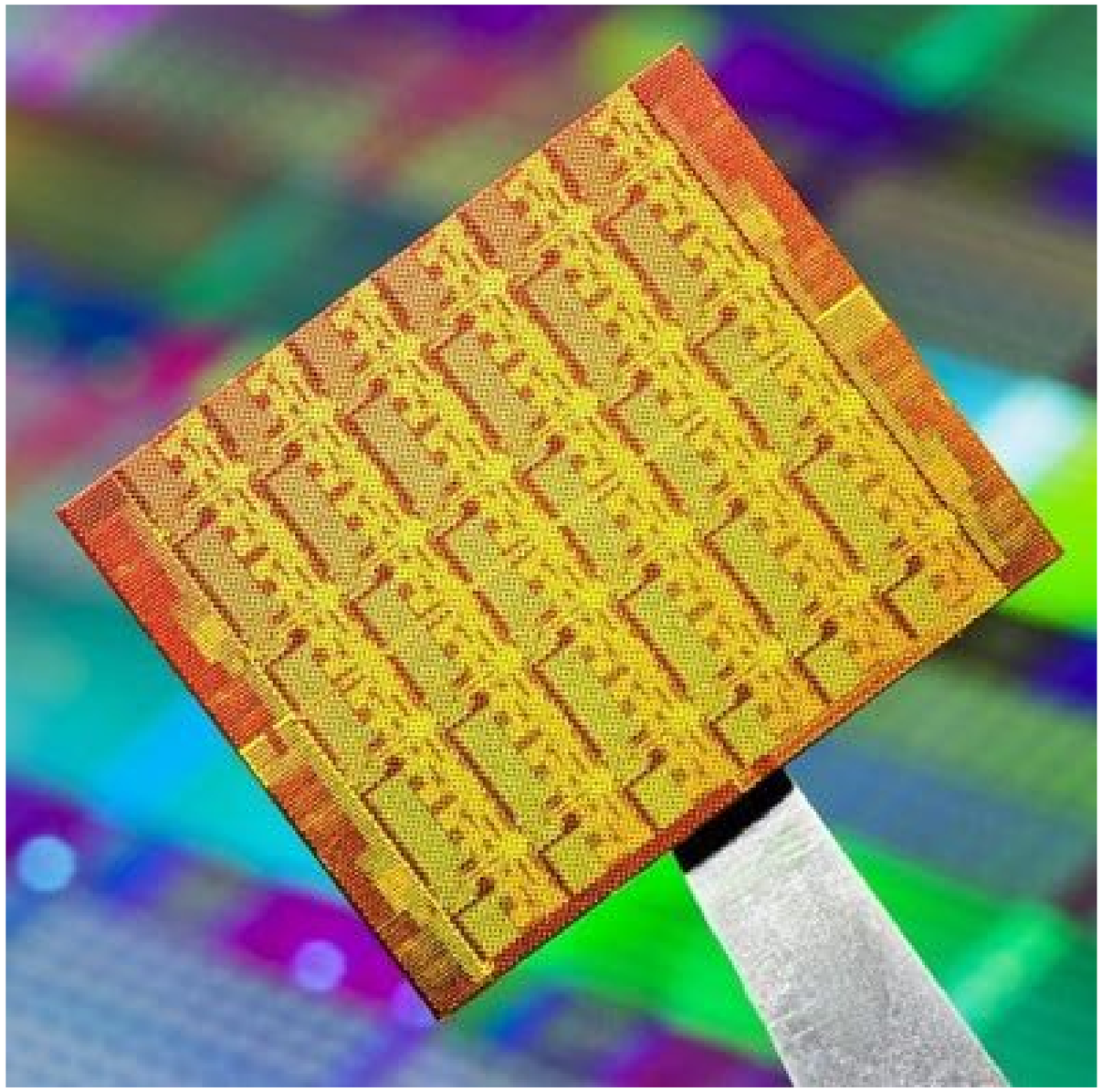
- Reactively reduce operating frequency or core shutdown
- Poor performance

Proactive techiniques:

Thermal aware workload dispatching and frequency control

- Thermal model needed

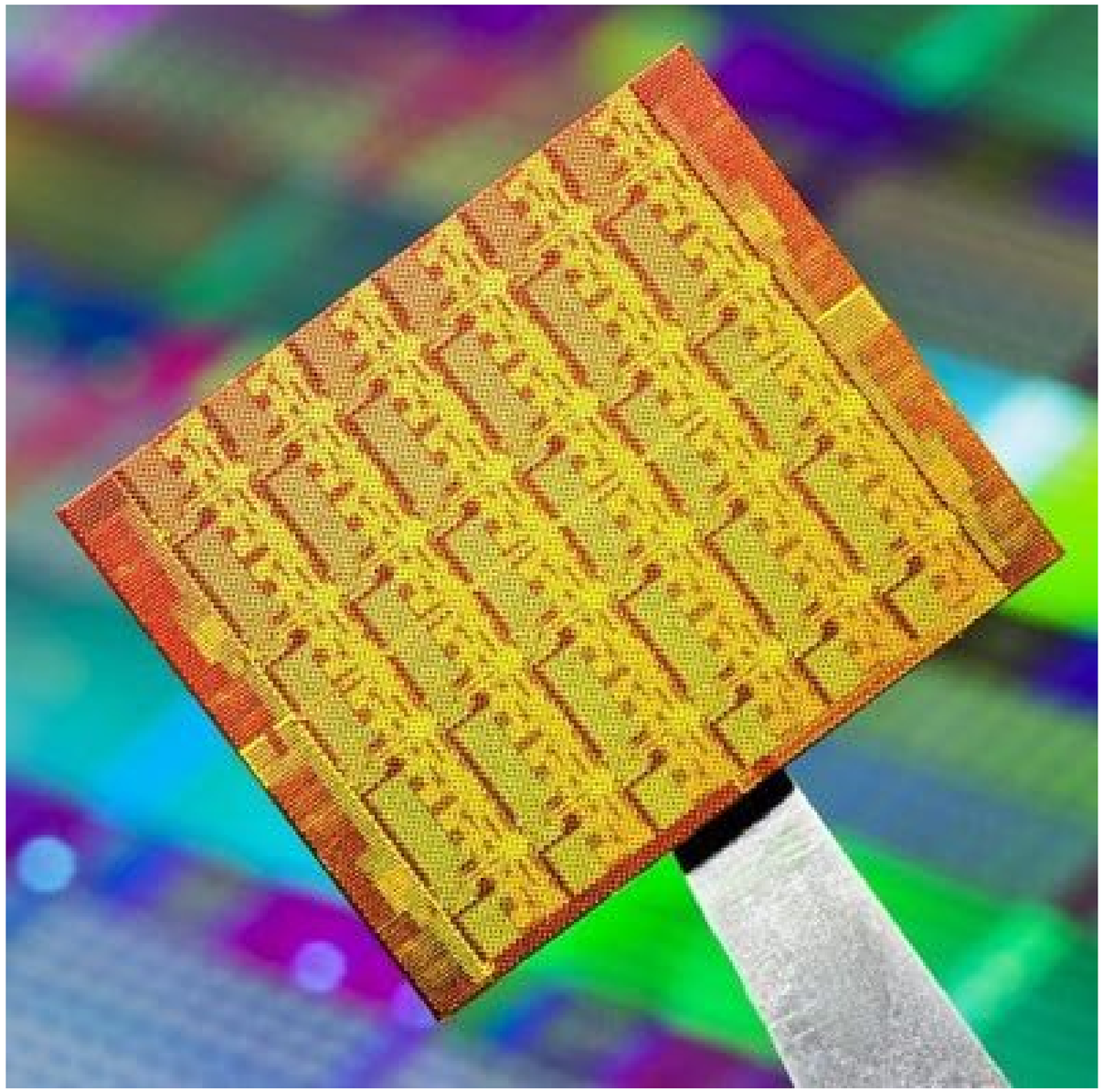
Thermal Management for Multicore Systems



The thermal behavior of a multicore system depends on:

- The room temperature
- The workload of each core
- The chip fabric
- The position of cores and heat sinks
- Heat transfer between cores
- On-chip thermal controllers

Thermal Management for Multicore Systems

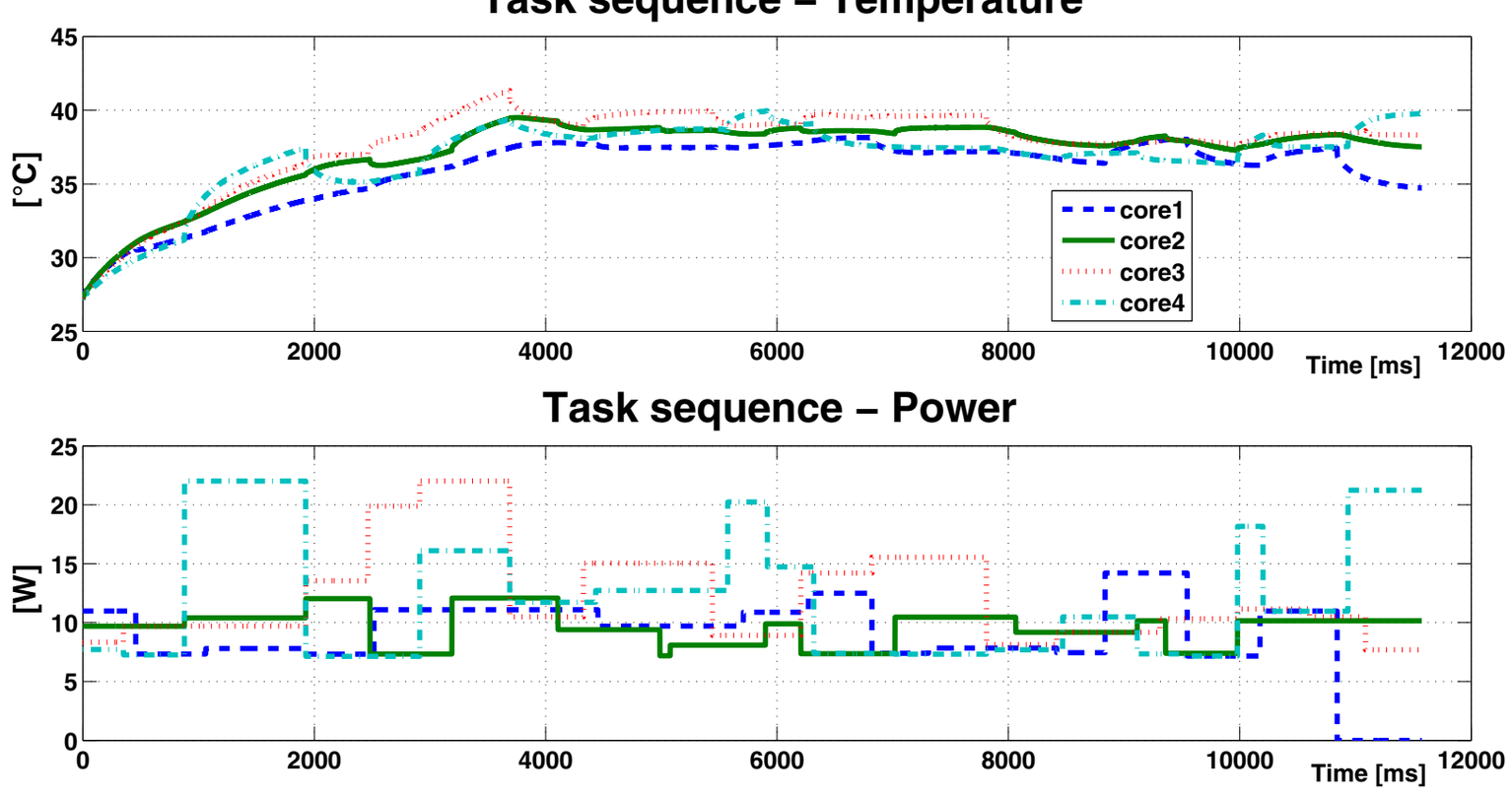


- **Effective computational models (thermal simulators),** but too slow for being embedded in an optimization algorithm (10s of seconds)

- **Declarative models hard to define:** currently strongly simplified models designed by experts, or tailored for a specific platform

Empirical Model Learning - EML

- Start from a set of observations extracted from runs on a real system
  - Sample the allocation space by running the system with different inputs of workload dispatching
  - Produce a training set

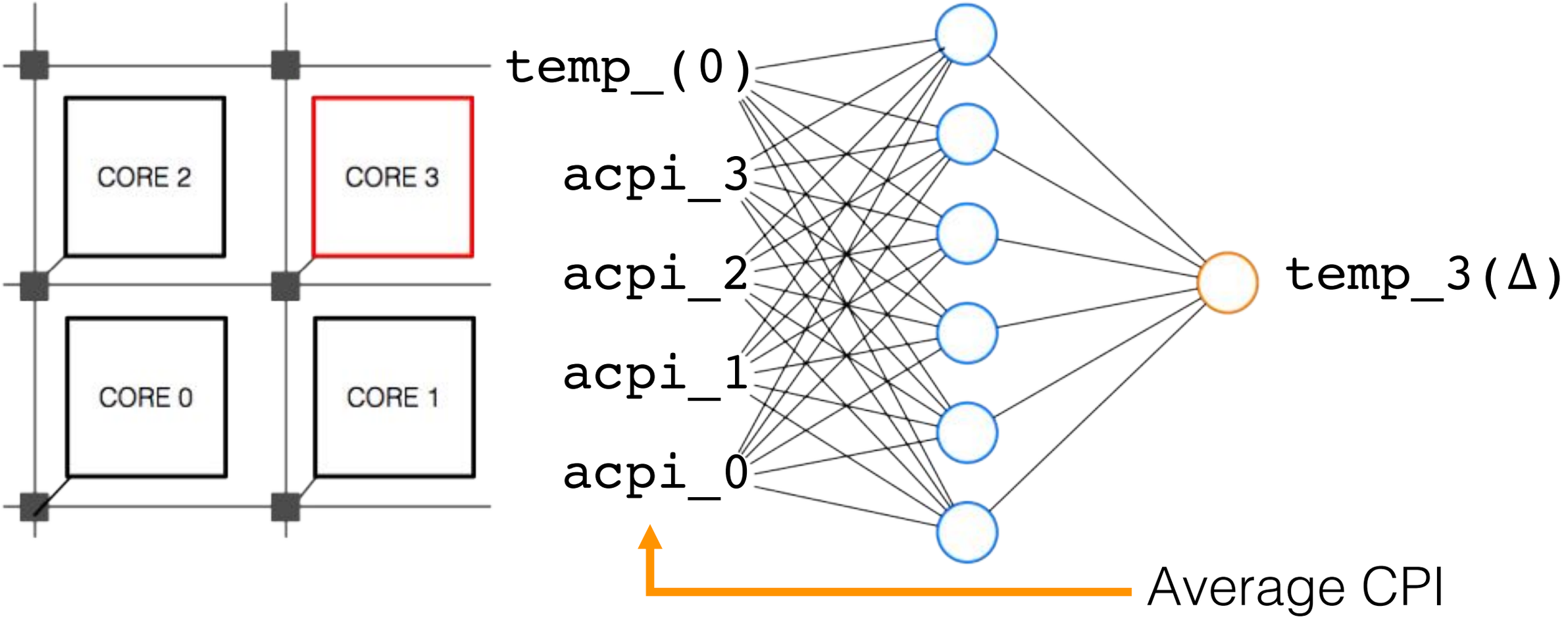


Empirical Model Learning - EML

Start from a set of monitoring data from the real system:

- Learn an approximate function via Machine Learning

$$f\text{: dispatching} \longrightarrow \text{temperature}$$



Empirical Model Learning - EML

Start from a set of observations

- Learn an approximate function via Machine Learning
- Embed this "empirical model" inside an optimization model

$$\min z = f(\vec{y})$$

subject to: $\vec{y} = g(\vec{x})$

all manner of complex constraints

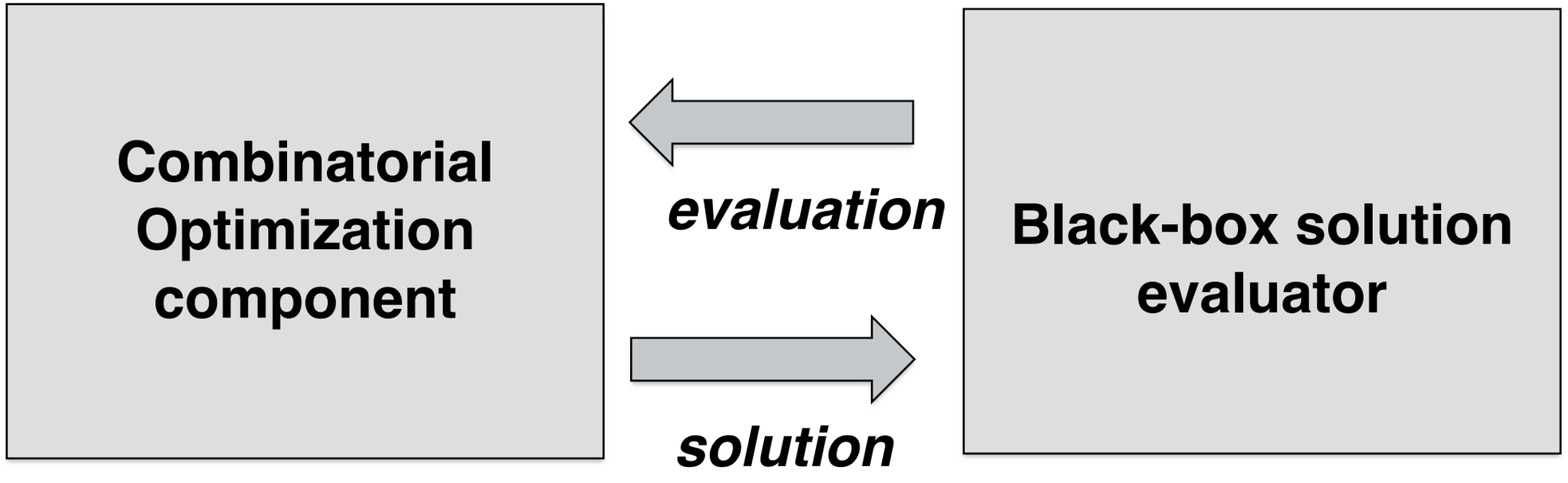
$$\vdots$$

In principle……

we can have a combinatorial problem solver

and a black box tool to test the solution

Can be very expensive

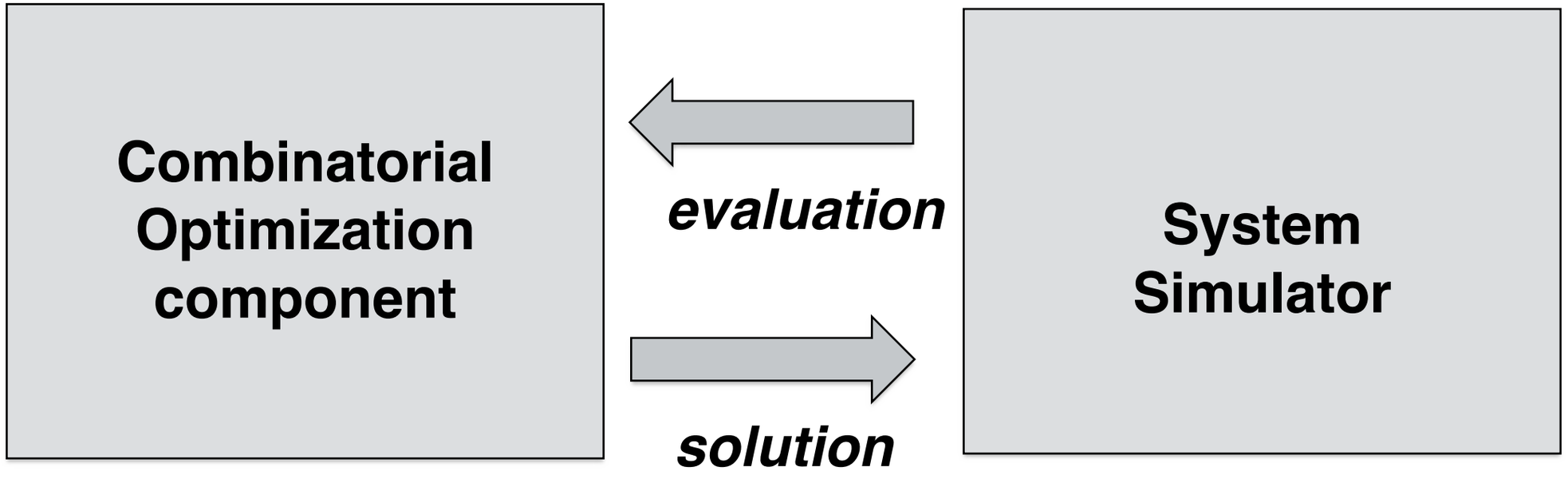


In principle……

we can have a combinatorial problem solver

and a black box tool to test the solution

Can be very expensive

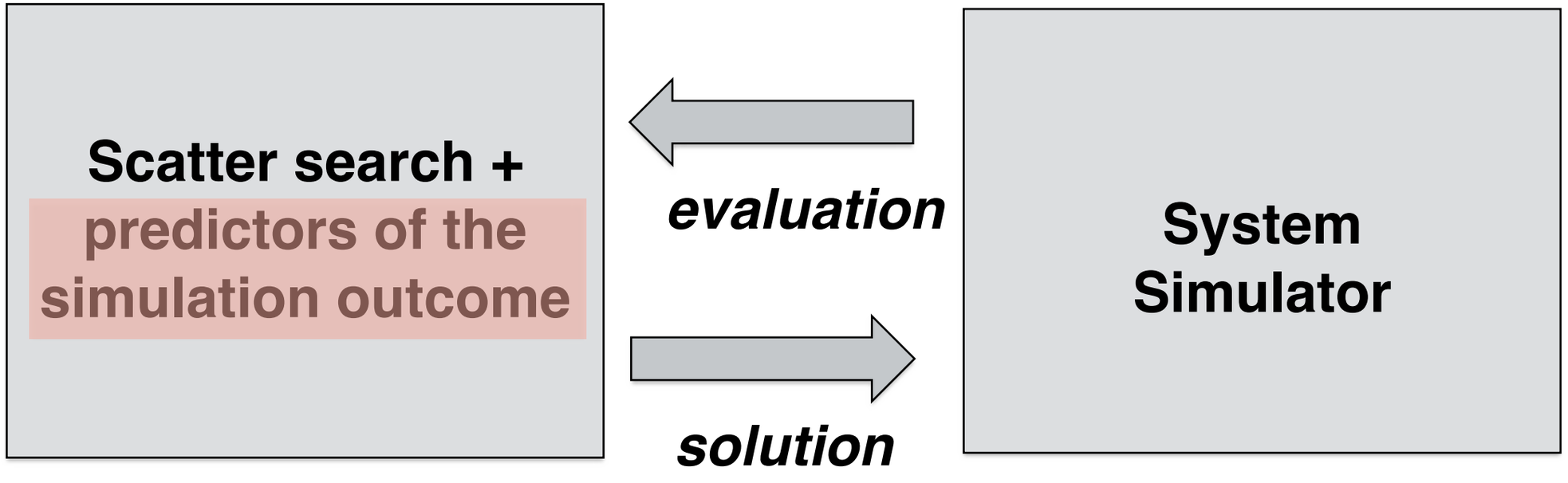


In principle……

we can have a combinatorial problem solver

and a black box tool to test the solution

Can be very expensive



Note: these predictors enable a selection of solutions to be sent to the simulator

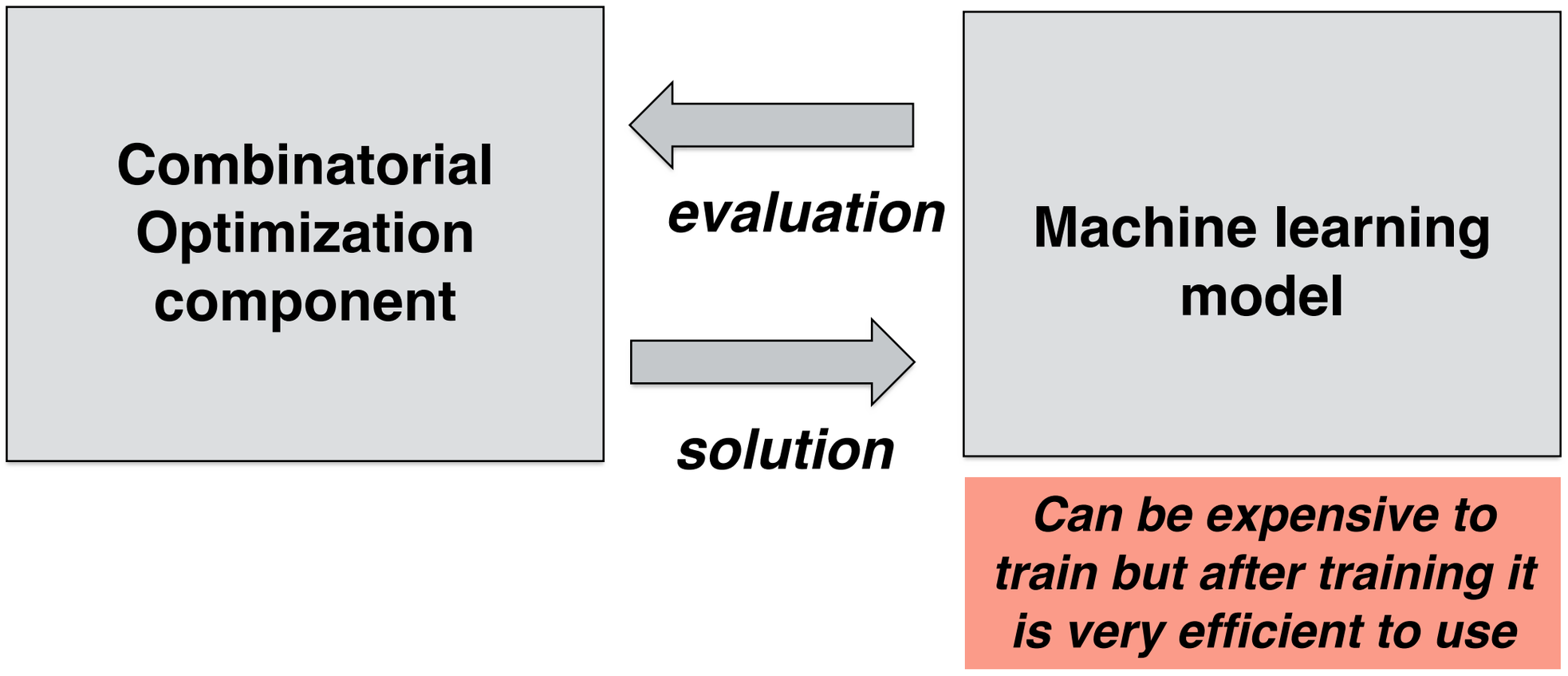
[OptQuest M. Laguna, 2011]

In principle……

we can have a combinatorial problem solver

and a black box tool to test the solution

What if the black box tool is a machine learning model?



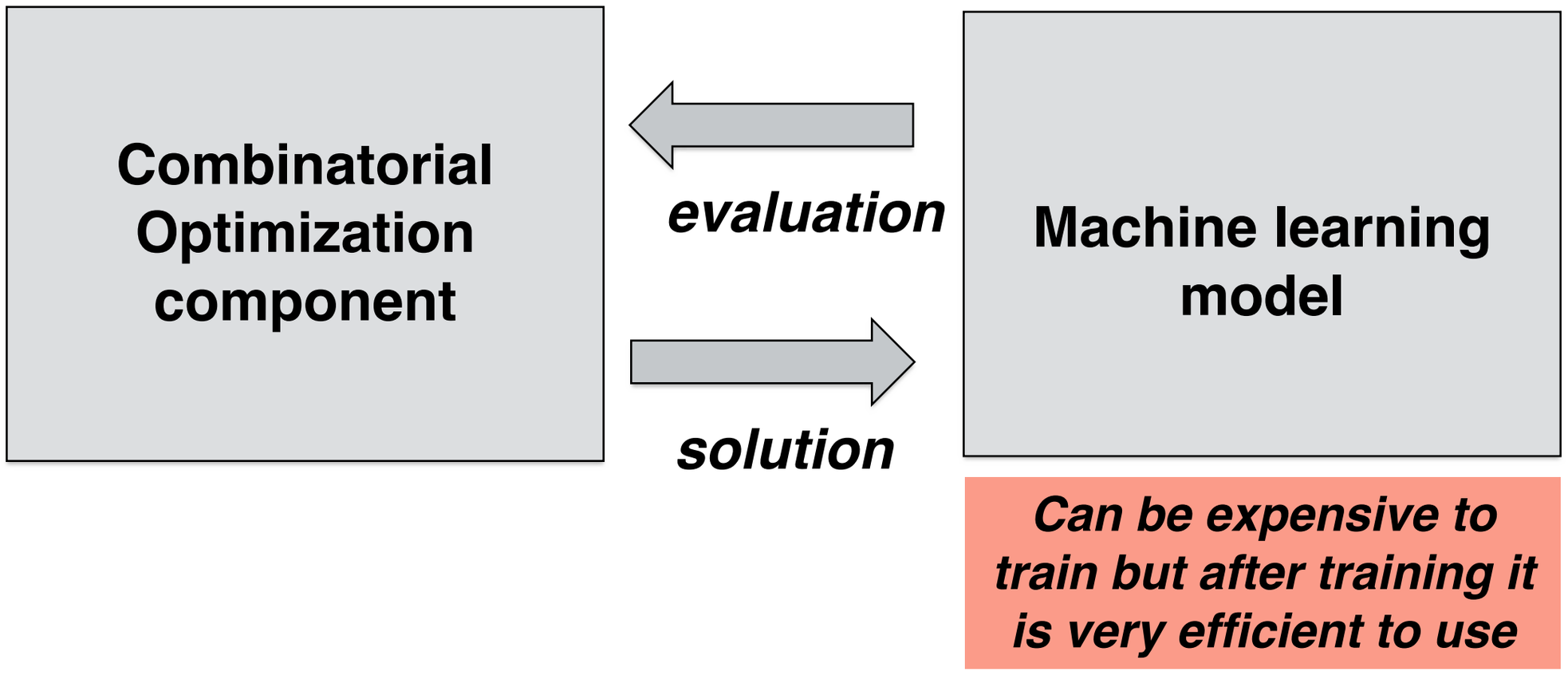
Surrogate models: [Henao, Maravelias, AIChE 2011], [Cozad, Sahinidis, Miller, 2014]

In principle……

we can have a combinatorial problem solver

and a black box tool to test the solution

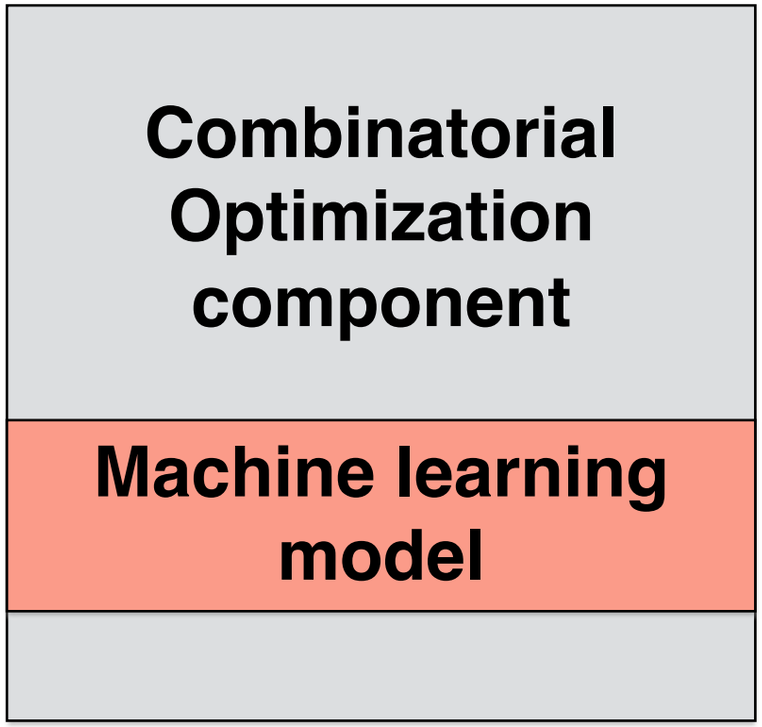
What if the black box tool is a machine learning model?



Still….. a generate and test mechanism

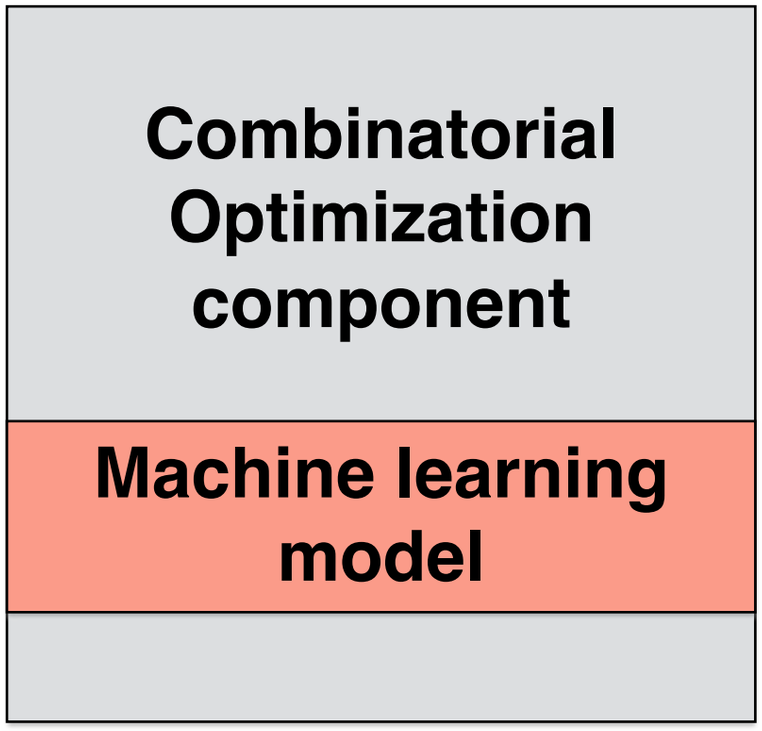
Empirical Model Learning – EML

In EML the learned model is embedded into the optimization component and used to actively reduce the search space during execution



Empirical Model Learning – EML

In EML the learned model is embedded into the optimization component an used to actively reduce the search space during execution



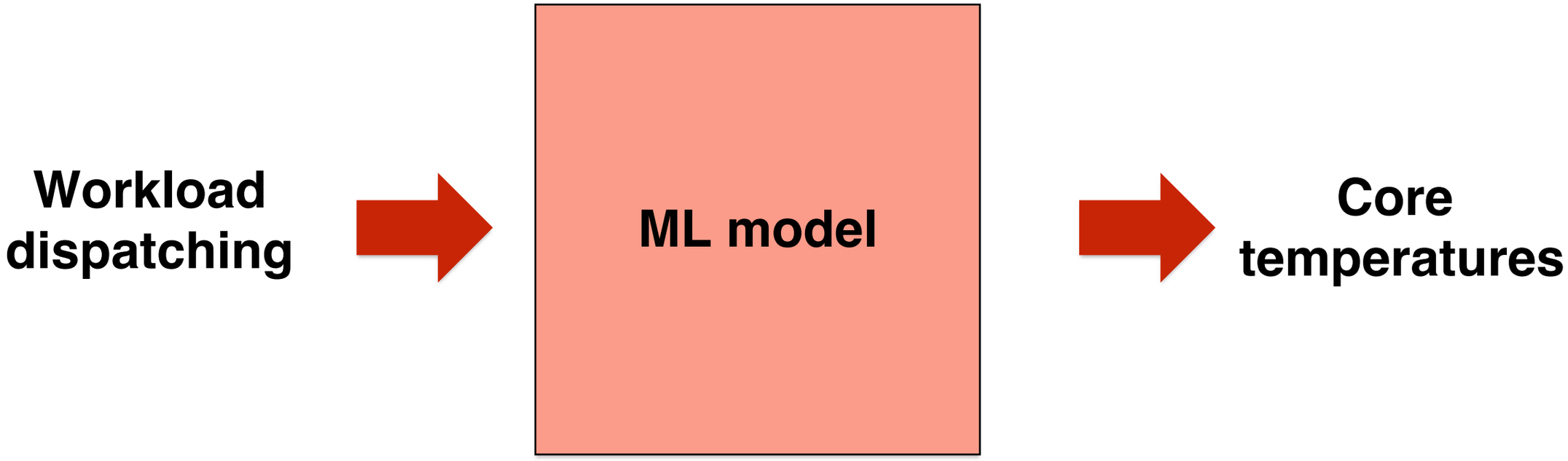
Not limited to objective functions, but also constraints and any relation between decidables and observables

Lombardi, Milano, Bartolini, Empirical Decision Model Learning, AIJ (244), 2017

Empirical Model Learning – EML

What is the difference between EML and the traditional use of ML models?

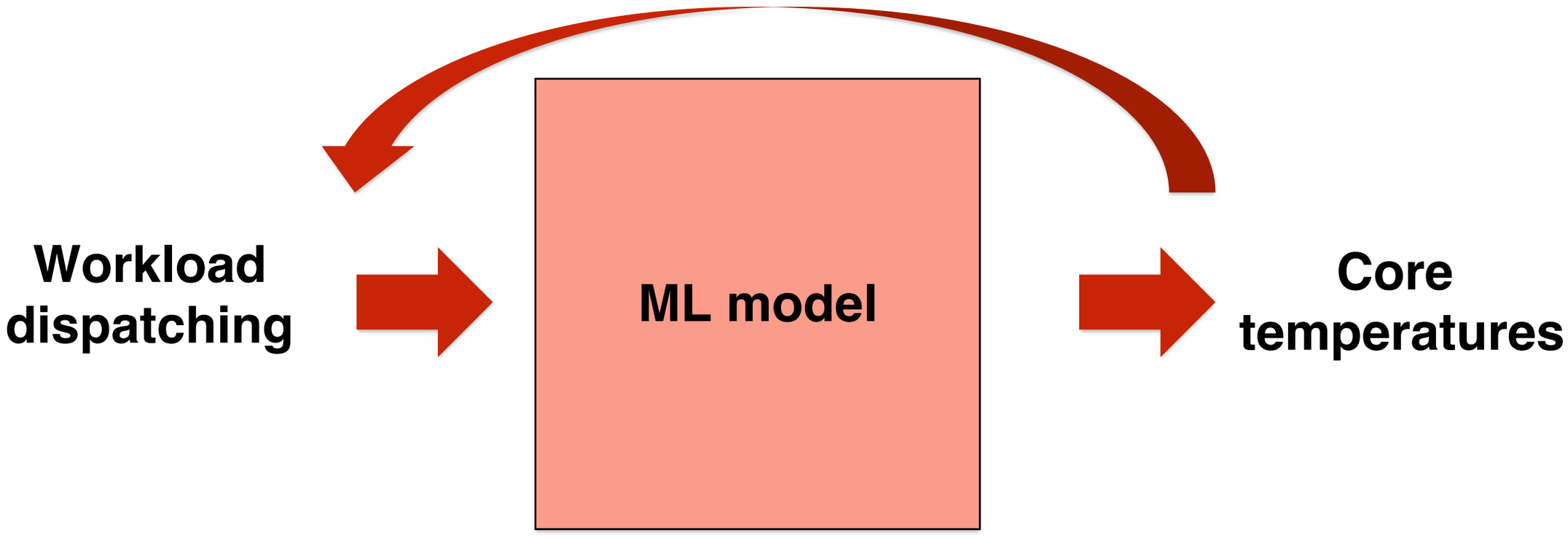
Traditional approaches work forward



Empirical Model Learning – EML

What is the difference between EML and the traditional use of ML models?

EML works forward and backward



The ML model becomes a constraint: given temperature limits we remove combinations of workload decisions that lead to inconsistent temperatures

Thermal-aware workload dispatching

COMBINATORIAL COMPONENT

Given n tasks m cores

$\sum_j \sum_i x_{ij} = 1$

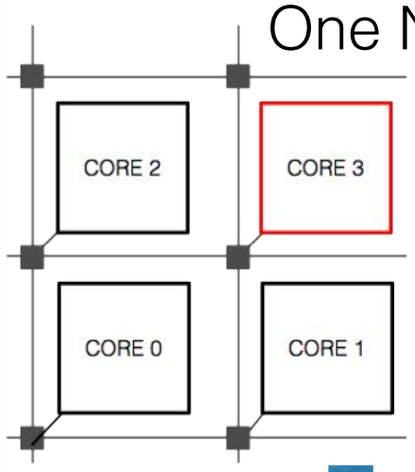
$\sum_i x_{ij} = n/m$ Load balancing

Other constraints (e.g. scheduling)

x_{ij} is 1 if task i is allocated on core j

MACHINE LEARNING MODEL

One NN per core



min (max $temp_j$)

$\sum_j \sum_i x_{ij} \leq 1$

$\sum_i x_{ij} = n/m$

EM(x, temp)

Other constraints (e.g. scheduling)

x_{ij} is 1 if task i is allocated on core j

How to cast a ML model into a combinatorial optimization one?

Empirical Model Learning – EML

The questions are:

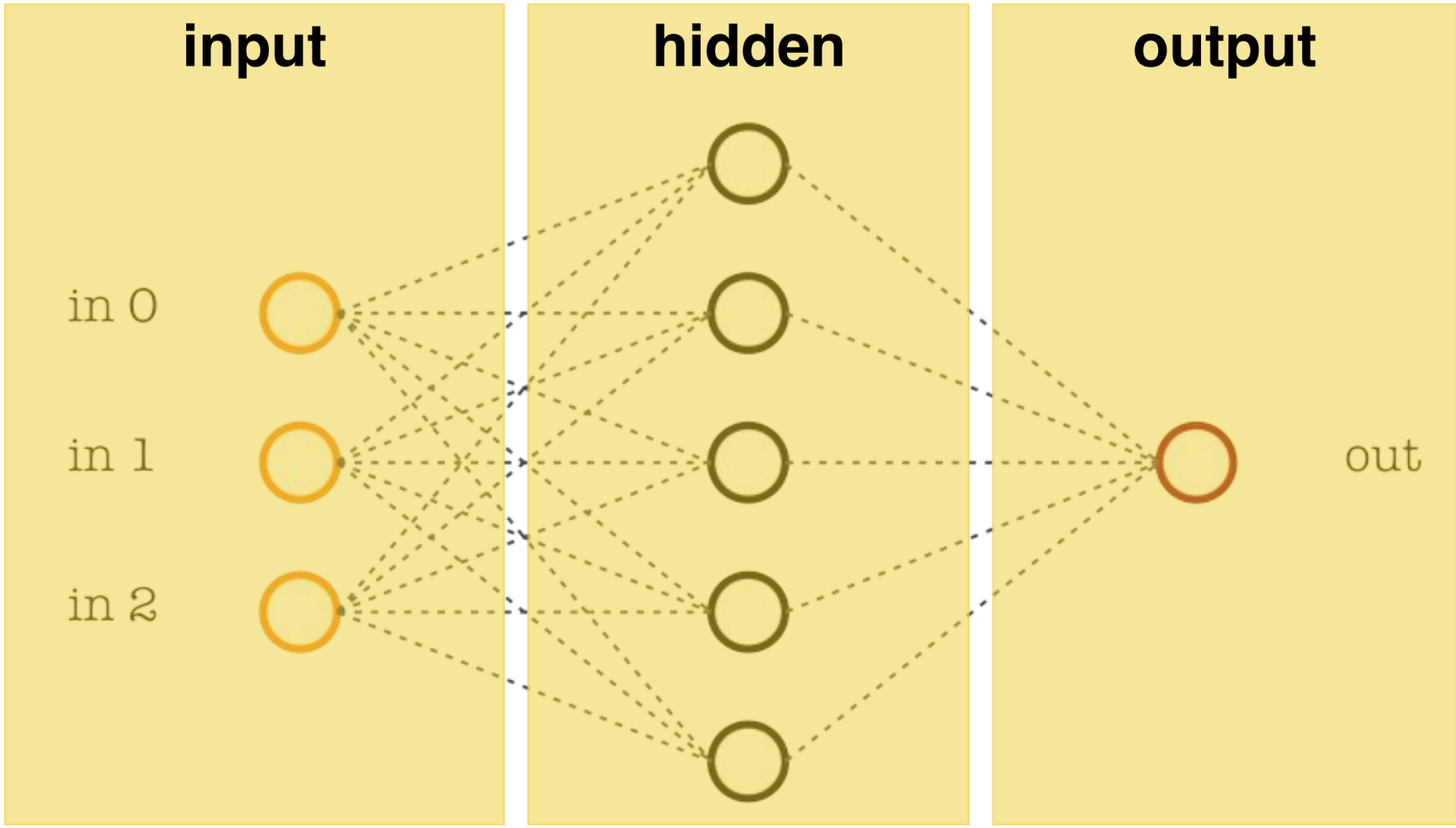
1) how do we embedded Machine learning models into a combinatorial model?
2) How do we provide operational semantics to these models so that they can actively prune the search space?

We will explore:

- Neural Networks
- Decision Trees

In different optimization techniques: CP in the talk, but also MINLP/ILP, SMT

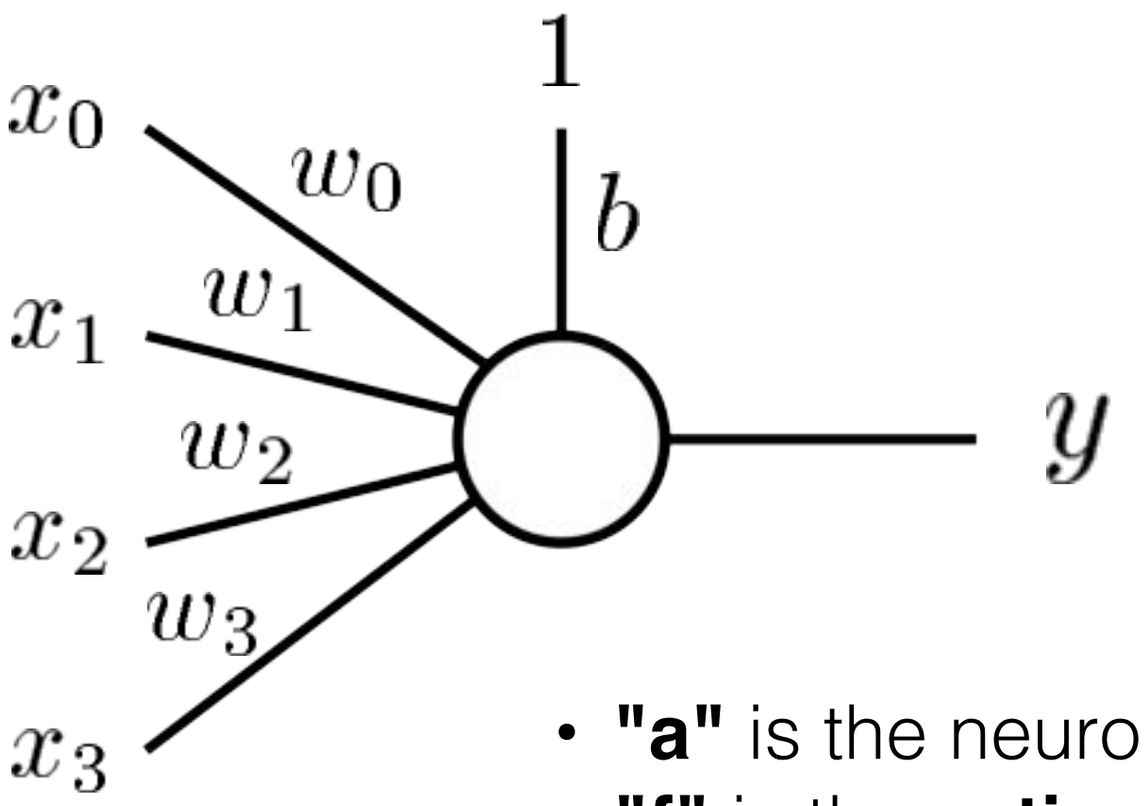
Artificial Neural Networks



- Computational model, consisting of parallel computation units (neurons) connected in a network structure
- Usually: no cycles (feed-forward) and multiple layers

Artificial Neural Networks

Artificial Neuron = scalar function with vector input



$$a = \sum_i w_i \cdot x_i + b$$

$$y = f(a)$$

- **"a"** is the neuron **activity**
- **"f"** is the **activation function**
- **weights** are obtained in the **learning phase**

The activation function is a **monotonic non-decreasing function** of the neuron activity

Bartolini, Lombardi, Milano, Benini, Neuron Constraints to Model Complex Real World Problems, CP2011.

Neural Networks in a Combinatorial Model

A Neural Network is a declarative (nonlinear) model

- Can be embedded in a MINLP model directly

1. We should insert variables for NN inputs and outputs in the model
2. Insert the NN non linear equations in the model

Neural Networks in a Combinatorial Model

A Neural Network is a declarative (nonlinear) model

- Can be embedded in a CP model via variables and constraints
- Here: **artificial neuron = a global "neuron constraint"**

```
act.function([X], Y, [W], b)
```

Input variables

Output variable

Weights

Bias

Structure of the Neuron Constraint

- An internal variable **A** for the neuron activity
- Bound consistency on the equality: $\mathtt{A} = \sum_i w_i \cdot \mathtt{X}_i + b$

Filtering for the activation function:

1. $\max(\mathtt{A})$ updated $\Rightarrow \max(\mathtt{Y}) = f(max(\mathtt{A}))$
2. $\max(\mathtt{Y})$ updated $\Rightarrow \max(\mathtt{A}) = f^{-1}(\max(\mathtt{Y}))$

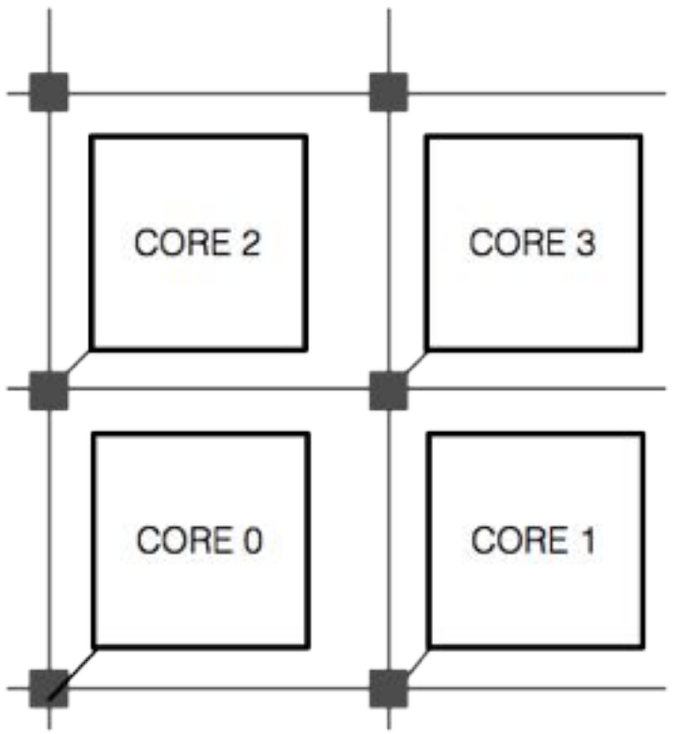
- Similar rules for minima
- The value $f^{-1}(max(\mathtt{Y}))$ is replaced by $\max\{a' \mid f(a') = \max(\mathtt{Y})\}$ to avoid precision errors
- Filtering from **Y** to **A** and from **A** to **Y**

We can combine many neuron constraints in a artificial neural network global constraint: **`ann(output, inputs)`**

How it works at search time

Tasks

Allocation Variables



Feature computation

NN In Vars

temp
acpi_3
acpi_2
acpi_1
acpi_0

NN Out Vars

temp_2 (Δ)

NN In Vars

temp
acpi_3
acpi_2
acpi_1
acpi_0

NN Out Vars

temp_3 (Δ)

NN In Vars

temp
acpi_3
acpi_2
acpi_1
acpi_0

NN Out Vars

temp_0 (Δ)

NN In Vars

temp
acpi_3
acpi_2
acpi_1
acpi_0

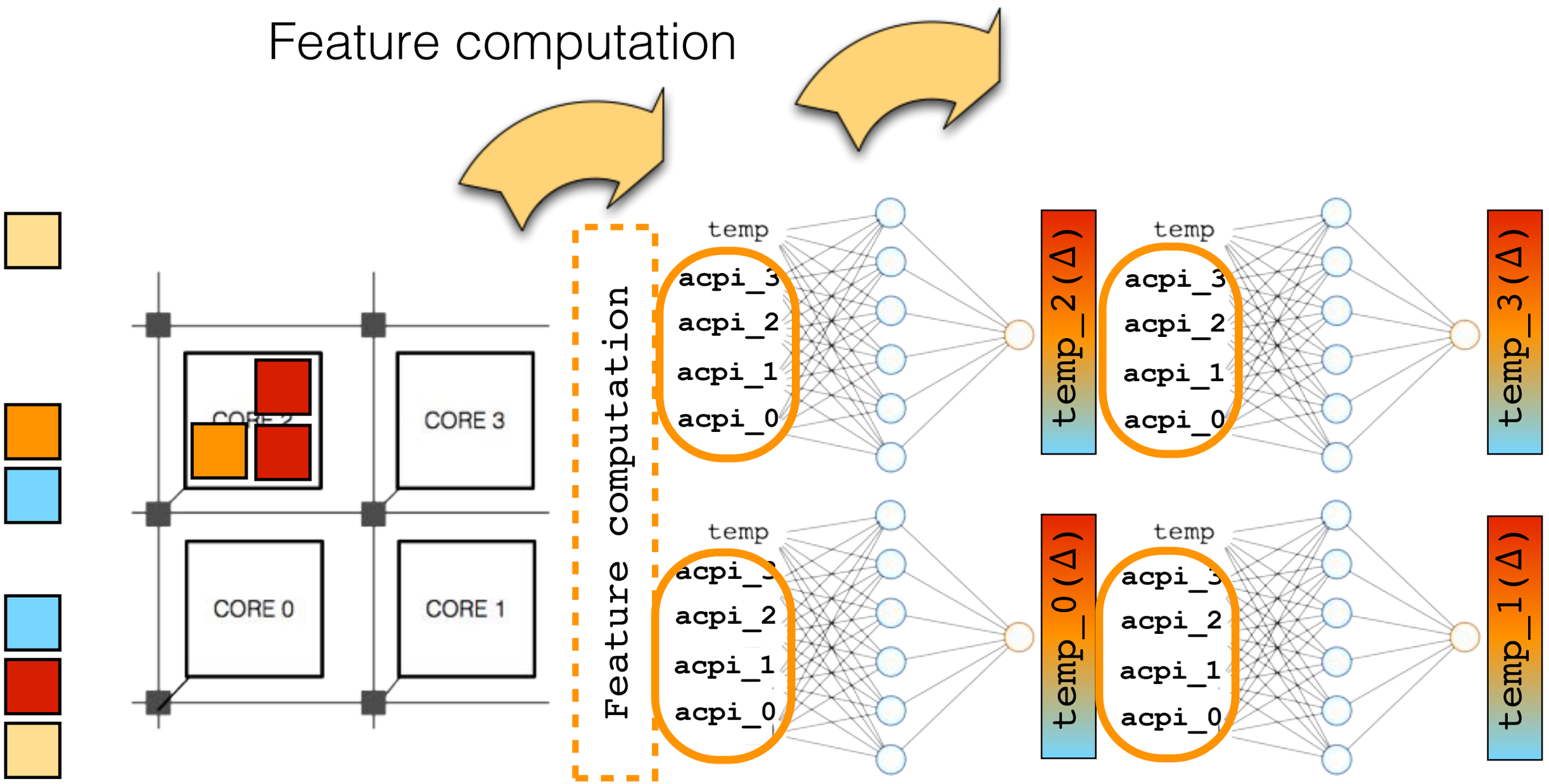
NN Out Vars

temp_1 (Δ)

How it works at search time

Neuron Constraints

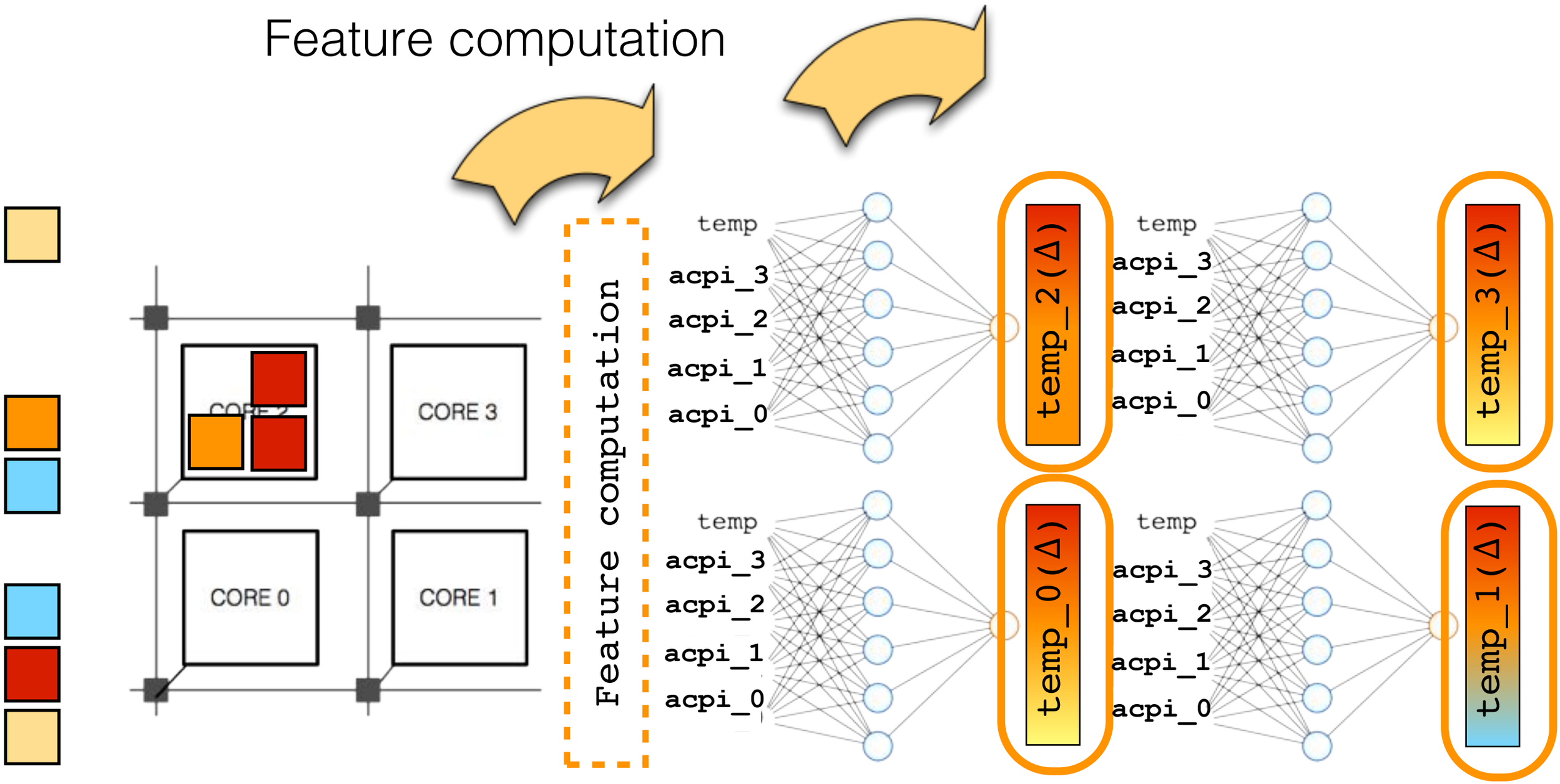
Feature computation



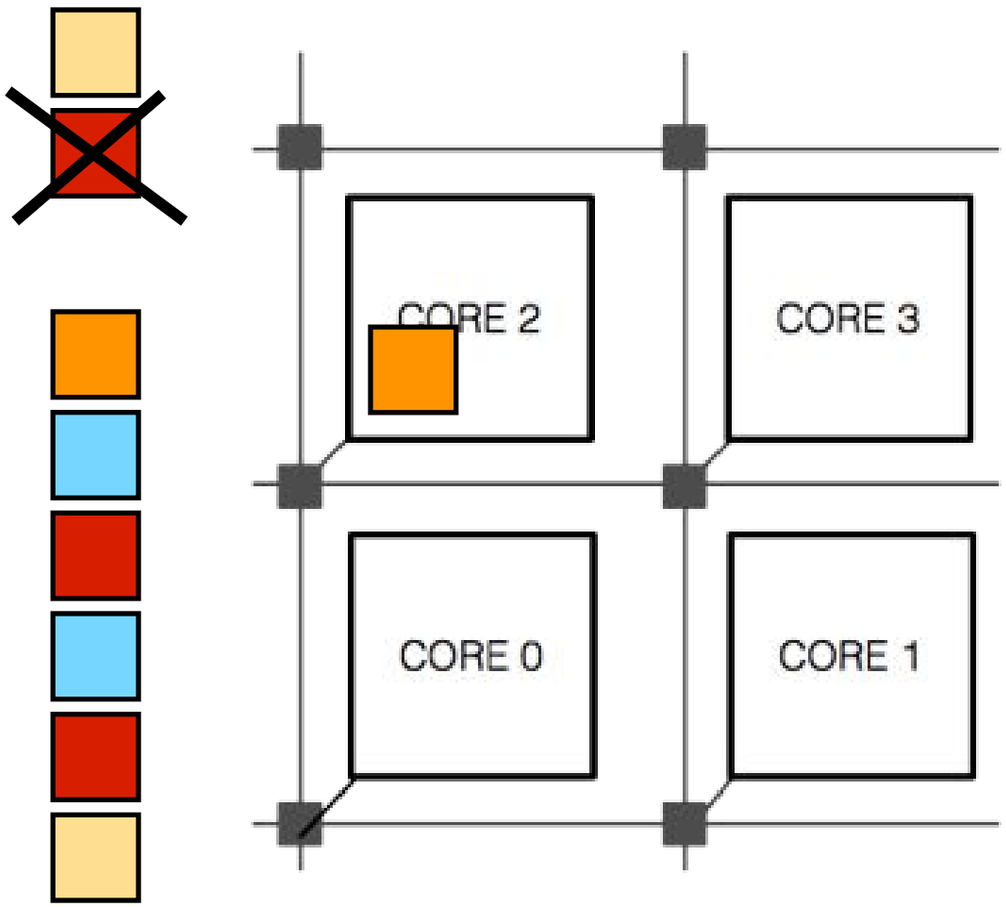
Forward Propagation

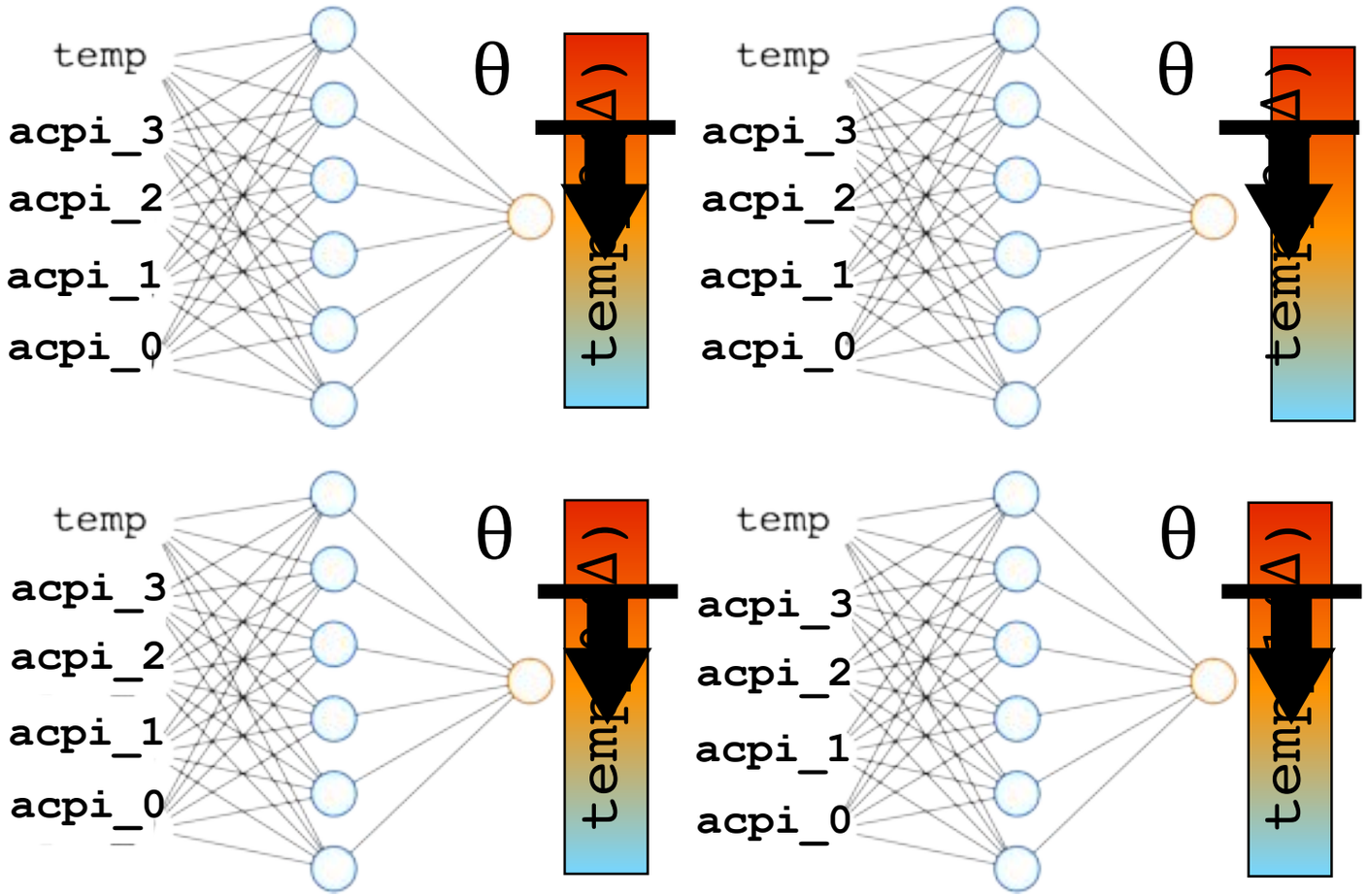
Neuron Constraints

Feature computation



Backward propagation





EML: CP model

Problem data:

- $\mathtt{n}$ tasks (with CPI values)
- Each task is assumed to run indefinitely
- $\mathtt{m}$ = 48 cores
- Room temperature

Decision variables:

$$\mathtt{CORE_i} \in \{0..m-1\} \quad \forall t_i$$

Balancing constraints:

Roughly the same number of tasks per core

$$gcc([\mathtt{CORE_i}], \{0..m-1\}, \lfloor {}^{n}/_{m} \rfloor, \lceil {}^{n}/_{m} \rceil)$$

EML: CP Model

Equivalent allocation variables:

$$\mathtt{X_{i,j}} = 1 \Leftrightarrow \mathtt{CORE_i} = j$$

Average CPI per core (FEATURE COMPUTATION):

$$average(\mathtt{ACPI_j}, [CPI_i], [\mathtt{X_{i,j}}|_{\mathtt{j=k}}]) \qquad \forall \text{ core } k$$

Neural Network: collection of act_function constraints

$$\mathtt{ann}_j(ctemp_j, ACPI_j, ACPI_{ngbr} \ldots, rtemp)$$

Objective:

$$\min \max_{j \in cores} ctemp_j$$

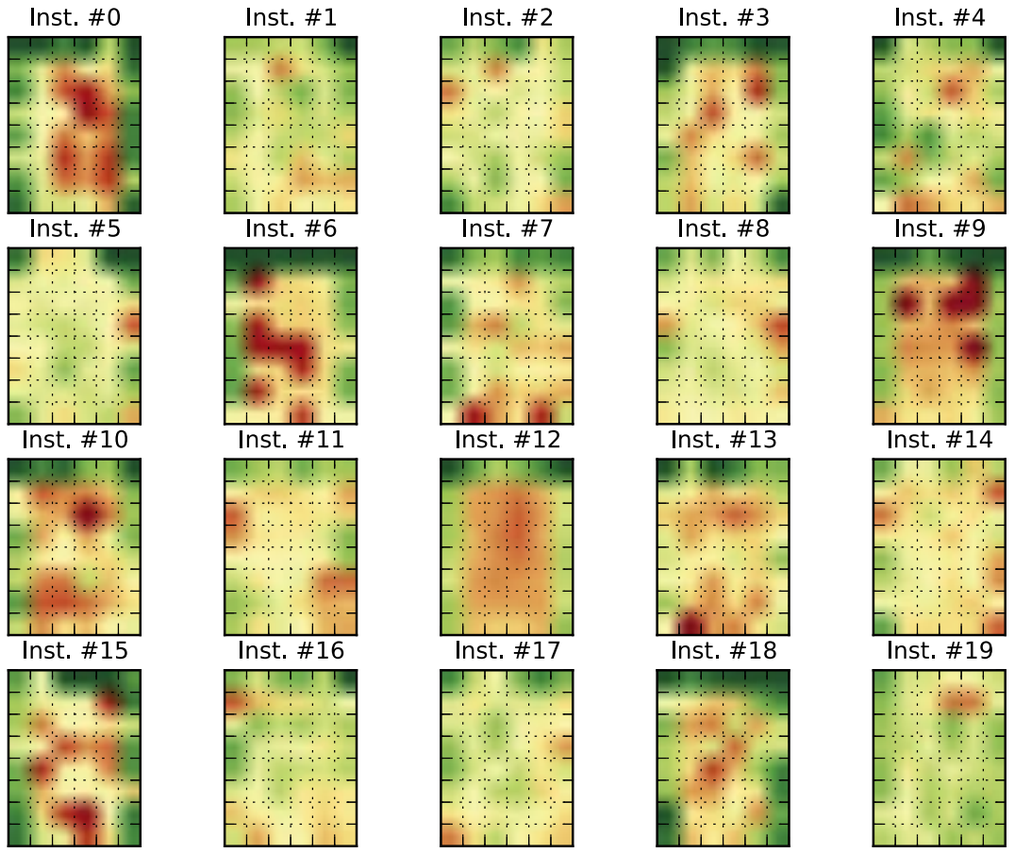
Bartolini, Lombardi, Milano, Benini, Optimization and Controlled Systems: A Case Study on Thermal Aware Workload Dispatching, AAAI 2012

Results

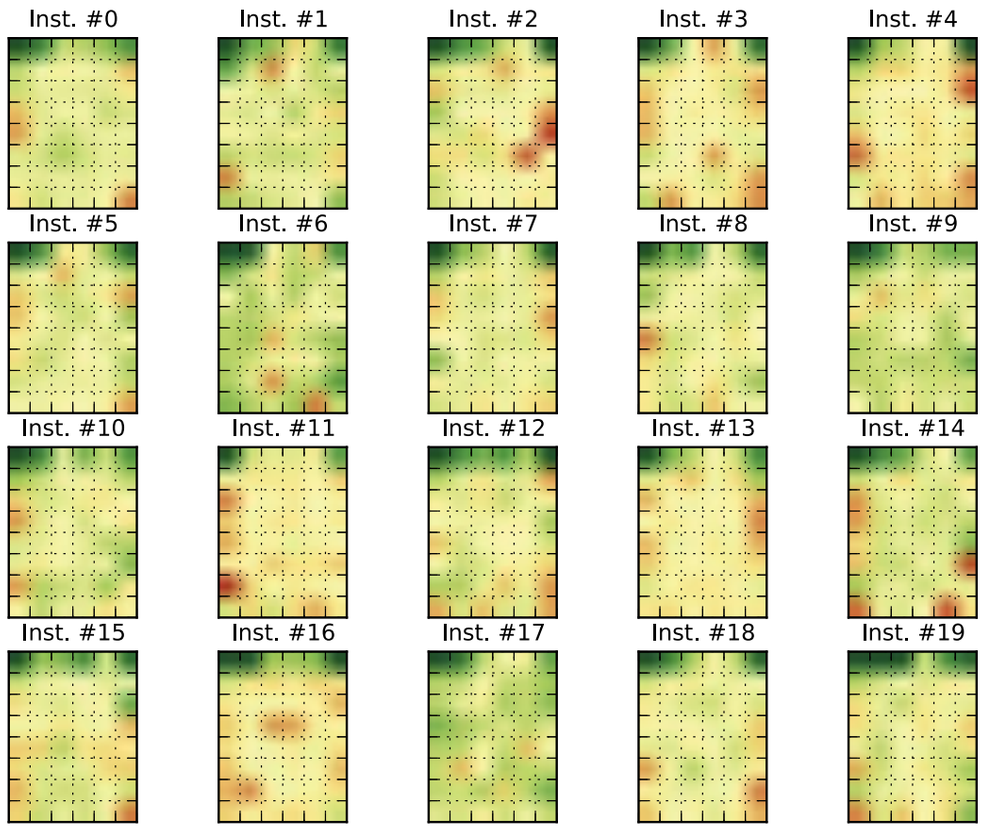
Solution Approach:

- Model solved via Large Neighborhood Search (LNS)
- First solution via a CPI balancing heuristic

Heat generated on the cores by MILP using an expert design model



Heat generated on the cores by the EML approach



How to cast a decision tree in a combinatorial model

Three encodings:

- Meta-constraints
- Table constraint
- Other encodings

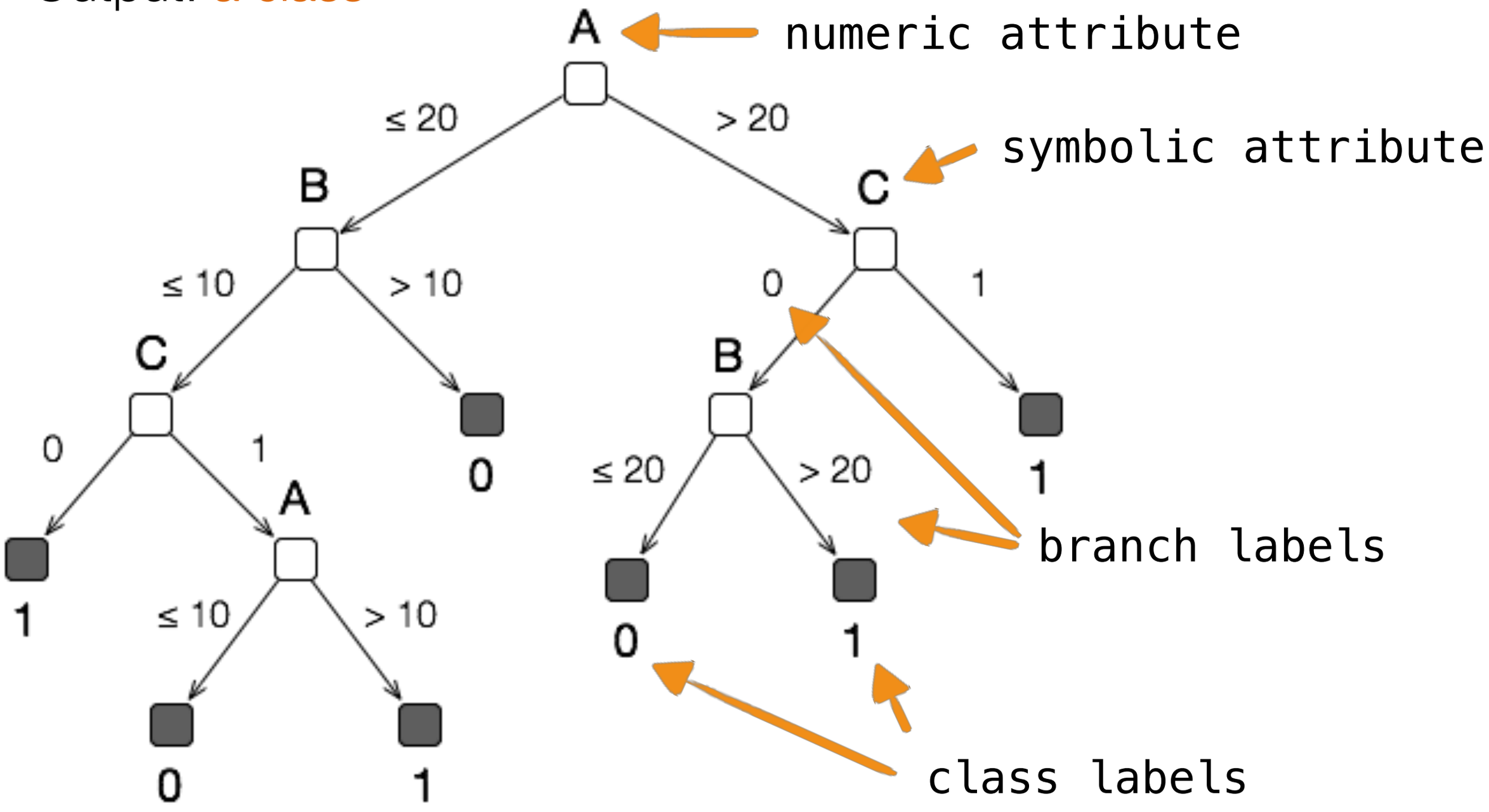
Extension: Random Forests

Bonfietti, Lombardi, Milano: Embedding Decision Trees and Random Forests in CP, CPAIOR 2011

Decision Tree

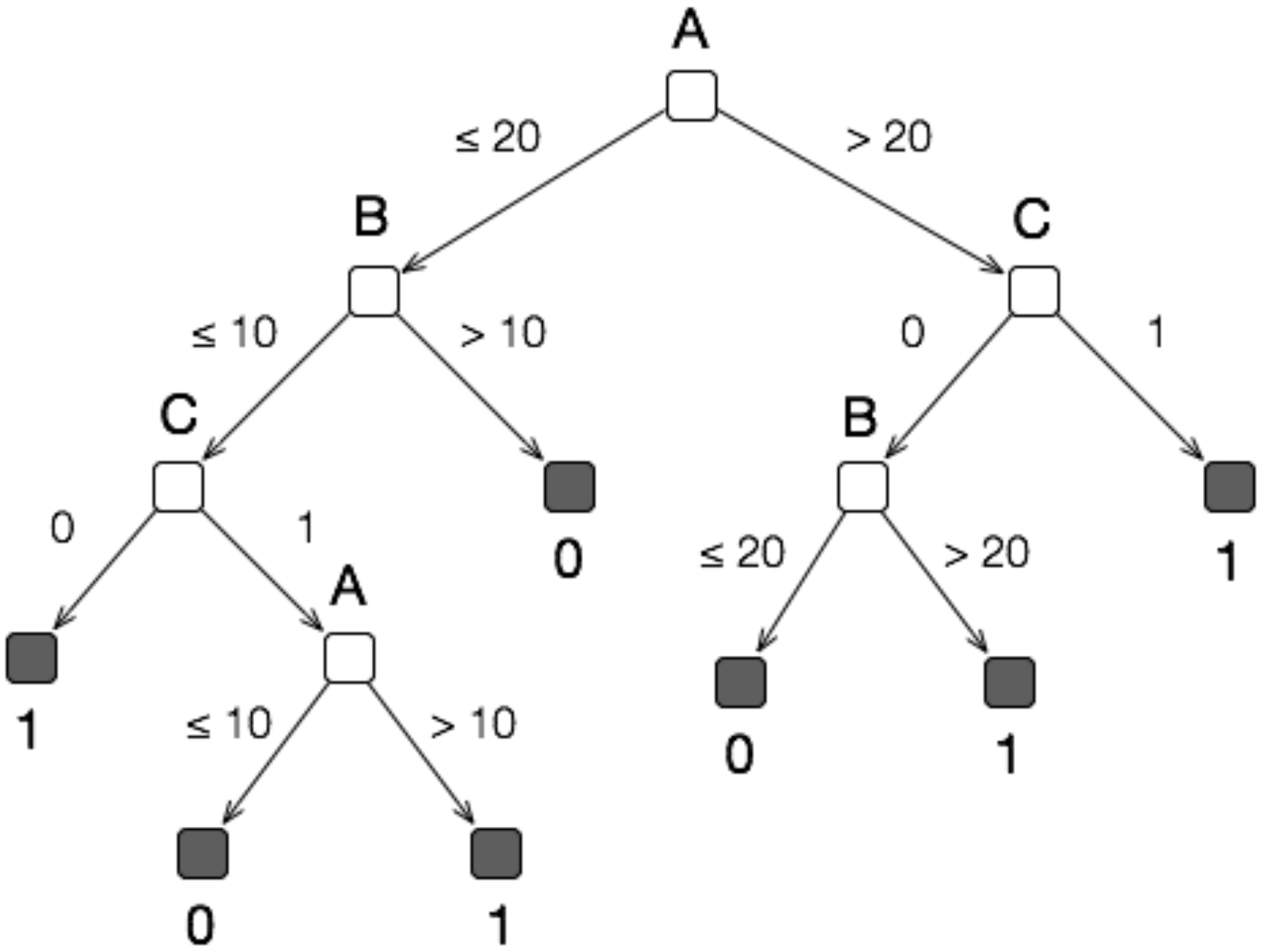
Input: tuple of attribute values

Output: a class

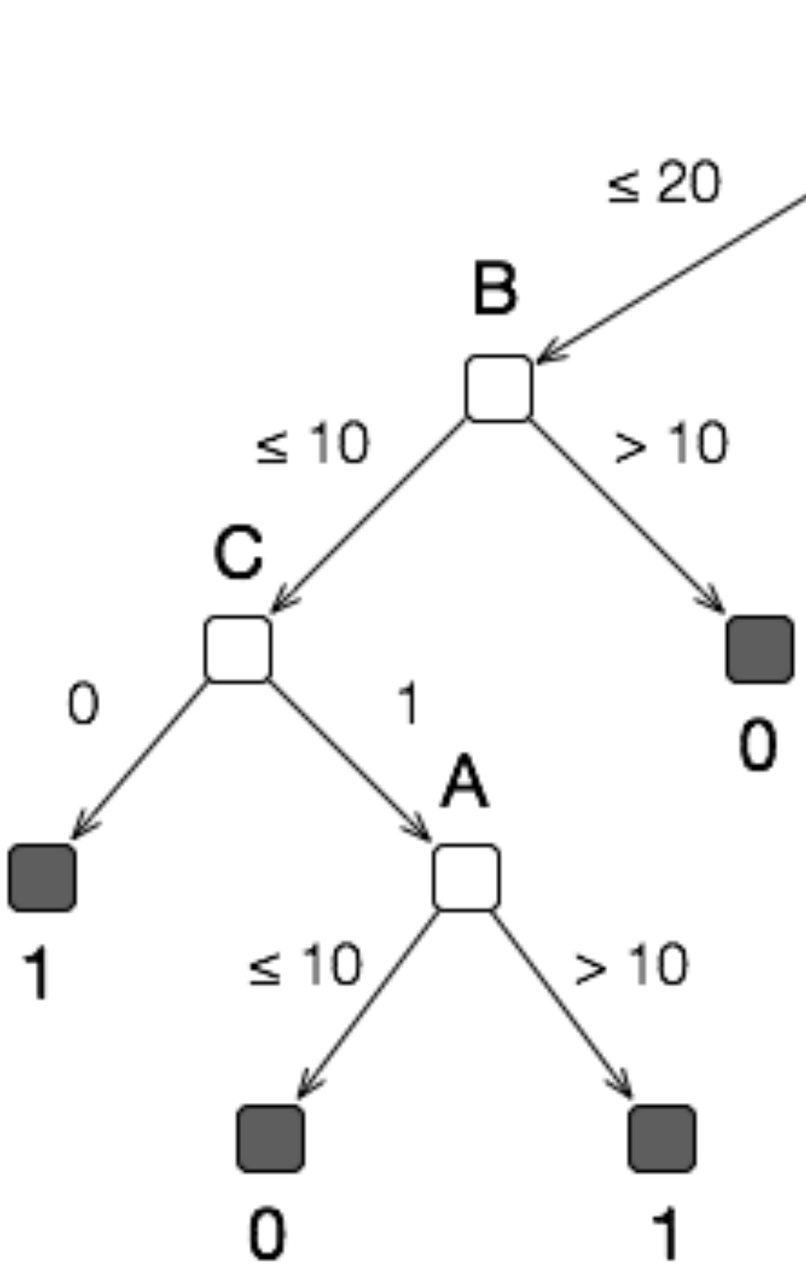


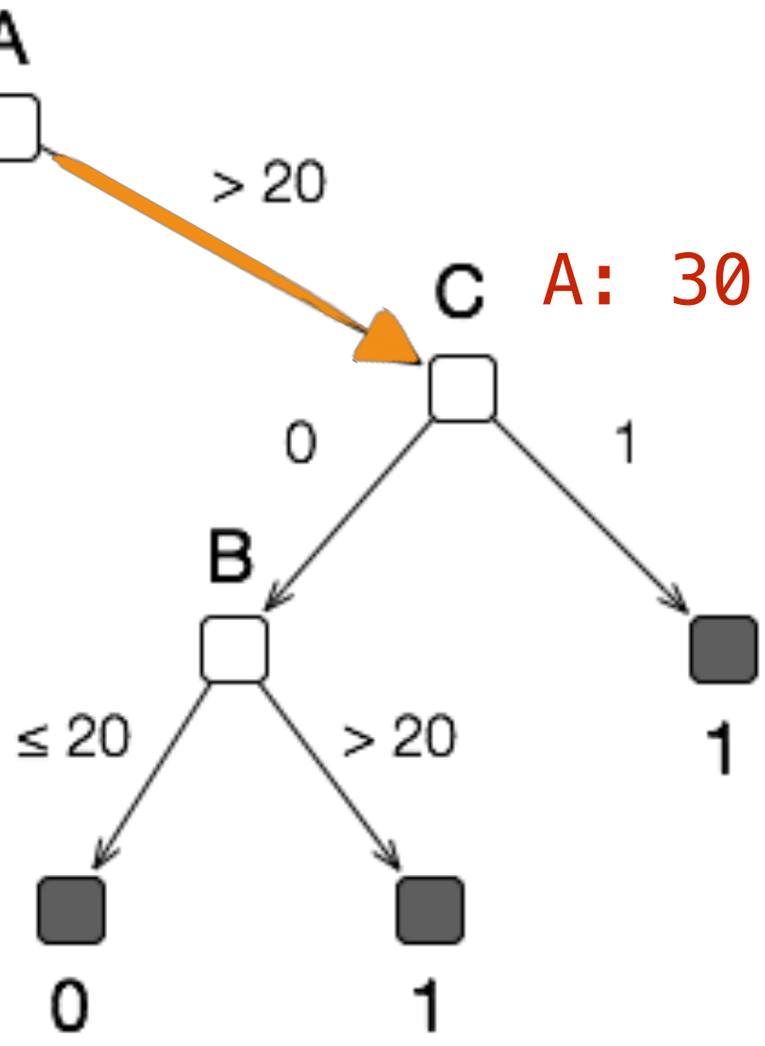
Classification example

A: 30, B: 20, C: 1

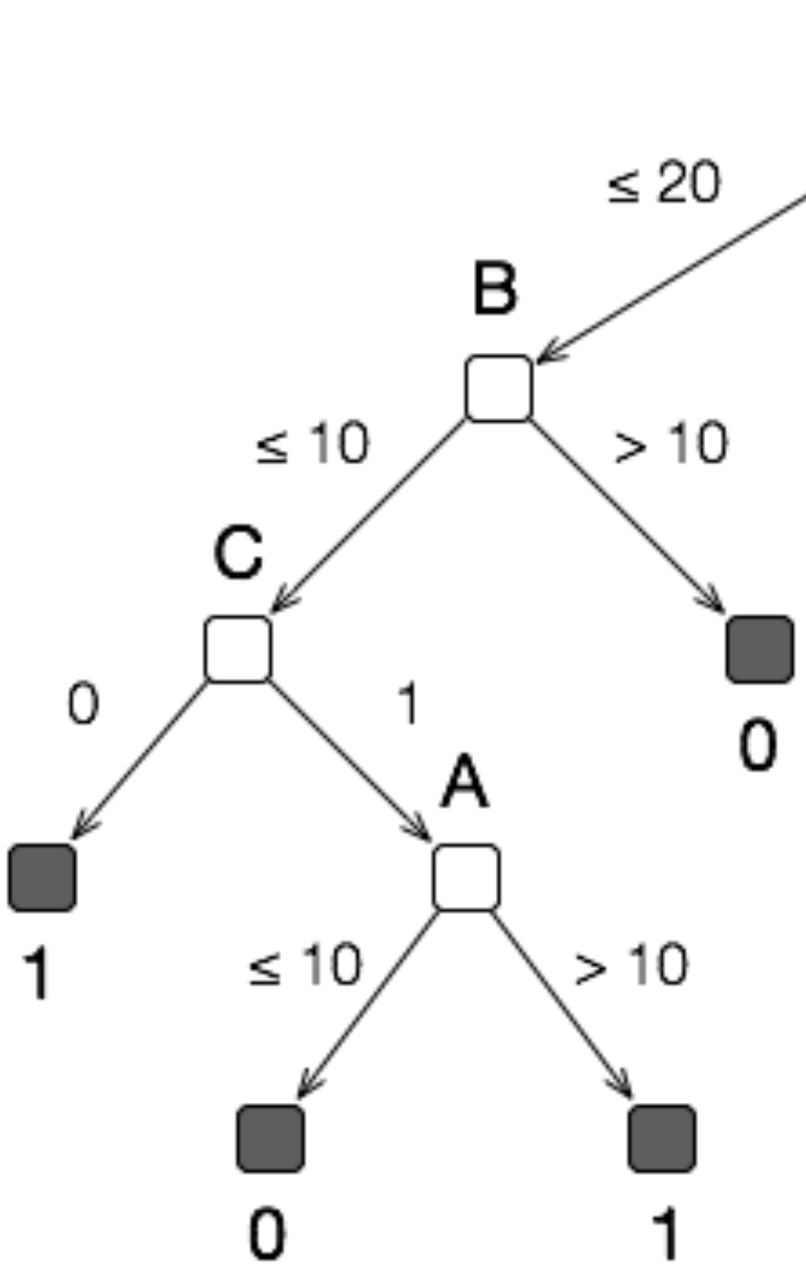


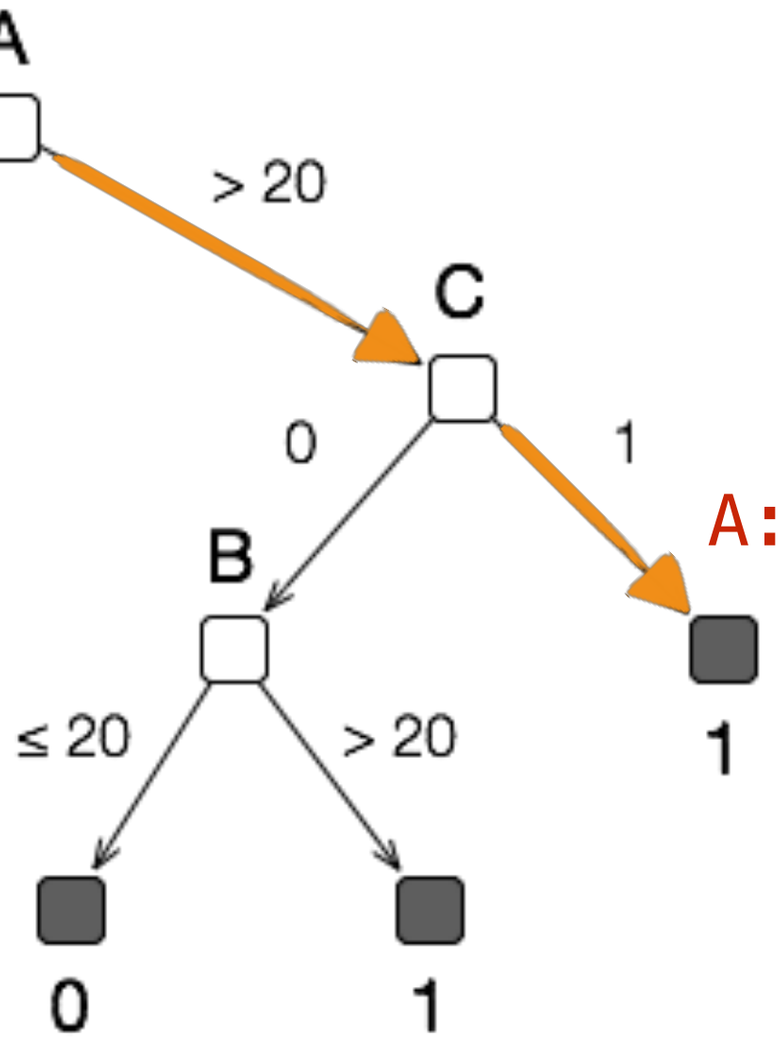
Classification example





Classification example





A: 30, B: 20, C: 1
class = 1

How do we embed a DT in CP?

- A decision variable for each attribute

```
A ∈ {-inf, inf}
B ∈ {-inf, inf}
C ∈ {0, 1}
```

- A decision variable for the class

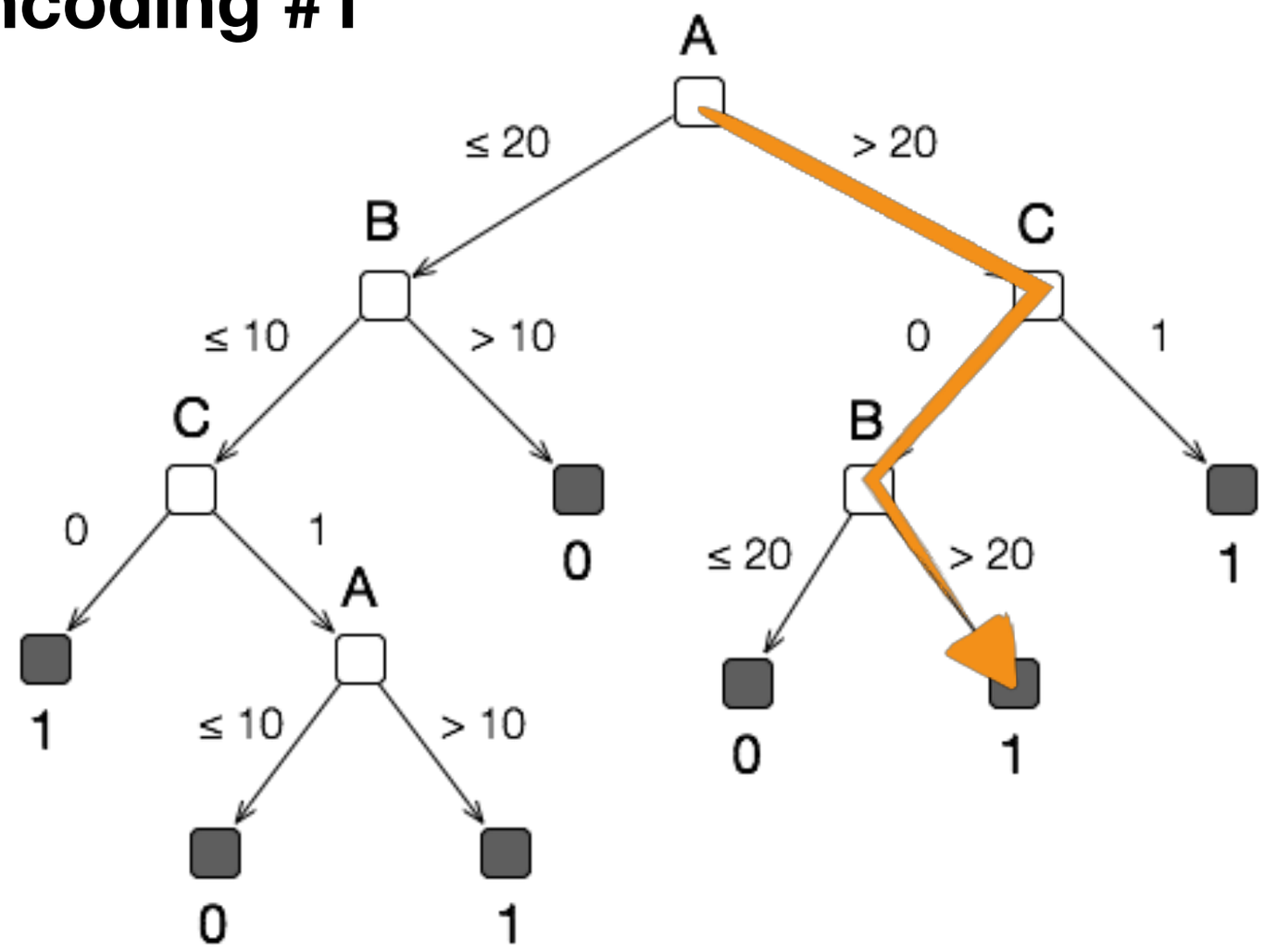
```
Y ∈ {0, 1}
```

- Enforce consistency on:

$$Y = DT(A, B, C)$$

This is the tricky part

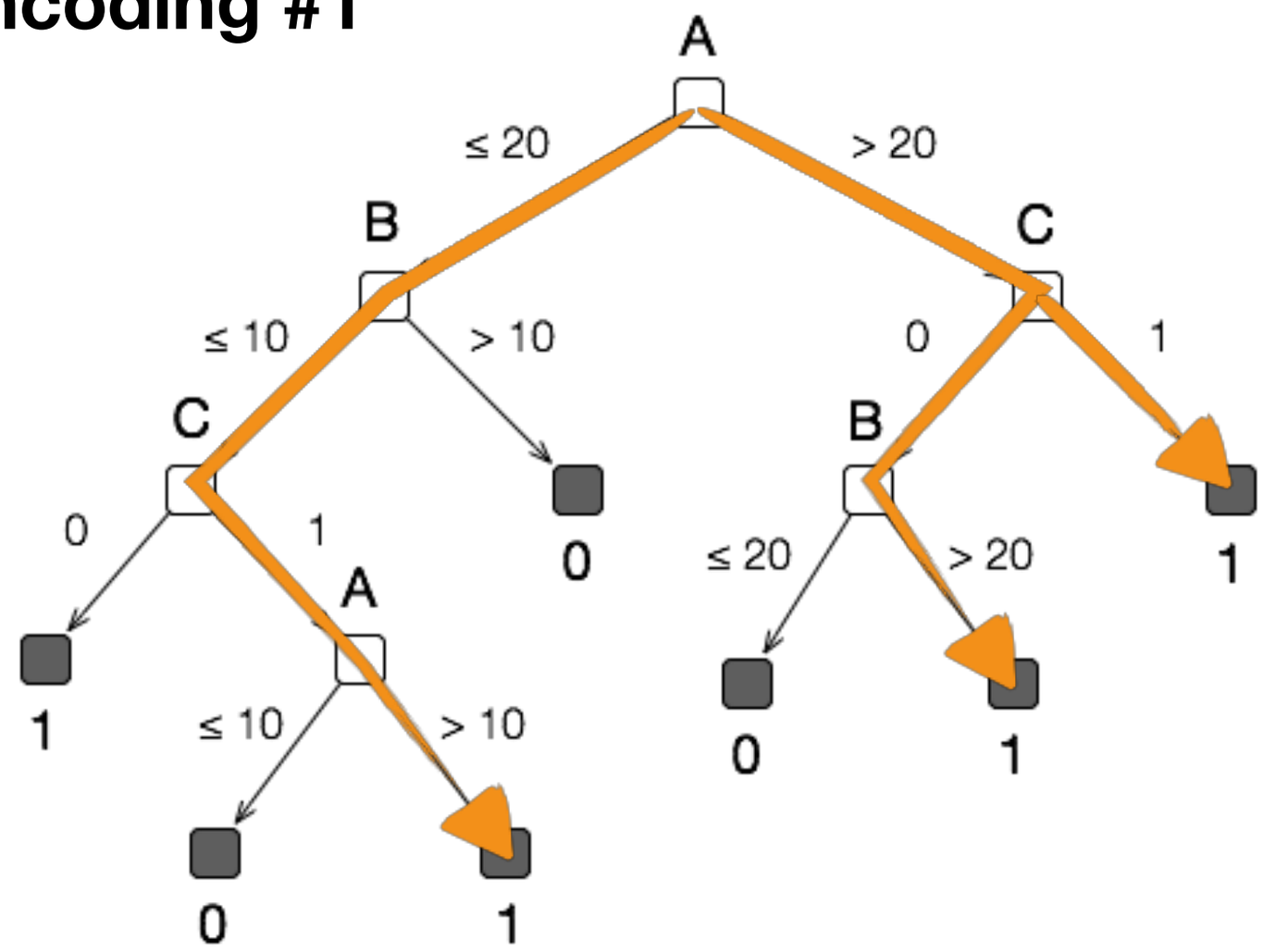
Encoding #1



A path is an implication

$$[A > 20] \wedge [C = 0] \wedge [B > 20] \Rightarrow [Y = 1]$$

Encoding #1

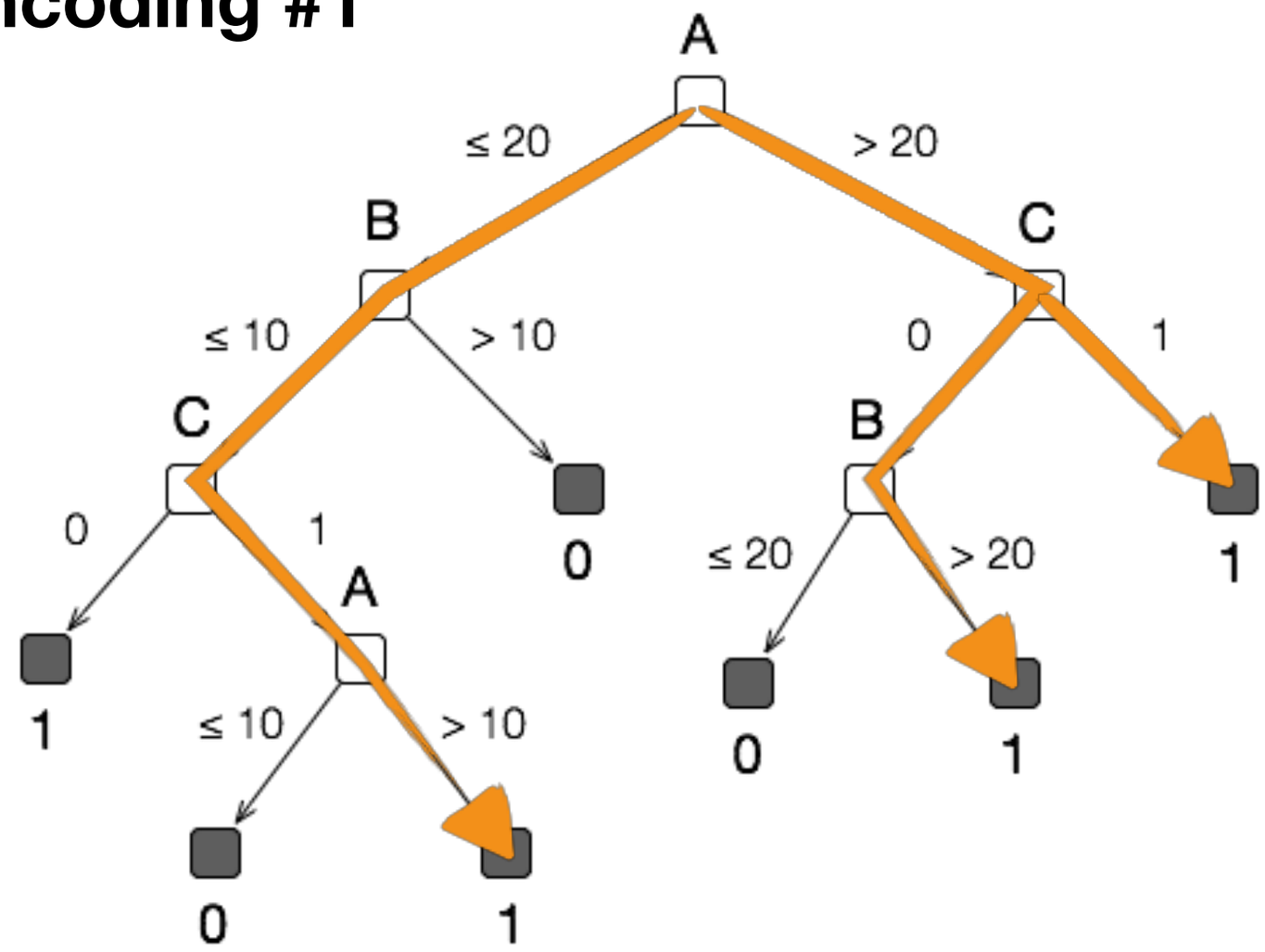


A path is an implication

The tree defines all the paths

$$
\begin{aligned}
 & ([A > 20] \land [C = 0] \land [B > 20]) \lor \\
[Y = 1] \Leftrightarrow\ & ([A > 20] \land [C = 1]) \lor \\
 & ([A \leq 20] \land [B \leq 10] \land [C = 1] \land [A > 10])
\end{aligned}
$$

Encoding #1



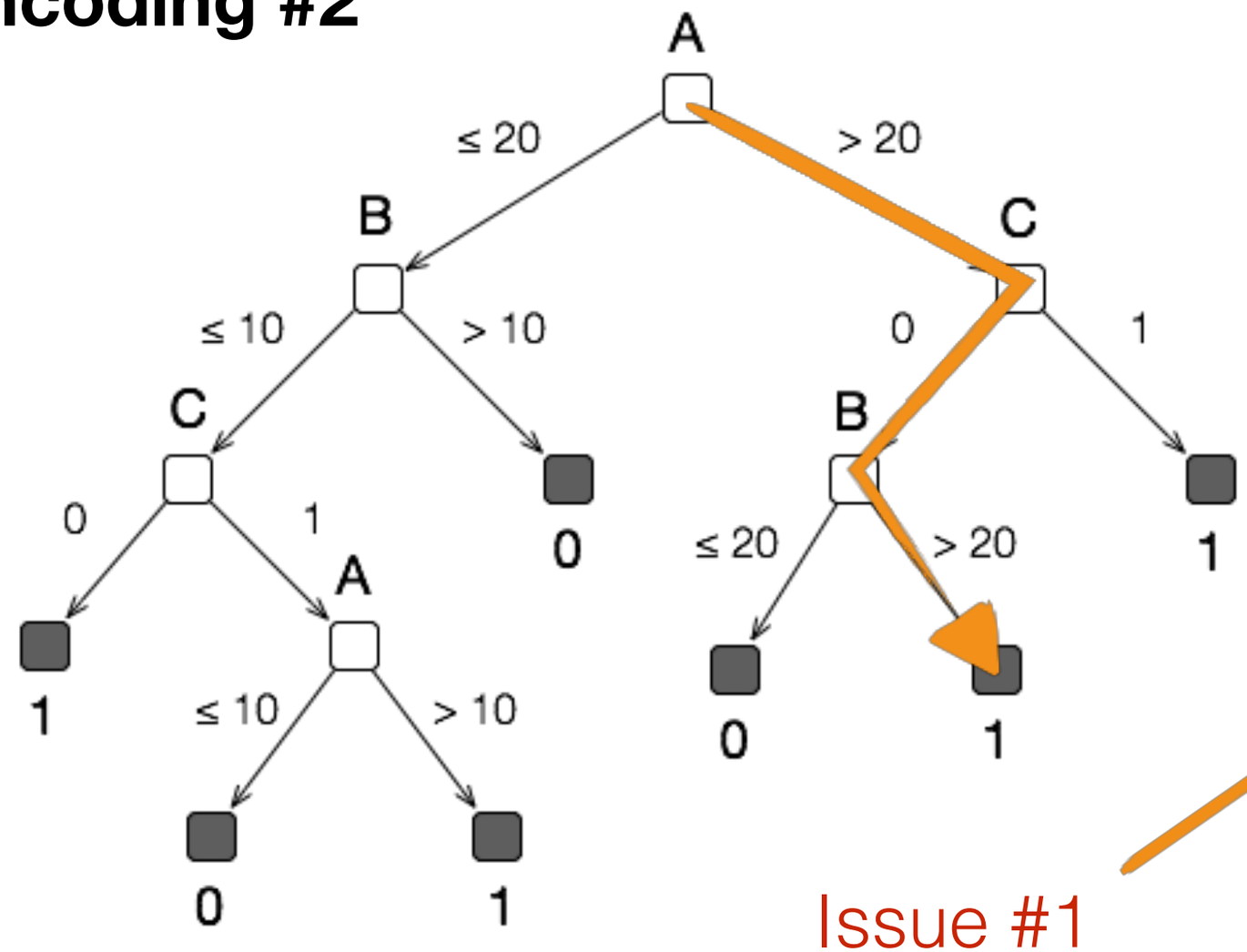
A path is an implication

The tree defines all the paths

- Polynomial size
- Polynomial time propagation
- Does not enforce GAC
- Very suitable for SMT

Can we do better?

Encoding #2



Path ↔ set of feasible assignments

A	B	C	Y
21	21	0	1
21	22	0	1
21	23	0	1
…	…	…	…

Issue #1

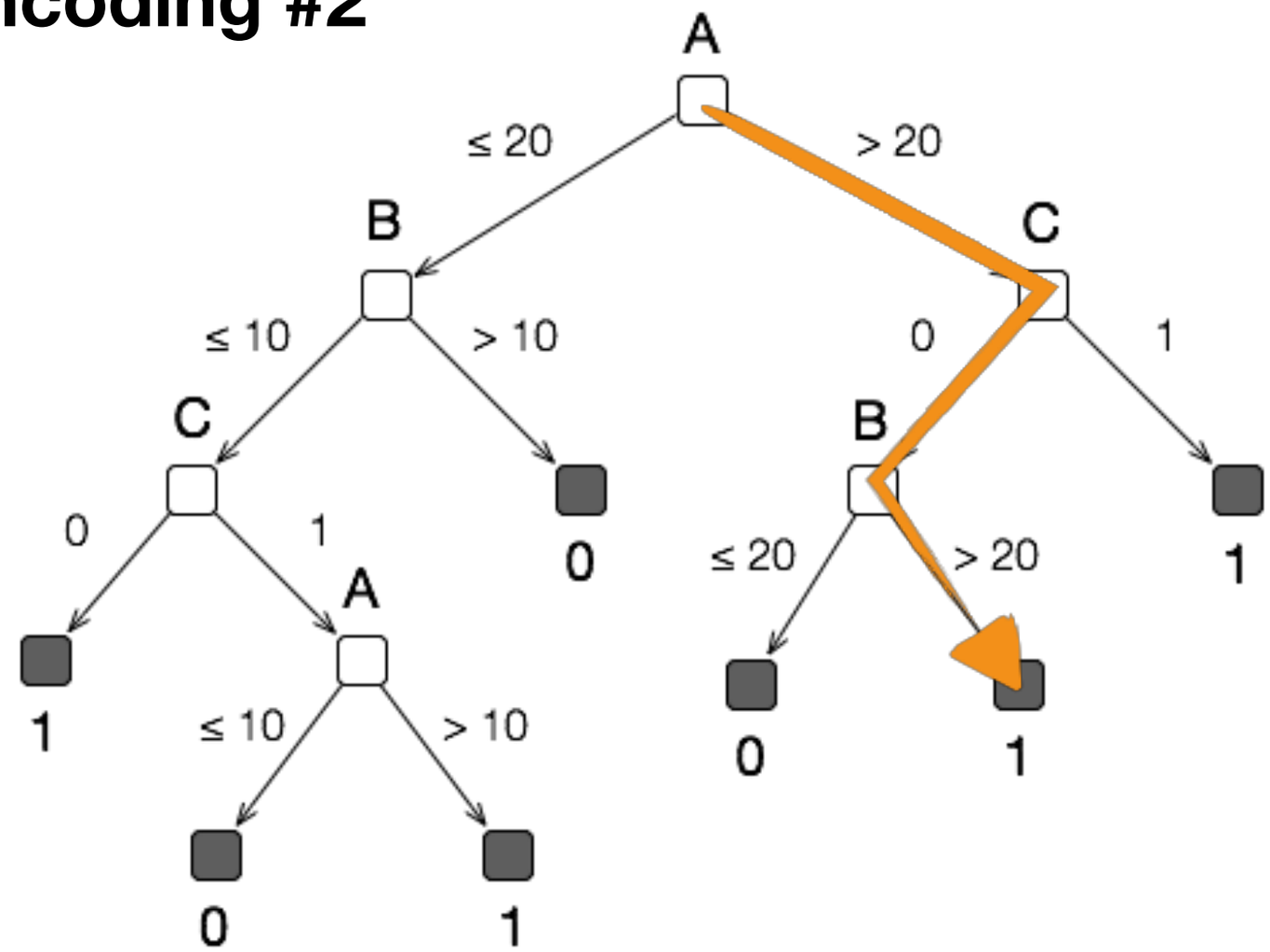
numeric attributes

use discretization

We can use a table constraint!

Bessiere, Regin, IJCAI 97

Encoding #2



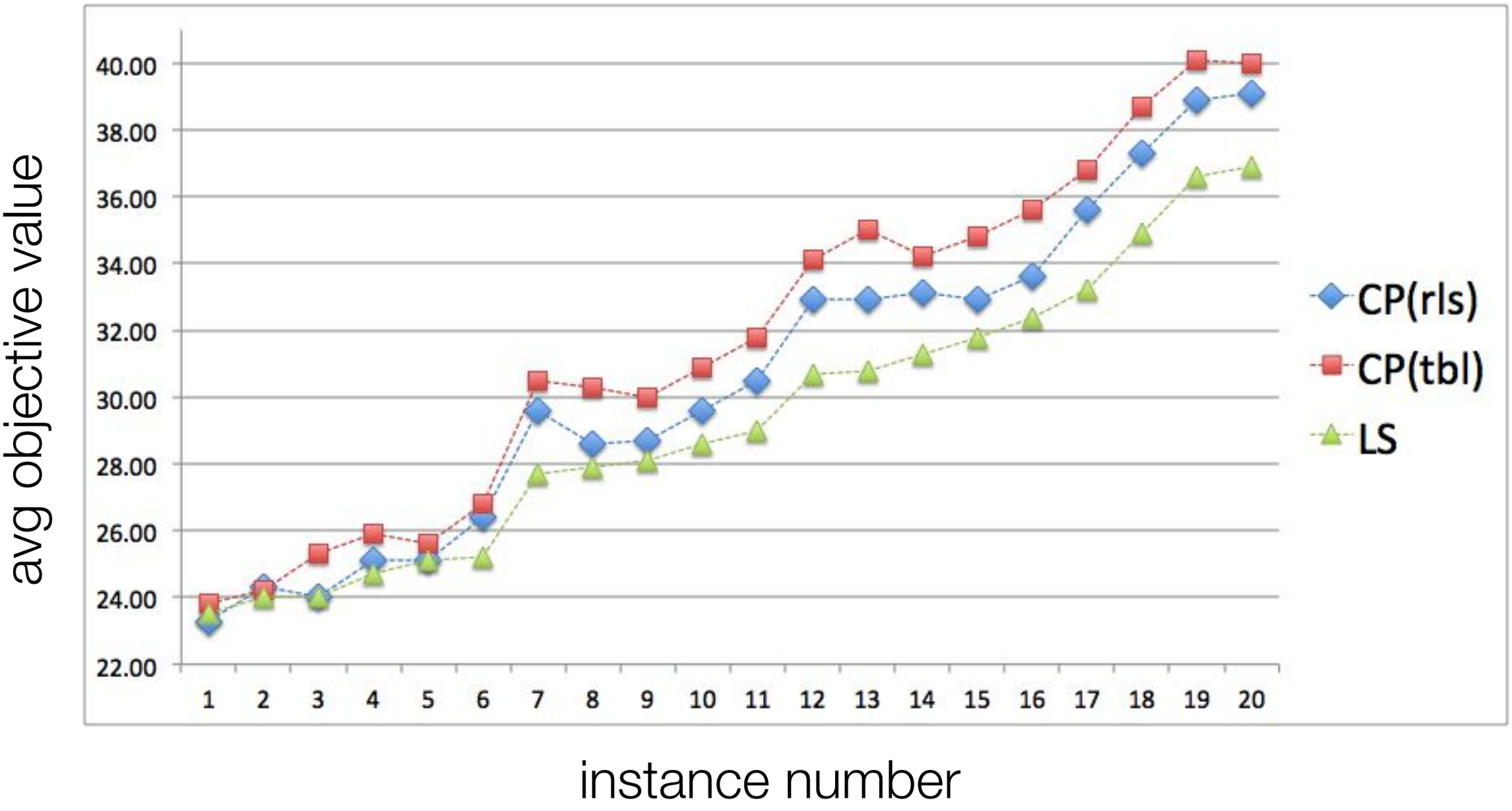
Path ↔ set of feasible assignments

A	B	C	Y
21	21	0	1
21	22	0	1
21	23	0	1
…	…	…	…

Continuous attributes can be discretised using the splitting thresholds appearing in the tree

Solution quality

- 20 instances, 48 cores, 288 jobs
- CP with LNS against Localsolver, 90 sec time limit



More compact encodings:

- **use an MDD**
- **use compressed tuples**
- **convert to a sNNF - sdNNF**
- **decomposition approaches**

Technical report

Variants

- **Random forests**
- **Regression trees**
- **Multi-output trees**

Concluding remarks

In a nutshell:

EML enables combinatorial optimization over complex real world systems

- It's not the only solution…
- …and it does not need to be about CP

But CP is very well suited for EML!

- Ease of embedding (a EM in a global constraint)
- Flexibility and modularity (the EM is "just a constraint")

In a nutshell:

EML enables combinatorial optimization over complex real world systems

It’s not the only solution…

…and it does not need to be about CP

But CP is very well suited for EML!

- Ease of embedding (a Neural Network in a global constraint)
- Flexibility and modularity (the EM is “just a constraint”)

Which issues shall we address to make it work?

Issue #1

- You can always embed a ML in (e.g.) Local Search
- Can CP work better?

Yes (we have data)! Mainly thanks to propagation

Issue #1

- You can always embed a ML in (e.g.) Local Search
- Can CP work better?

Yes (we have data)! Mainly thanks to propagation

We need inference methods for ML models

(e.g. bounds on the I/O of a Neural Network)

- Inference should be effective (tight bounds)
- Inference should be efficient

Issue #2

Complex ML models

more accurate

slow, weak propagation

Risk: poor quality solutions

Simple ML models

less accurate

effective propagation

Risk: apparently good solutions

Issue #2

Complex ML models	Simple ML models
more accurate	less accurate
slow, weak propagation	effective propagation
Risk: poor quality solutions	Risk: apparently good solutions

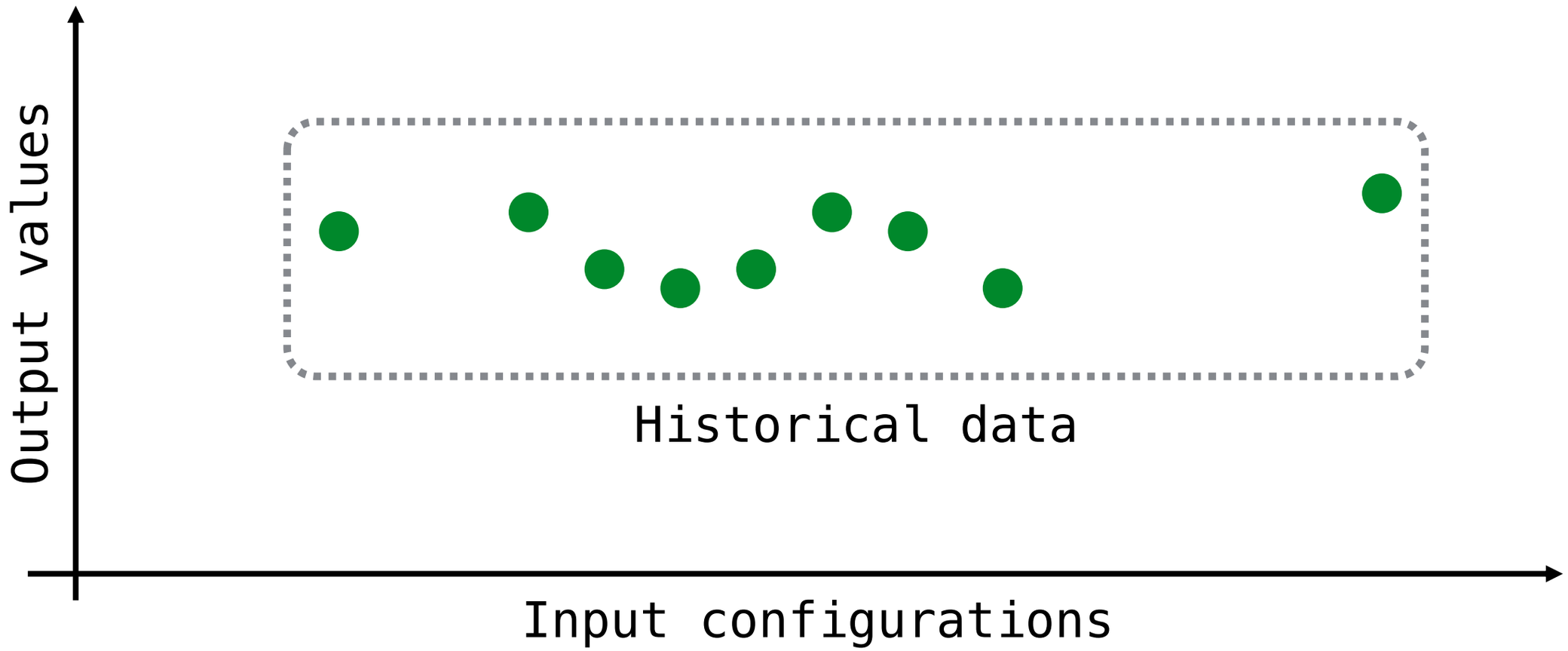
There is an accuracy/optimization trade-off

- How to characterize it?
- Can we find design rules?

We are currently working on Deep NN with a Google Faculty Award

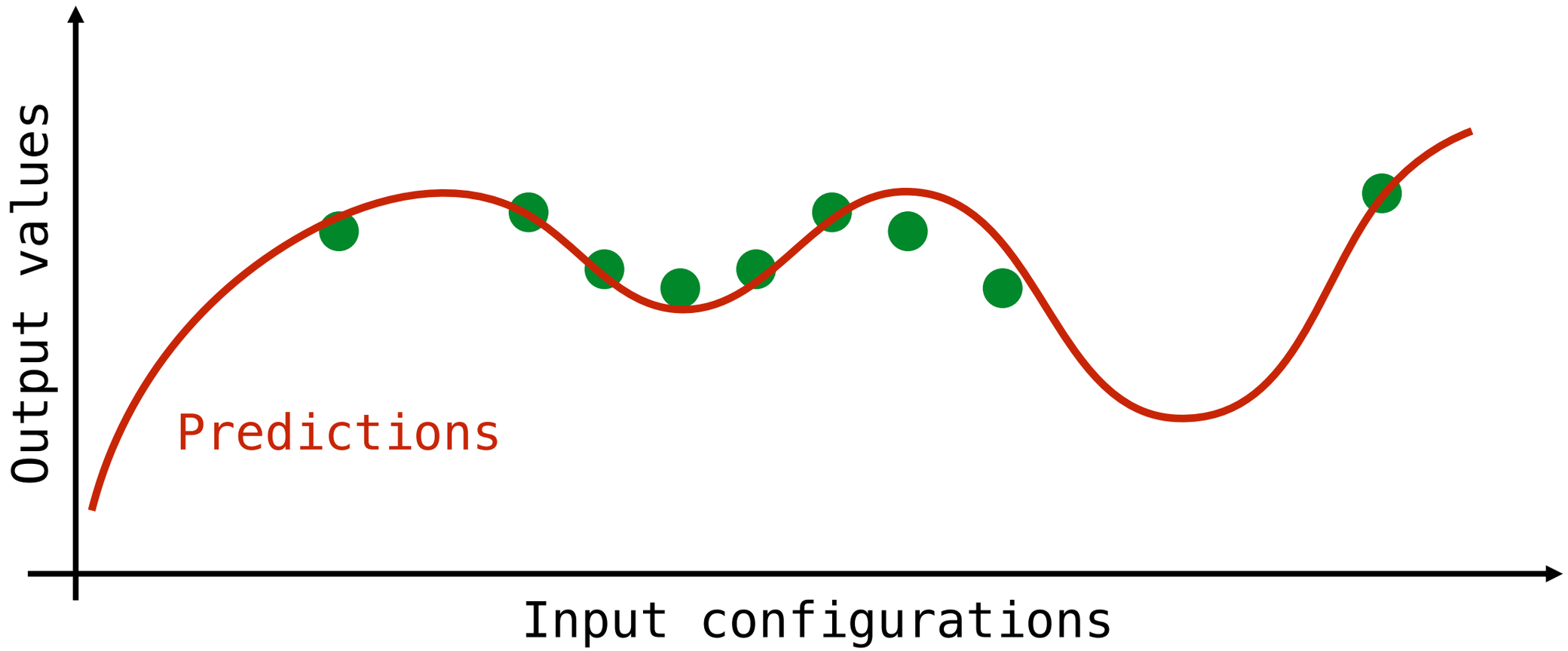
Issue #3

A typical training set in ML looks like this:



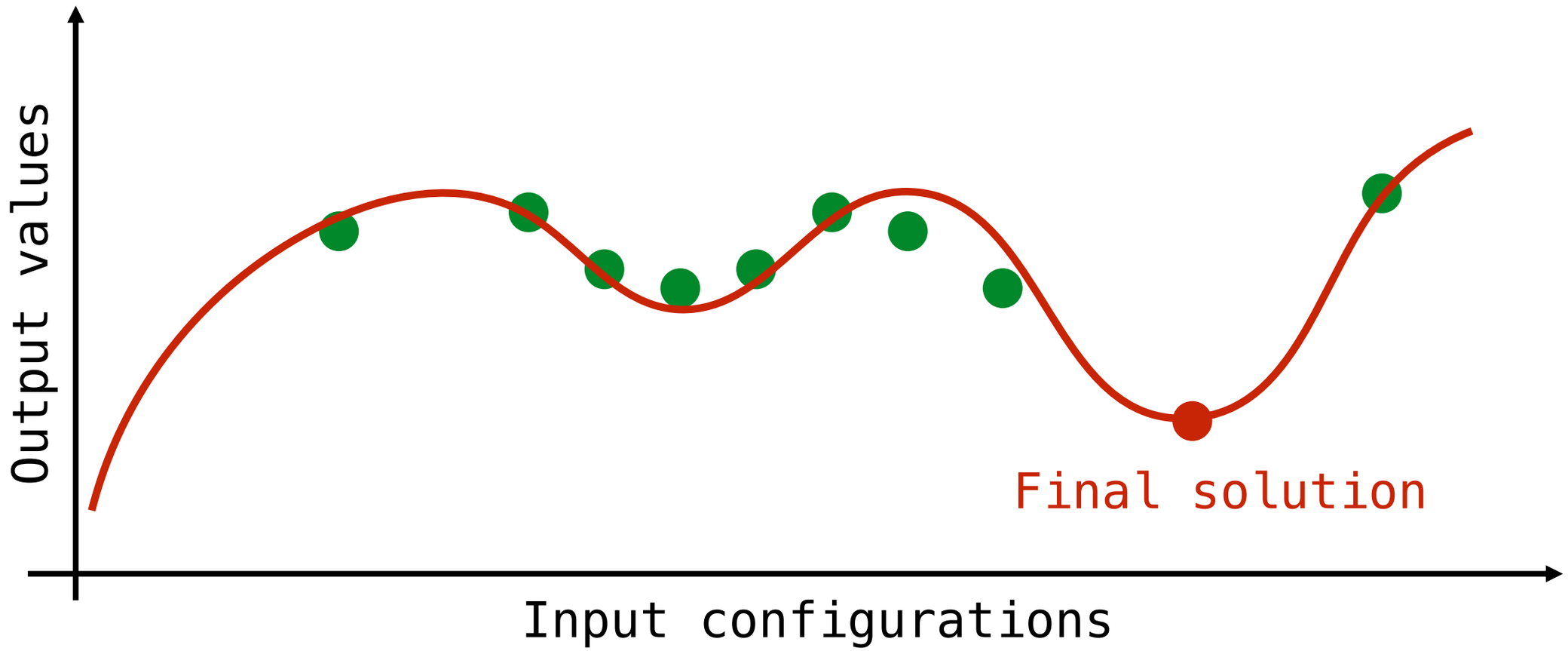
Issue #3

The ML model provides a prediction for every input value



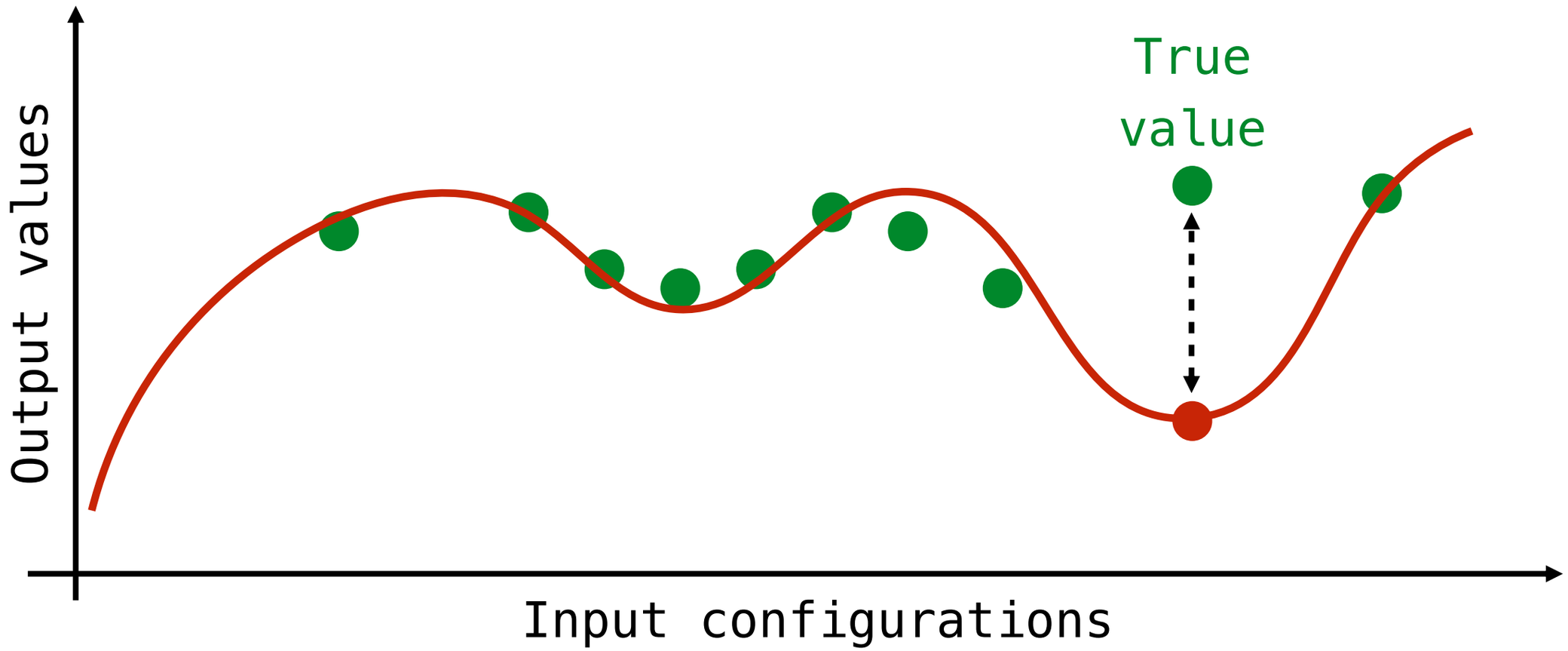
Issue #3

The optimizer will search for the best one (HP: prediction = cost)



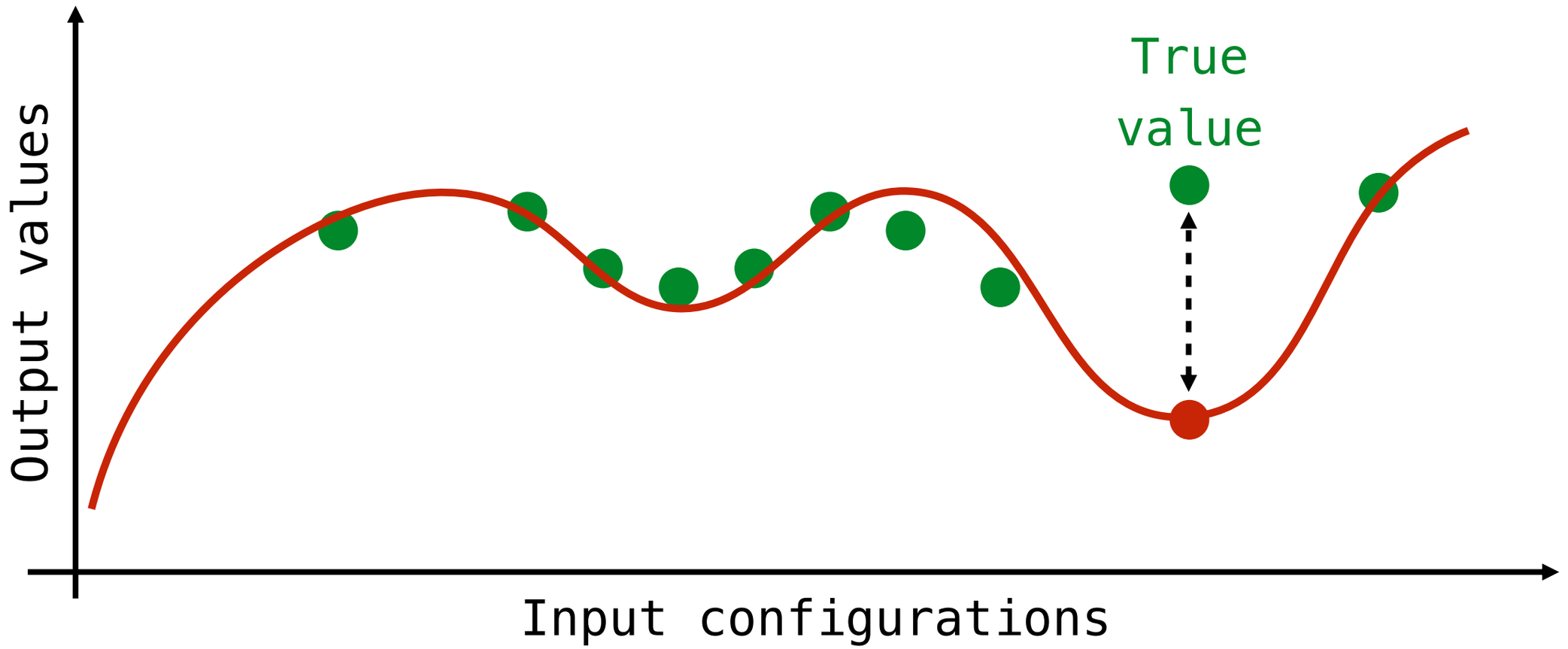
Issue #3

If this is far from known examples, there may be a large error



Issue #3

If this is far from known examples, there may be a large error



How do we choose representative observations?

Borrow concepts from black-box optimization, surrogate model construction by adaptive sampling

Issue #4

Many ML model provide estimated confidence levels

Can we take them into account when optimizing?

Issue #4

Many ML model provide estimated confidence levels

Can we take them into account when optimizing?

Some ML models can deal with uncertain inputs

Can we employ EML in stochastic optimization?

Some errors are more *important* than other

Extensions

- Hierarchical optimization:
- The higher levels in the hierarchy do not need to know all the details of the lower levels.
- We can learn their behaviour and cast this into the higher level oprimizer

- Off-line/on-line optimization
- Let the off-line solver learn the behaviour of the on-line one

Material

- https://emlopt.github.io/

Description

Application examples

Resources:

1. Papers,
2. Running survey,
3. EMLlib embedding techniques,
4. Pre- and post- processing methods
5. I/O support (in particular readers for popular ML libraries)
6. Tutorial with an hands-on example

Thanks for listening!

Questions?