Efficient **Auto-Tuning** of Parallel Programs with **Interdependent Tuning Parameters** via **Auto-Tuning Framework (ATF)**

Ari Rasch, Richard Schulze, Michel Steuwer,
Sergei Gorlatch

⚠ Please Note ⚠

This talk will be on a quite high level:

- **focus of this talk are the “what” & “why” questions;**
- **we address the “how” question by illustrating our basic ideas only;**
- **details about our approach can be found in the paper.**

What is *Auto-Tuning*?

Auto-Tuning (AT) aims at automatizing the process of code optimization:

Auto-Tunable Optimization Process		**Auto-Tuned Optimization Process**
`ppcg … -tile_size = ???` *Polyhedral Compiler*	**AT** →	`ppcg … -tile_size = 128` *Polyhedral Compiler*
`lift … -split_size = ???` *Lift Compiler*	**AT** →	`lift … -split_size = 32` *Lift Compiler*
`XgemmDirectTN … -VWM = ??? -MDIMC = ???` *CLBlast JIT Compiler*	**AT** →	`XgemmDirectTN … -VWM = 4 -MDIMC = 64` *CLBlast JIT Compiler*
`mdh … -NUM_THR_<l,d> = ??? -MEM_INP_MDA = ???` *MDH Compiler*	**AT** →	`mdh … -NUM_THR_<l,d> = 10 -MEM_INP_MDA = LCL` *MDH Compiler*
Generic Optimization Parameters		***Instantiated Optimization Parameters***

What is *Auto-Tuning*?

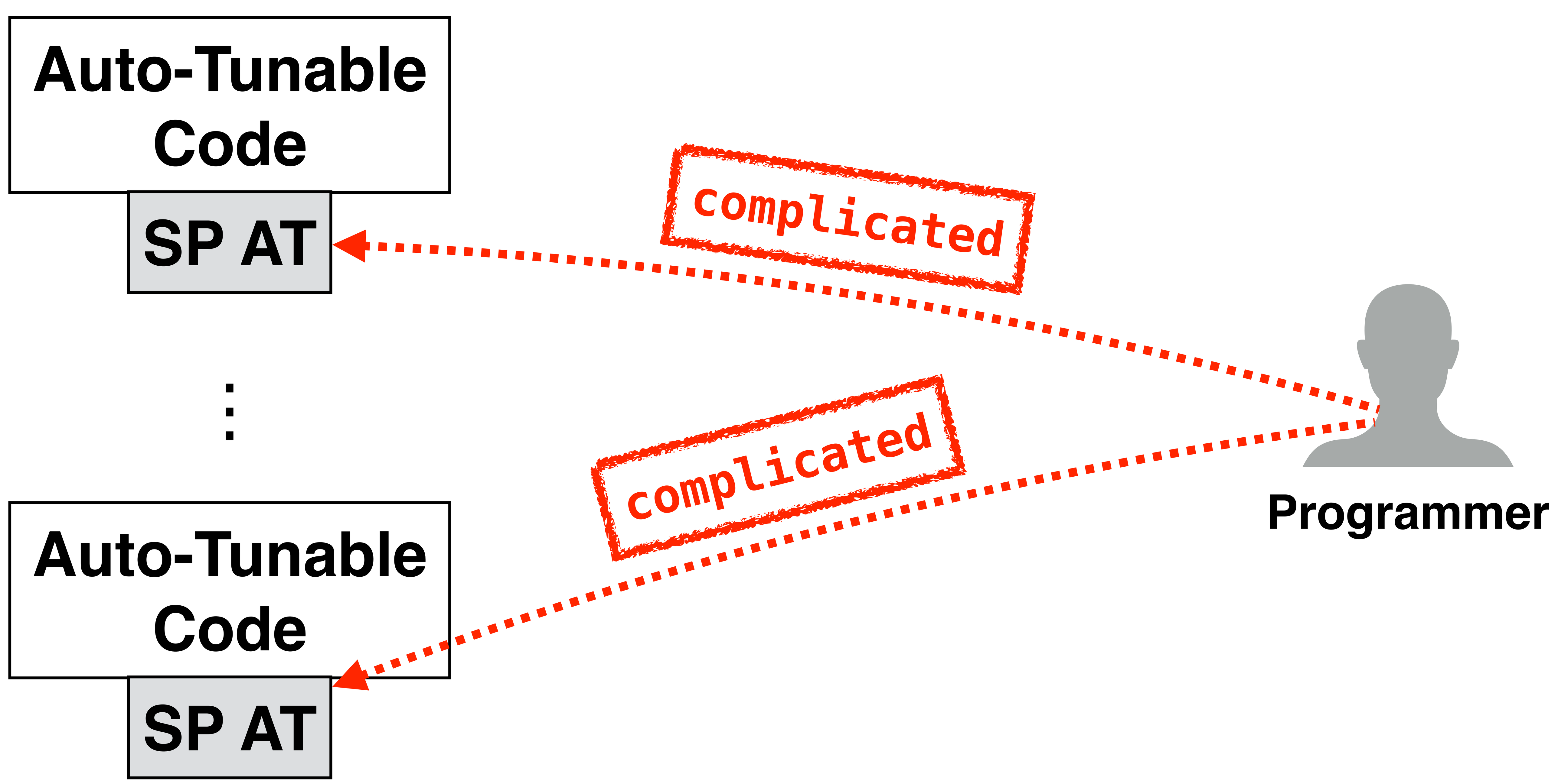
Auto-Tuning usually consists of three major phases:

What is *Auto-Tuning*?

Auto-Tuning (AT) can be categorized into two major categories:

Special-Purpose (SP) Auto-Tuning

Basic Idea:

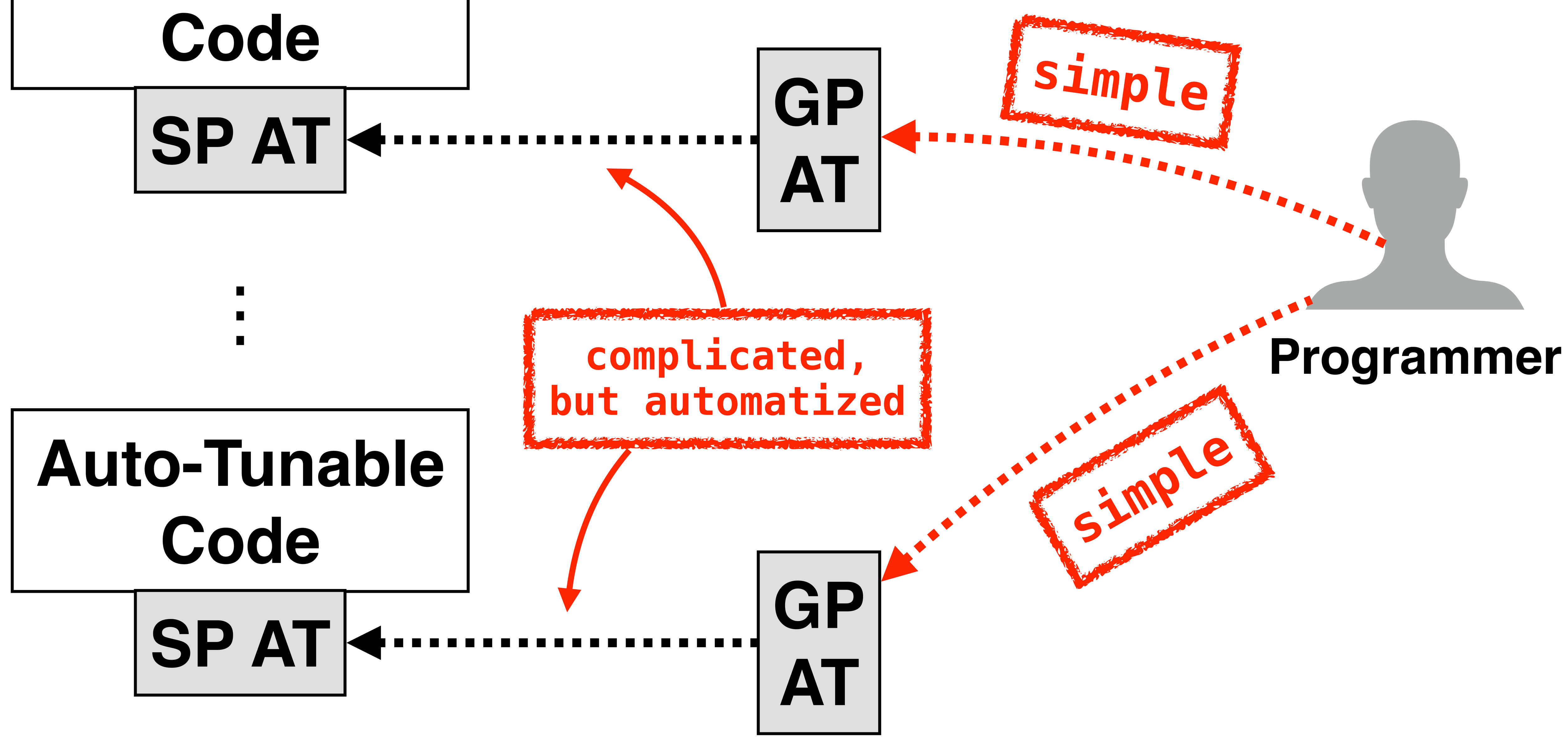


Notable Examples:

ATLAS, FFTW, MilePost, CHiLL, PolyMage, PATUS, SPIRAL, Nitro, Apollo

General-Purpose (GP) Auto-Tuning

Basic Idea:



Notable Examples:

OpenTuner, libTuning, Orio, ActiveHarmony, KernelTuner, CLTune

What is ***Auto-Tuning***?

<u>Auto-Tuning (AT) can be categorized into two major categories:</u>

<u>Special</u>-Purpose Auto-Tuning

Special-Purpose Auto-Tuners usually achieve good tuning results 🙂👍

<u>However:</u> they have to be designed & implemented from scratch for each new target application, which requires expert knowledge and is cumbersome 😕👎

<u>General</u>-Purpose Auto-Tuning

General-Purpose Auto-Tuners automatically generate special-purpose tuners 🙂👍

<u>However:</u> current approaches struggle with programs that have *<u>interdependent</u>* tuning parameters 😕👎

What are *Interdependent Tuning-Parameters*?

Independent Parameters

Parameters:

$\texttt{tile\_size} \in \{1, \ldots, 128\}$

$\texttt{mem\_region} \in \{\texttt{GLB}, \texttt{LCL}, \texttt{PRV}\}$

$\texttt{num\_threads} \in \{1, \ldots, 1024\}$

Constraints:

<none>

Configurations:

$\{\ (1, \texttt{GLB}, 1)\ , \ldots ,\ (128, \texttt{PRV}, 1024)\ \}$

Each combination of parameters' values represents a valid parameter configuration

Interdependent Parameters

Parameters:

$\texttt{tile\_size\_1} \in \{1, \ldots, 128\}$

$\texttt{tile\_size\_2} \in \{1, \ldots, 128\}$

$\texttt{tile\_size\_3} \in \{1, \ldots, 128\}$

Constraints:

$\texttt{tile\_size\_3} \mid \texttt{tile\_size\_2} \mid \texttt{tile\_size\_1}$

Configurations:

$\{\ (1,1,1),\ (1,1,2),\ , \ldots ,\ (128,128,128)\ \}$

tile_size_2 **not multiple of** *tile_size_3*

Only combinations that satisfy the constraints represent valid configurations

Interdependent Tuning-Parameters in General-Purpose Auto-Tuning

Current approach have either ***no support*** or only ***limited efficiency*** for programs with interdependent tuning parameters:

`OpenTuner` `Orio` `libTuning`

no support: invalid configurations are kept in the search space, which severely hinders the tuners from finding well-performing configurations

---

`ActiveHarmony` `CLTune` `KernelTuner`

limited efficiency: the approaches are efficient for small search spaces only, because of sub-optimal process to generating, storing, and exploring the search spaces of interdependent parameters

Goal of this Work

We present three new contributions of our general-purpose Auto-Tuning Framework (ATF):

1. ATF *generates* the search space of interdependent parameters with high performance
2. ATF *stores* such spaces with low memory footprint
3. ATF *explores* these spaces efficiently

Parameter Constraints

Chain-of-Trees Structure

1. Generation

ATF relies on *parameter constraints*, rather than traditional *search space constraints*:

ATF's parameter constraints enable:

1. Checking Constraints Early
2. Generating groups of interdependent parameters independently & in parallel
3. Generating individual groups in parallel

2. Storing

Basic Idea

ATF relies on a new *chain-of-trees* search space structure & *parameter constraints*:

$p_1 := (n_1,$	$\{22, 35\},$	_	)
$p_2 := (n_2,$	$\{2, 5, 7, 11\},$	divides(n_1)	)
$p_3 := (n_3,$	$\{26, 51\},$	_	)
$p_4 := (n_4,$	$\{1, 3, 13, 17\},$	divides(n_3)	)
$p_5 := (n_5,$	$\{27, 39, 52, 54, 68\},$	equals($n_3 + n_4$)	)

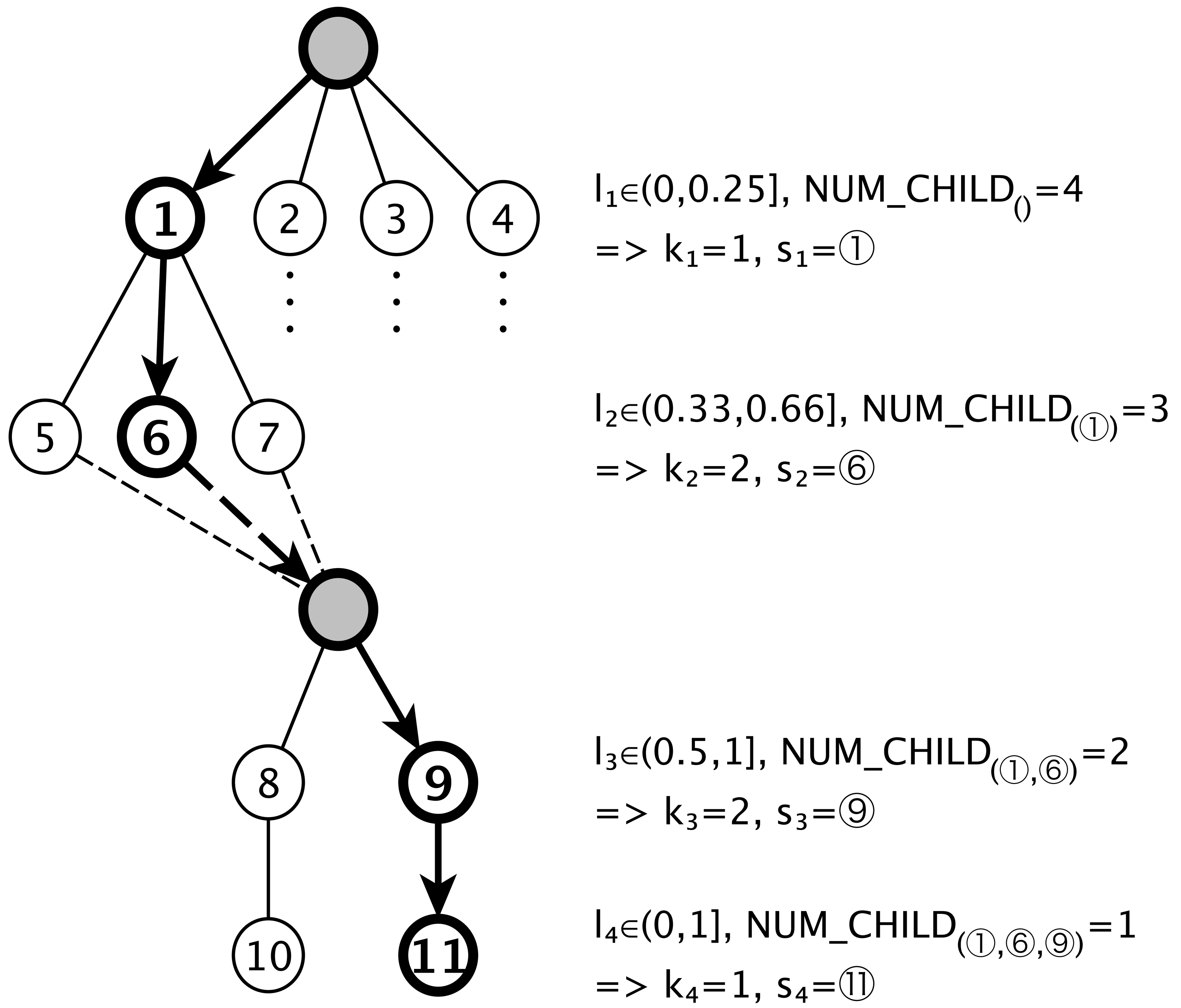
Example Parameters

ATF's search space structure avoids memory-intensive redundancies

3. Exploration

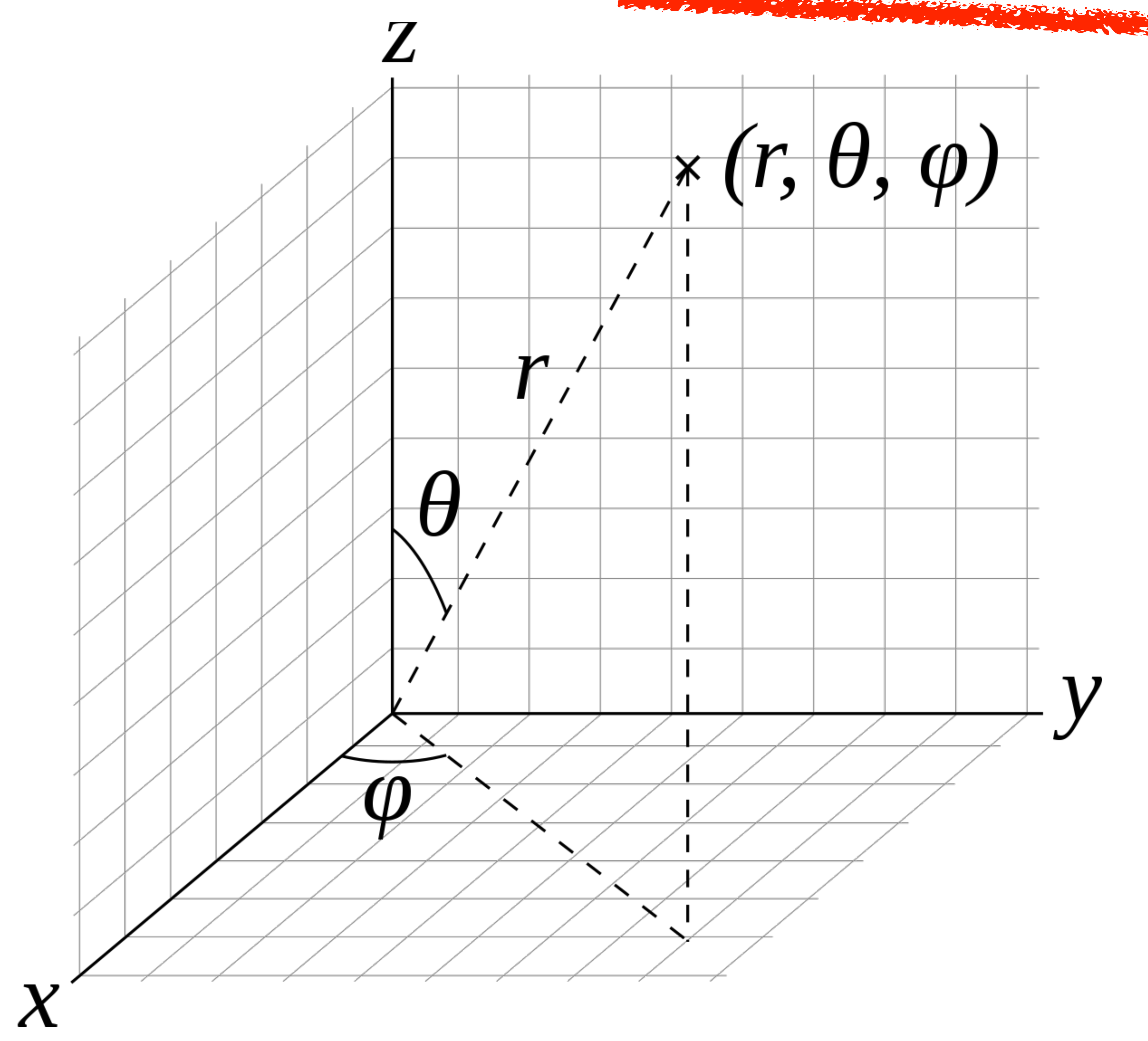
Basic Idea

ATF exploits its new *chain-of-trees* search space structure for a multi-dimensional search:



ATF's
Search Space

Efficient Structure for Search Techniques



Coordinate Space

ATF's search space structure enables reducing the complexity of exploration to exploring a Coordinate Spaces

Experimental Evaluation

Highlights

ATF is able to auto-tune important applications for ***CPU & GPU*** on ***real-world data sets*** to high performance:

Stencil

ATF is able to auto-tune the `CONV` implementation in [1] to:

>40x higher performance than CONV+CLTune on CPU

>10⁴x higher performance than CONV+CLTune on GPU

>3x higher performance than Intel MKL-DNN on CPU

>15x higher performance than NVIDIA cuDNN on GPU

Linear Algebra

ATF is able to auto-tune the `GEMM` implementation in [1] to:

>2x higher performance than GEMM+CLTune on CPU

>120x higher performance than GEMM+CLTune on GPU

>2x higher performance than Intel MKL on CPU

>2x higher performance than NVIDIA cuBLAS on GPU

Quantum Chemistry

ATF is able to auto-tune the `CCSD(T)` implementation in [1] to:

>2x higher performance than TensorComprehensions on GPU

CLTune **fails**!
(too high search space generation time)

Data Mining

ATF is able to auto-tune the `PRL` implementation in [1] to:

>1.66x higher performance than PRL+CLTune on CPU

>1.07x higher performance than PRL+CLTune on GPU

OpenTuner fails for all applications because of a too high amount of invalid configurations within its search space

[1] Rasch, Schulze, Gorlatch. "Generating Portable High-Performance Code via Multi-Dimensional Homomorphisms", PACT'19

ATF — User Interface

ATF's user interface is focus of our previous work [1]:

```
#atf::tp name       NUM_WG_1
         range      interval<int>( 1, N_1 )

#atf::tp name       NUM_WI_1
         range      interval<int>( 1, N_1 )

// ...

#atf::tp name       LM_SIZE_1
         range      interval<int>( 1, N_1 )
         constraint LM_SIZE_1 <= N_1

#atf::tp name       PM_SIZE_1
         range      interval<int>( 1, N_1 )
         constraint PM_SIZE_1 <= LM_SIZE_1

// …

// OpenCL kernel code
```

Auto-Tuning in ATF via easy-to-use *Tuning Directives* [1]

(ATF is also available as programming library in C++ [2] / Python (WIP)— for online auto-tuning)

[1] Rasch, Gorlatch. "ATF: A Generic, Directive-Based Auto-Tuning Framework", Concurrency and Computation: Practice and Experience, 2019

[2] Rasch, Haidl, Gorlatch. "ATF: A Generic Auto-Tuning Framework", HPCC, 2017

Questions?

Ari Rasch
a.rasch@wwu.de