Chair of Scientific Computing in Computer Science
Univ.-Prof. Dr. Hans-Joachim Bungartz

Technische Universität München

Bachelor Thesis Final Presentation

Exploring Fuzzy Tuning Technique for Molecular Dynamics Simulations in AutoPas

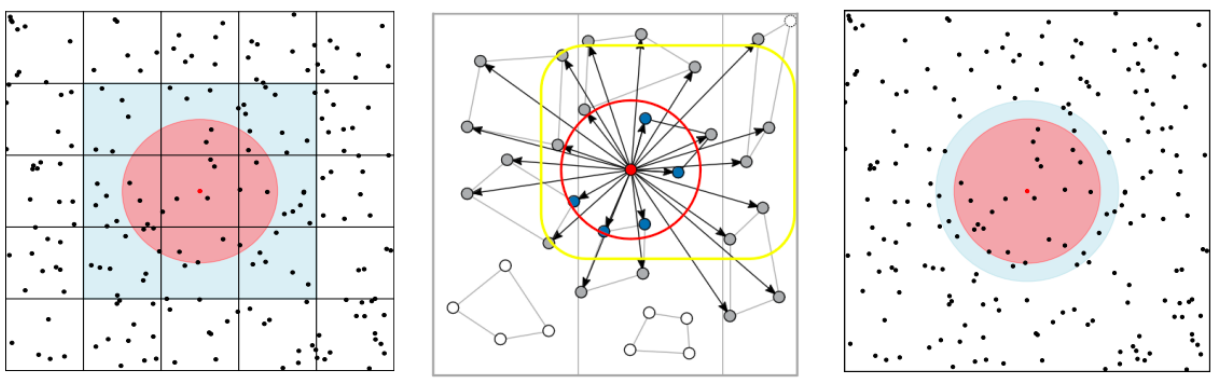
Manuel Lerchner
manuel.lerchner@tum.de

Advisors:
Manish Kumar Mishra, M.Sc.
Samuel James Newcome, M.Sc.

AutoPas

What is AutoPas?

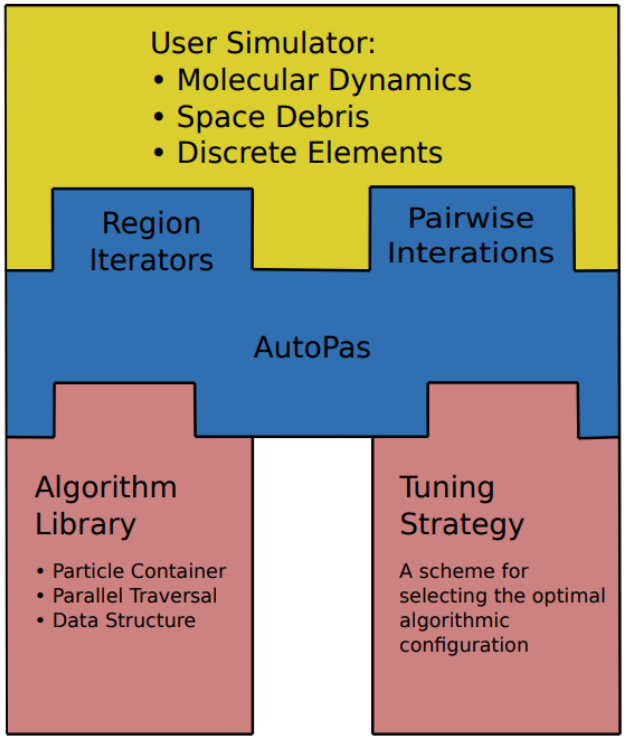
- Library for arbitrary N-body simulations
- Optimal performance by switching implementations
  - **Container:** Finding neighboring particles
  - **Traversal:** Parallel force calculations
  - **Data Layout:** Memory access optimization
  - **Newton 3:** Force calculation optimization



[Newcome et al., 2024]

Structure of AutoPas

- Three main areas:
  - User Application
  - Algorithm Library
  - Tuning Strategies
- Algorithm Library:
  - Huge Search Space[1]
- Tuning Strategies:
  - Full Search
  - Random Search
  - Predictive Tuning
  - Bayesian Search
  - Rule Based Tuning

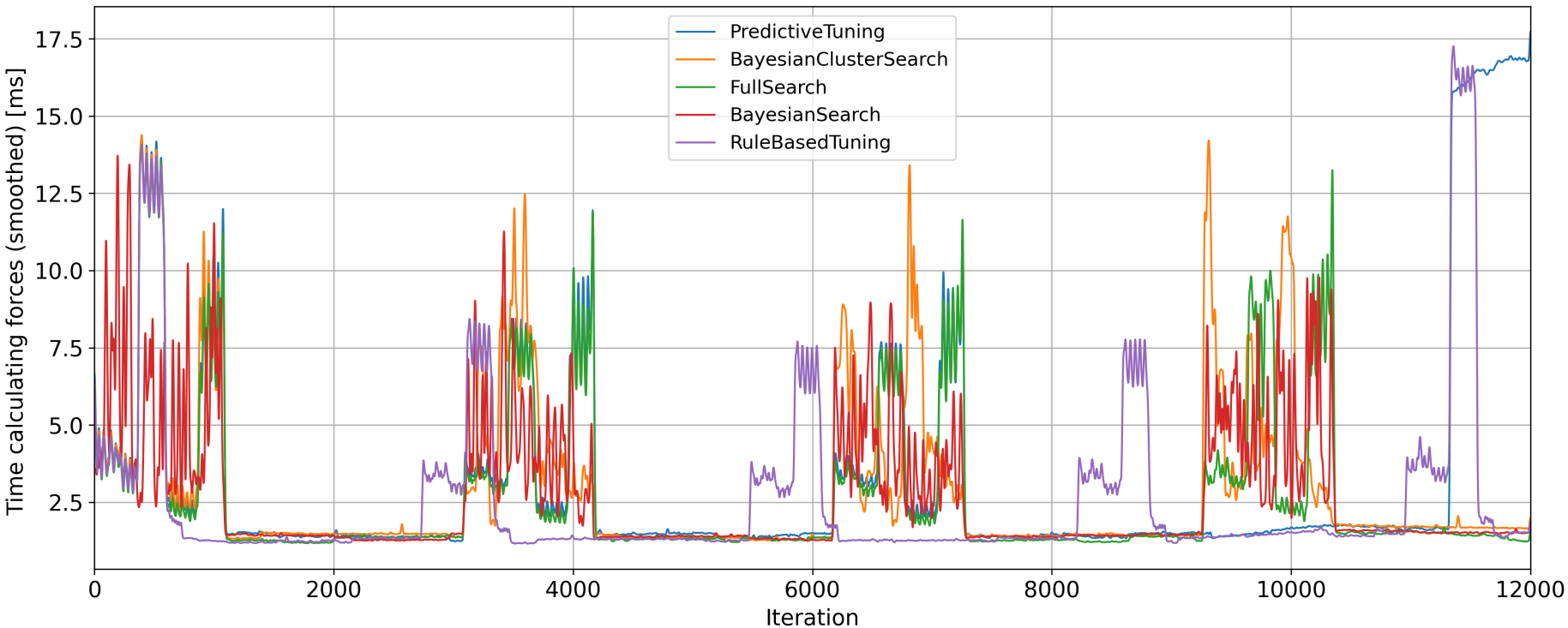


Source: [Newcome et al., 2023]

[1] Container × Traversal × Data Layout × Newton 3 × Load Estimator × Cell Size Factor

Auto-Tuning

- Tuning Phase → Simulation Phase → Repeat
- Potential Tuning Overhead



Fuzzy Logic Systems

- Human-like reasoning to model systems $f : \mathbb{R}^n \to \mathbb{R}$
- (Fuzzy) Rule based systems
  - Interpolation effect between conflicting rules
  - No hard boundaries $\to$ robust against noise
- Smooth transition instead of hard boundaries
  - E.g. *cold, warm, hot* instead of *20° C, 30° C, 40° C*

Example (Heater Control)

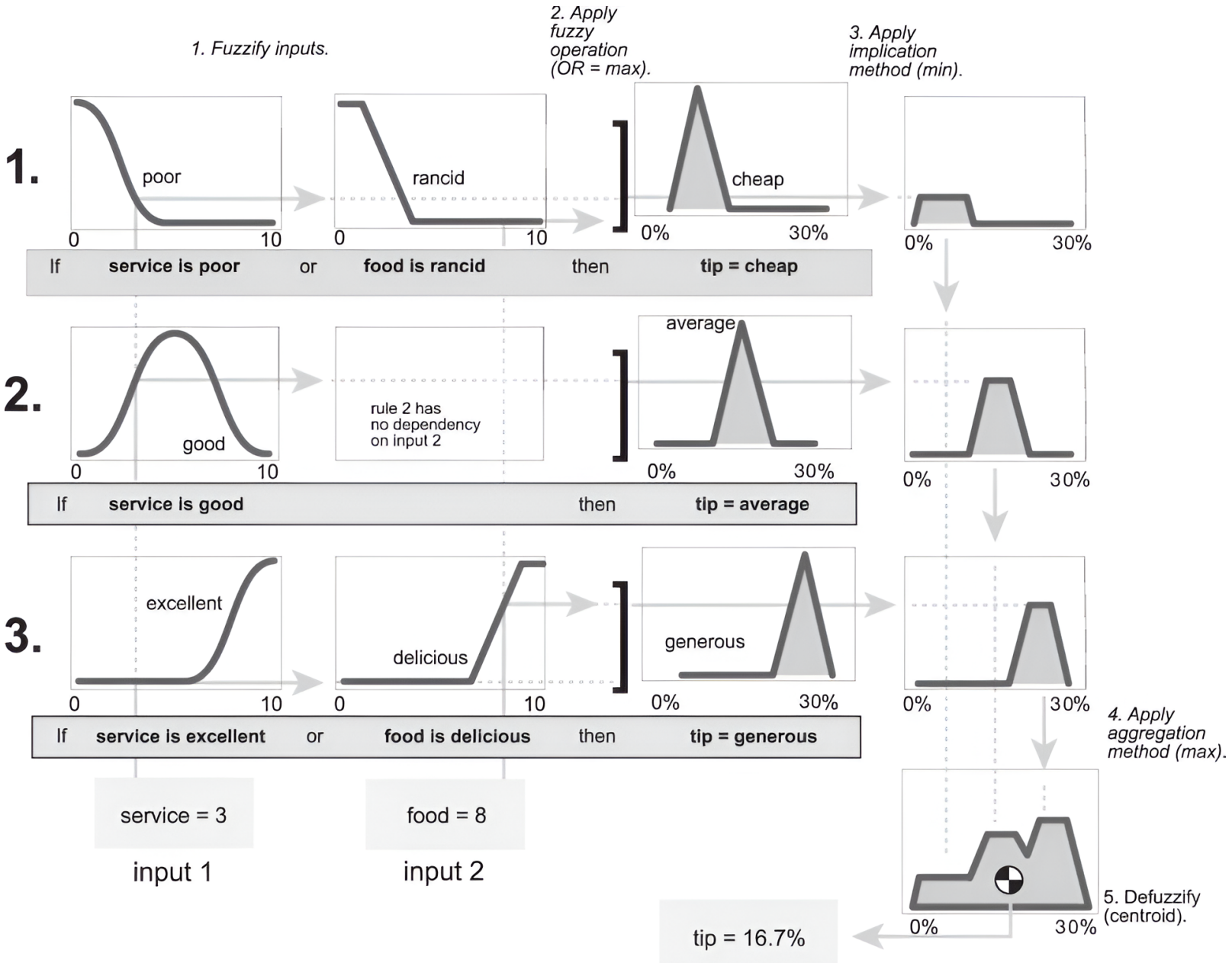
Input: temperature (e.g. 20°C), humidity (e.g. 60%)

Output: heater power (e.g. 50%)

Rules:

IF temp is *cold*	**AND**	humidity is *dry*	**THEN** power is *high*
IF temp is *hot*	**OR**	humidity is *wet*	**THEN** power is *low*
IF temp is *warm*			**THEN** power is *medium*

Source: MathWorks - Fuzzy Inference Process



Fuzzy Tuning for AutoPas

- Main Idea: Use Fuzzy Logic to tune AutoPas
- Expected Benefits:
  - Expert knowledge allows efficient tuning phases
    - ✓ Minimal tuning overhead
    - ✓ Allows frequent tuning phases
  - Comes with *interpolation* effect
    - ✓ Fewer rules than Rule-Based tuning
    - ✓ Generalization to new environments
  - Simplicity of the model
    - ✓ Easy to understand and interpret
    - ✓ Easy to maintain

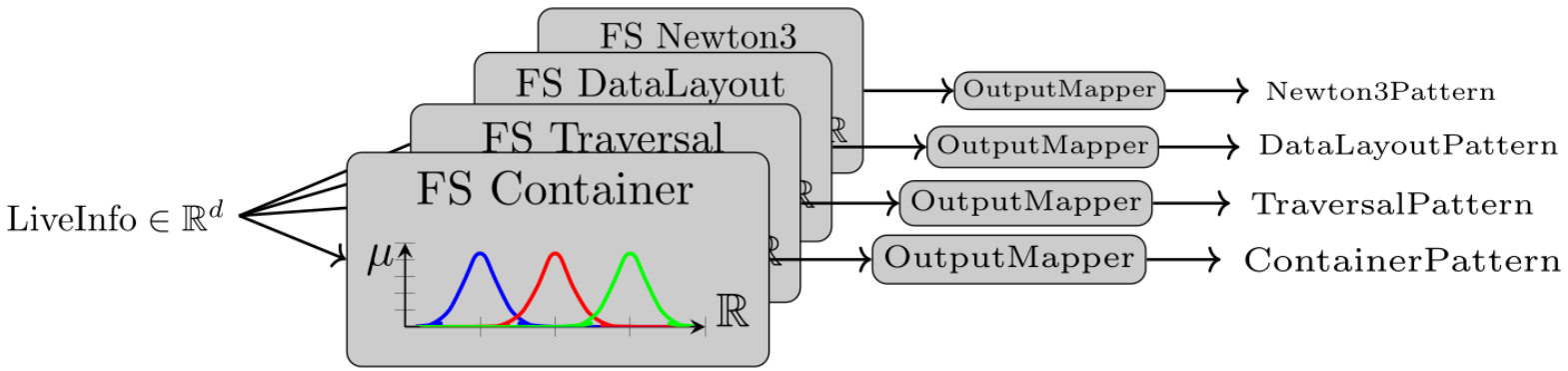
Challenges and Questions

- Fuzzy Tuning not directly applicable to AutoPas
  - Tuning is an *algorithm selection* problem
  - Fuzzy tuning provides functions $f : \mathbb{R}^n \to \mathbb{R}$
  - Big Question: How to interpret the numerical output?
    - $\to$ *Component Tuning* Approach
    - $\to$ *Suitability Tuning* Approach
- How to design the Fuzzy System?
  - Creating it manually?
    - ✗ Requires extensive expert knowledge
    - ✗ Difficult to formalize non-trivial knowledge
  - Extracting rules from data?
    - ✓ No prior expert knowledge required
    - ✓ Semi-automated process
    - ✓ Good starting point for human tuning

Approach 1: Component Tuning

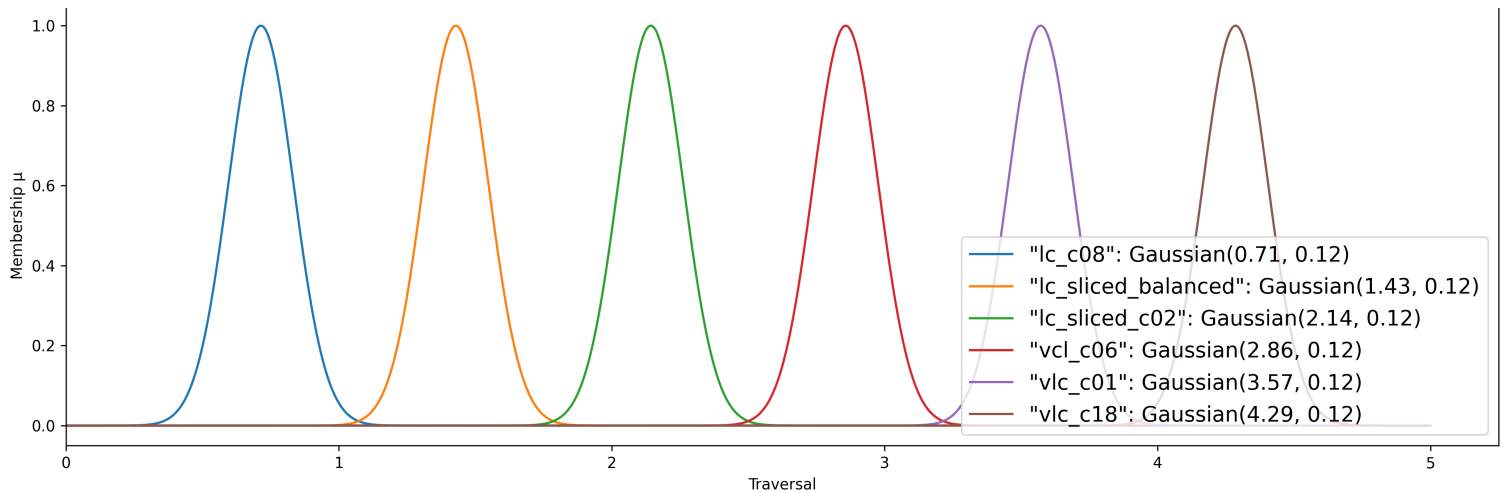
- Predict good values for each tunable parameter separately
- Explicit Fuzzy System for each tunable parameter
  - Output Variable: Continuous representation of the parameter
  - *Discrete* implementations are placed in the output space
  - Map numerical output to the *closest* value

    (Inspired by: [Mohammed et al., 2022])
- Combine individual predictions to form final configuration(s)



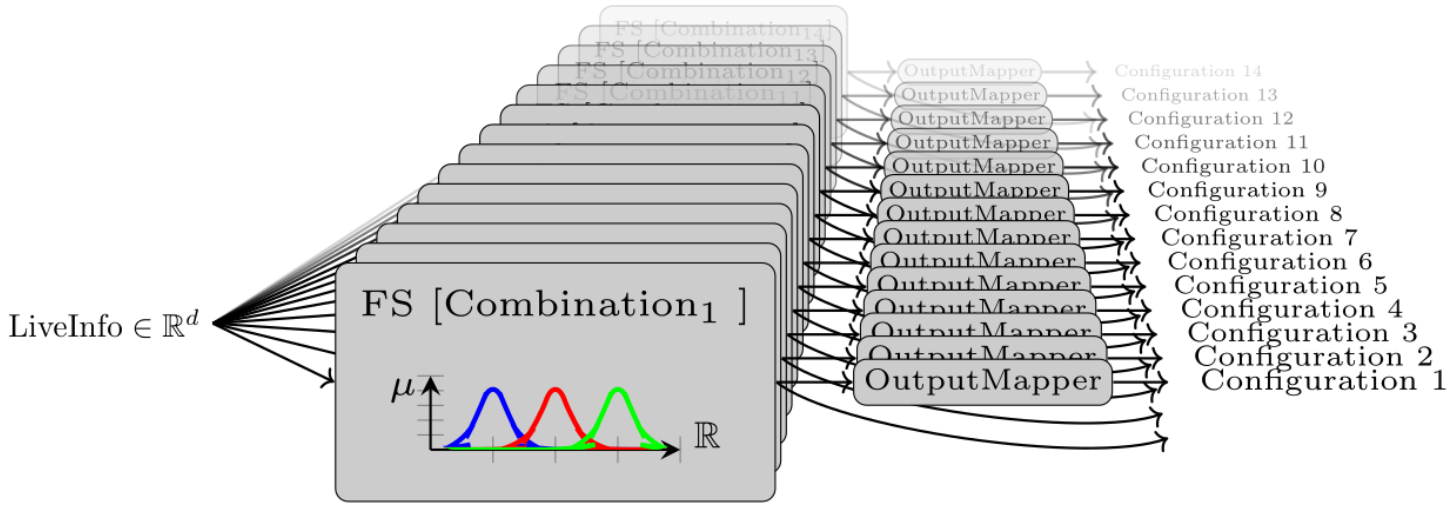
Approach 1: Component Tuning (Example)

- Style: **IF** avgParticlesPerCell is *low* **AND** threadCount is *low*
  THEN traversal is *vcl_c06*
- **Benefits:**
  - Few fuzzy systems
  - Few and natural rules
- **Drawbacks:**
  - Limited interpolation effect (MoM defuzzification)
  - Independence assumption



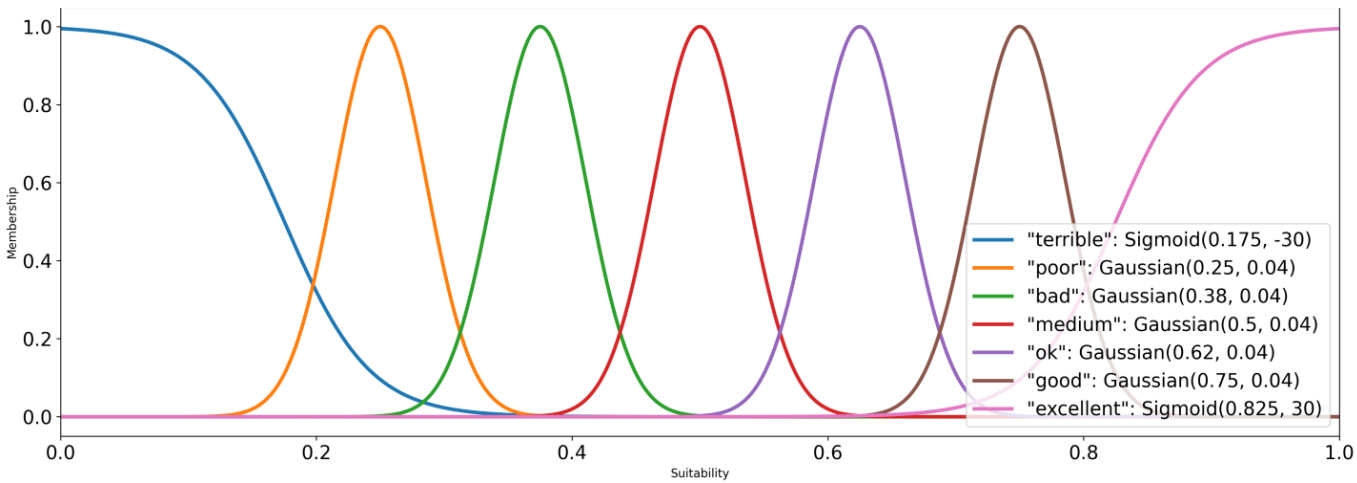
Approach 2: Suitability Tuning

- Predict the *suitability* of **each** configuration separately
- Explicit Fuzzy System for all possible configurations
  - Output Variable: Numerical suitability of the configuration
- Results in (*suitability*, *configuration*) pairs
  - *Best* configurations are selected



Approach 2: Suitability Tuning (Example)

- Style: **IF** threadCount is *high* **AND** avgParticlesPerCell is *low*
  THEN suitability_LinkedCells_AoS_lc_c18_disabled is *bad*
- **Benefits:**
  - Utilizes the full power of fuzzy logic (CoG defuzzification)
  - Dependencies and incompatibilities can be modeled
- **Drawbacks:**
  - Huge number of fuzzy systems
  - Impossible to maintain by hand

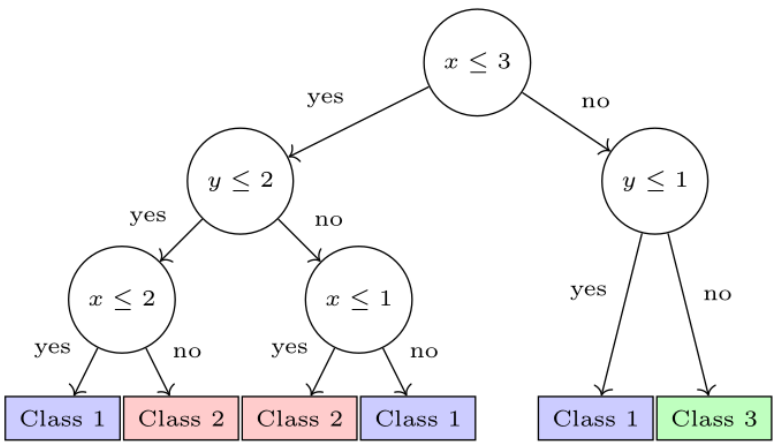


Data-Driven Rule Extraction

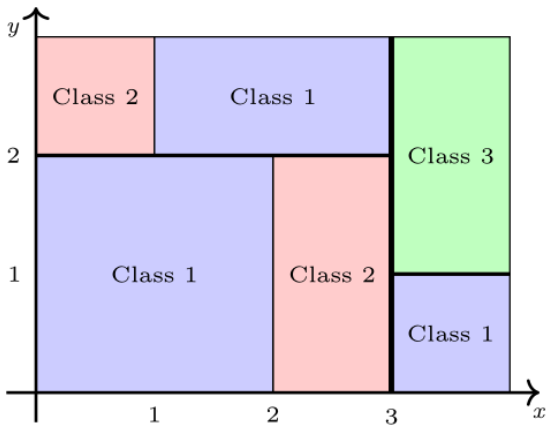
- Use data to automatically generate fuzzy systems
  - Utilize machine learning techniques to extract rules
  - We follow [Crockett et al., 2006]
    1. Train a Decision Tree on the data
    2. Decision Tree $\rightarrow$ Fuzzy Decision Tree
    3. Fuzzy Decision Tree $\rightarrow$ Fuzzy Rules
- Practical Considerations:
  - Ensemble Learning for Decision Trees
    - Train with selected features
    - Different splitting criteria
    - Different tree depths / pruning
  - *Overlapping* rules $\rightarrow$ *Interpolation* effect

Decision Trees

- Seperate classes recursively with axis-aligned splits
- Corresponds to nested *if-then-else* rules
  - Easy to understand and interpret
- Trained automatically from data
- Tree contains the entire expert knowledge



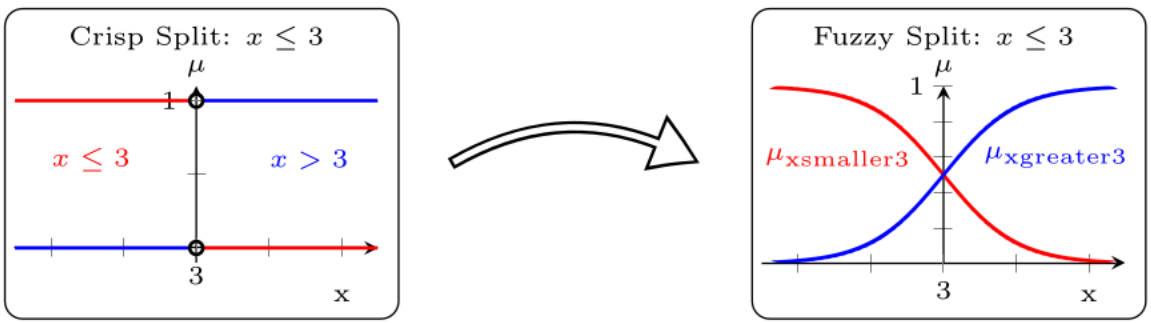
(a) Example decision tree

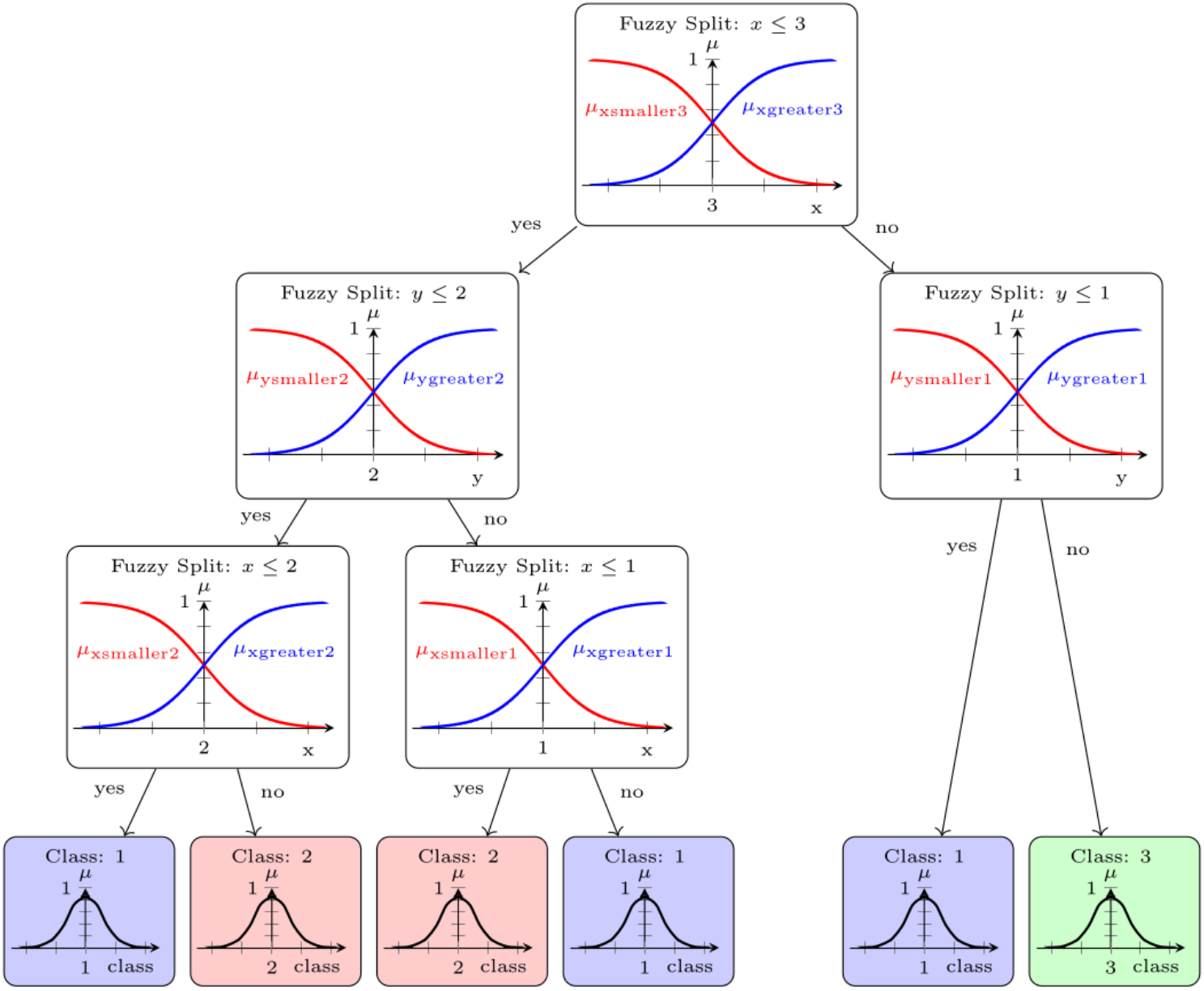


(b) Decision surface over $\mathcal{D} = [0, 4] \times [0, 3]$

Decision Trees → Fuzzy Decision Trees

- Conversion: Each (crisp) split is turned into two fuzzy sets
  - Fuzzy sets should maintain the semantics of the split
  - Provides robustness against noise
- Leaf nodes are turned into fuzzy sets
  - Each class corresponds to a fuzzy set




Fuzzy Split: $x \leq 3$
μ
1
$\mu_{\text{xsmaller3}}$
$\mu_{\text{xgreater3}}$
3
x
yes
no
Fuzzy Split: $y \leq 2$
$\mu_{\text{ysmaller2}}$
$\mu_{\text{ygreater2}}$
2
y
Fuzzy Split: $y \leq 1$
$\mu_{\text{ysmaller1}}$
$\mu_{\text{ygreater1}}$
1
y
Fuzzy Split: $x \leq 2$
$\mu_{\text{xsmaller2}}$
$\mu_{\text{xgreater2}}$
2
x
Fuzzy Split: $x \leq 1$
$\mu_{\text{xsmaller1}}$
$\mu_{\text{xgreater1}}$
1
x
Class: 1
1 class
Class: 2
2 class
Class: 2
2 class
Class: 1
1 class
Class: 1
1 class
Class: 3
3 class

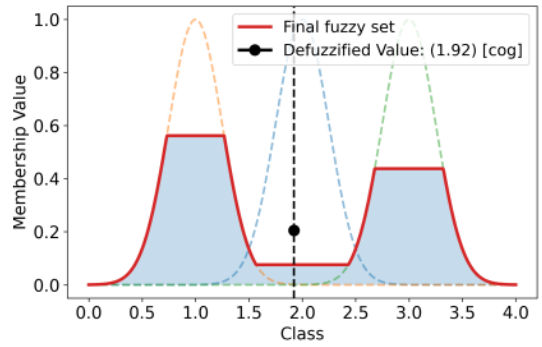
Fuzzy Decision Trees $\rightarrow$ Fuzzy Rules

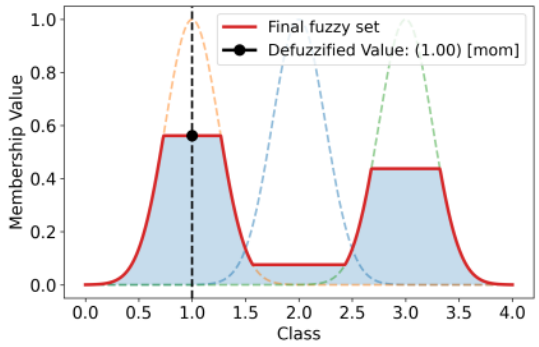
- Depth-First traversal of the Fuzzy Decision Tree
  - Fuzzy splits are collected in the antecedent
  - Leaf node corresponds to the consequent
- One rule per path from root to leaf
- Corresponds to *unnesting* the decision tree

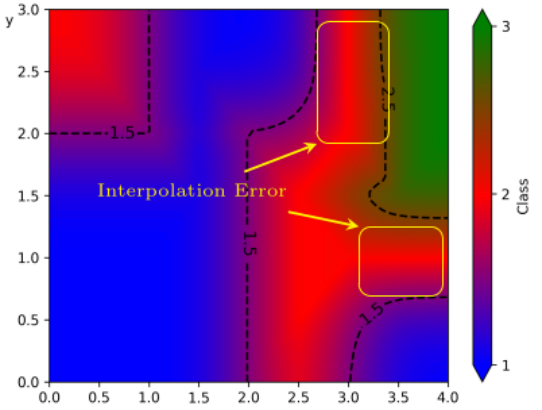
Rule	Antecedent	Consequent
1	$x\ is\ \text{smaller3} \wedge y\ is\ \text{smaller2} \wedge x\ is\ \text{smaller2}$	$class\ is\ 1$
2	$x\ is\ \text{smaller3} \wedge y\ is\ \text{smaller2} \wedge x\ is\ \text{greater2}$	$class\ is\ 2$
3	$x\ is\ \text{smaller3} \wedge y\ is\ \text{greater2} \wedge x\ is\ \text{smaller1}$	$class\ is\ 2$
4	$x\ is\ \text{smaller3} \wedge y\ is\ \text{greater2} \wedge x\ is\ \text{greater1}$	$class\ is\ 1$
5	$x\ is\ \text{greater3} \wedge y\ is\ \text{smaller1}$	$class\ is\ 1$
6	$x\ is\ \text{greater3} \wedge y\ is\ \text{greater1}$	$class\ is\ 3$

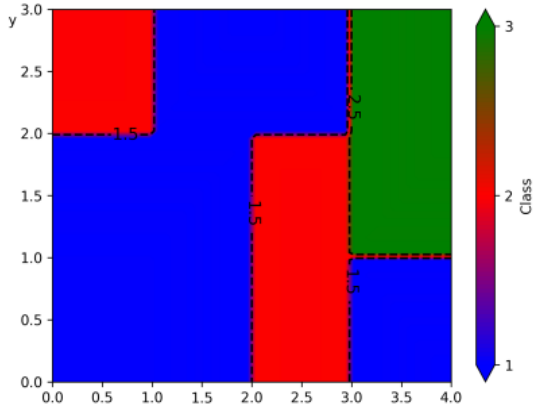
Decision Surfaces

- Defuzzification Method affects the decision surface
  - CoG: Interpolation effect, smooth boundaries
  - MoM: Most likely class, hard boundaries
- CoG: *Interpolation errors* for nominal values









Fuzzy Rule Extraction for `md_flexible`

- Collect dataset of `md_flexible` simulations (FullSearch)
- For each tuning phase *i*, store:
  - LiveInfoData: `maxDensity, homogeneity, threadCount, ...`
  - TuningData: `Container, Traversal, Newton3, ...,` **`Time`**
- Introduce *relative speed* metric
  - Absolute time is not meaningful across different *environments*

$$\text{relative speed}^{(i)}_{config} = \frac{t^{(i)}_{best}}{t^{(i)}_{\text{config}}}$$

Resulting Dataset

- Contains expected *relative speed* based on:
  - Simulation state
  - Configuration
- Forms basis for *Suitability* and *Component* tuning approaches

ParticlesPerCell			Miscellaneous			Configuration			
avg	max	stddev	homo-genity	max-density	threads	Container DataLayout	Traversal	Newton3	Relative Speed
0.905	23	0.0129	0.0354	0.531	1	LinkedCells_AoS	lc_sliced	enabled	0.450641
2.201	13	0.0144	0.0861	0.627	24	VerletListsCells_AoS	vlc_sliced	disabled	0.594117
0.905	18	0.0136	0.0431	0.319	4	LinkedCells_AoS	lc_sliced_c02	enabled	0.454632
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Component Tuning: Extracted Rules

- Preprocessing:
  1. Naively remove bad configurations (relative speed $< 70\%$)
  2. Group remaining dataset by tuning-phase
  3. Aggregate present (*good*) values for each parameter
- Apply Rule Extraction algorithm for each parameter

ParticlesPerCell			Miscellaneous			Aggregated Configuration Terms		
avg	max	stddev	homo-genity	max-density	threads	Container DataLayout	Traversal	Newton3
0.906	15	0.015	0.055	0.297	4	"LinkedCells_SoA, VerletClusterLists_SoA, VerletListsCells_AoS"	"lc_sliced, lc_sliced_balanced, lc_sliced_c02"	"enabled"
0.945	25	0.041	0.084	0.673	24	"LinkedCells_SoA, VerletClusterLists_SoA, VerletListsCells_AoS"	"lc_c04, lc_c08, lc_sliced, lc_sliced_balanced"	"disabled, enabled"

Antecedent				Consequent
avgParticlesPC	homogeneity	particlesPCStdDev	threadCount	Traversal
lower than 1.553	higher than 0.047	lower than 0.023	higher than 2.5	"lc_sliced, vlc_c18, lc_sliced_c02"
	lower than 0.037	lower than 0.023	lower than 26.0	"vcl_c06, vlc_c18, vlc_sliced_c02"
⋮	⋮	⋮	⋮	⋮

Suitability Tuning: Extracted Rules

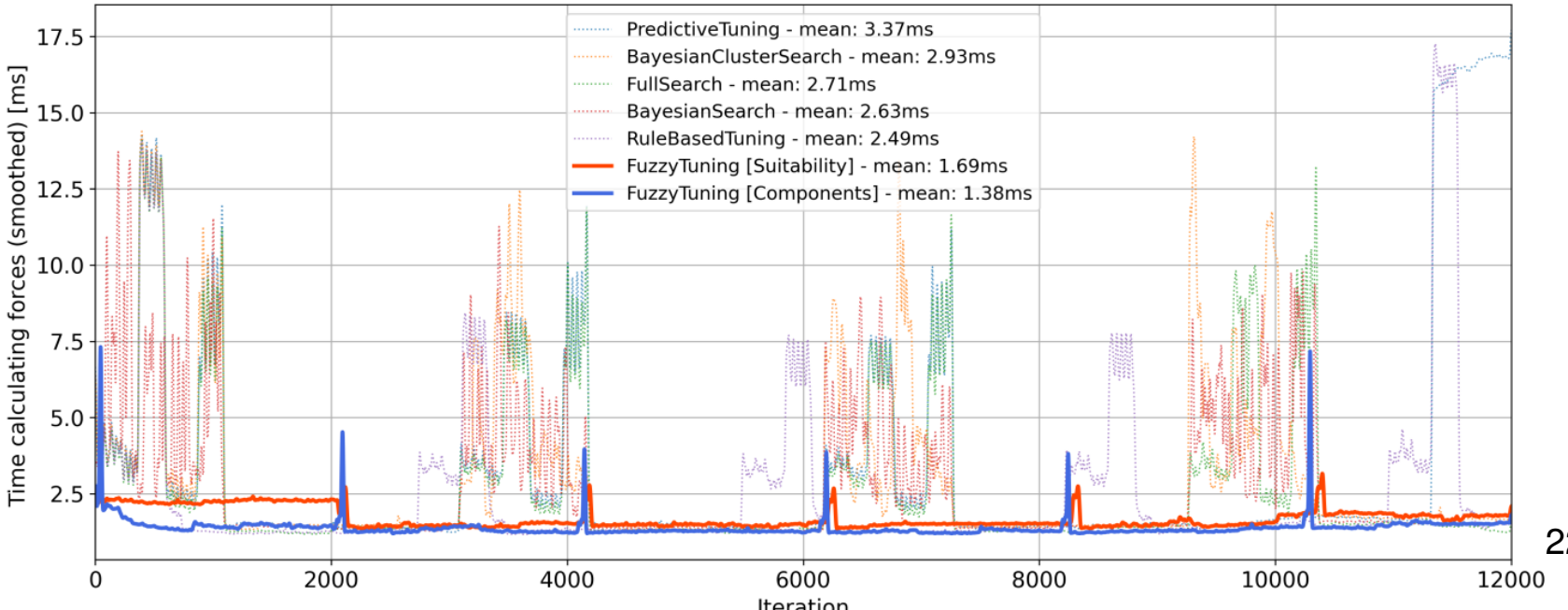
- Preprocessing:
  1. Assign suitability-classes to each entry in the dataset
- Extraction:
  1. Group dataset by configuration
  2. Apply Rule Extraction algorithm for each configuration

ParticlesPerCell			Miscellaneous			Configuration				
avg	max	stddev	homo-genity	max-density	threads	Container DataLayout	Traversal	Newton3	Relative speed	Suitability
0.905	15	0.012	0.035	0.531	1	LinkedCells_AoS	lc sliced	enabled	0.450	"bad"
0.944	25	0.012	0.083	0.691	28	VerletClusterLists_AoS	vcl_c06	disabled	0.319	"poor"
0.944	20	0.012	0.079	0.041	12	LinkedCell_SoA	vlc_ sliced	enabled	0.989	"excellent"

Antecedent				Consequent
avgParticlesPC	homogeneity	particlesPCStdDev	threadCount	Suitability LinkedCells_AoS lc_c01_disabled
	lower than 0.084	higher than 0.029	higher than 26.0	"medium"
	higher than 0.084	higher than 0.029	higher than 26.0	"bad"
		higher than 0.02	lower than 2.5	"poor"
⋮	⋮	⋮	⋮	⋮

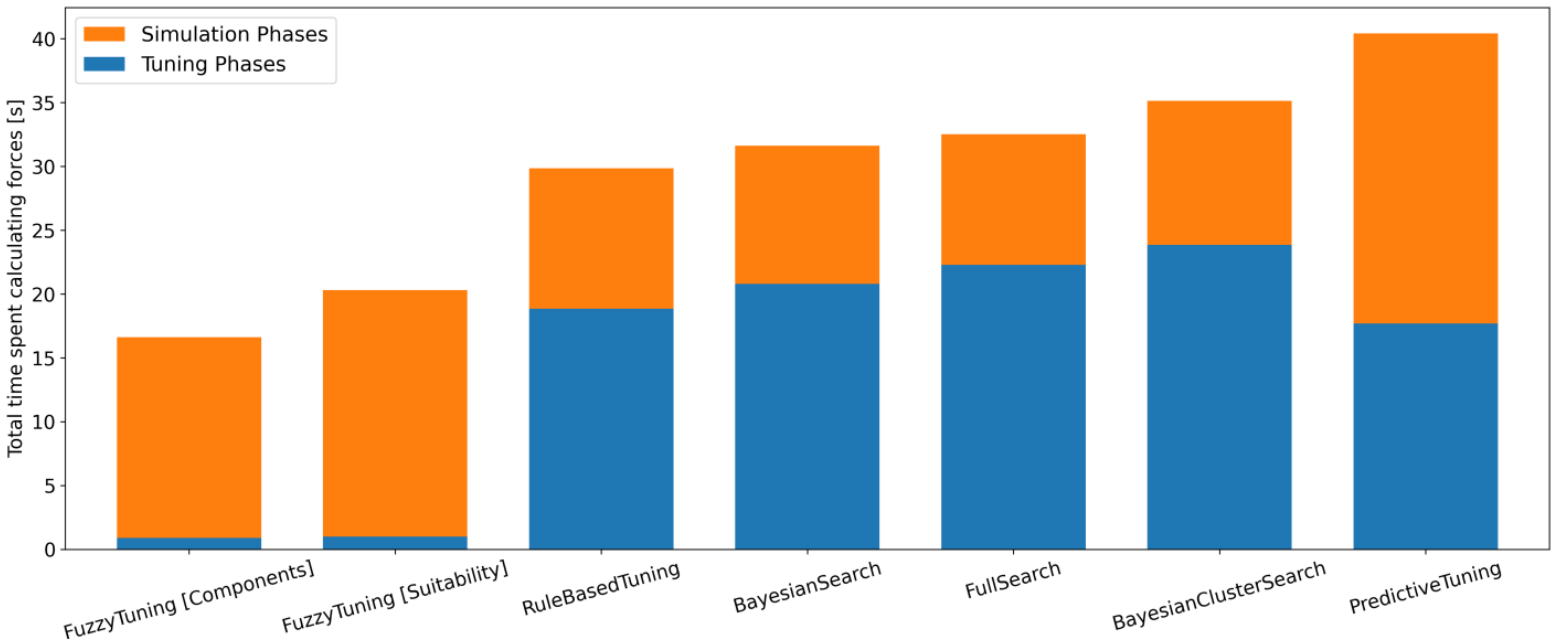
Benchmark 1: Exploding Liquid

- Exploding Liquid Benchmark (Included in Dataset)
  - Both fuzzy approaches look promising
  - Very short tuning phases (not much noise)
  - Selected configurations perform well (tiny spikes)
  - Winning configurations are (mostly) equivalent



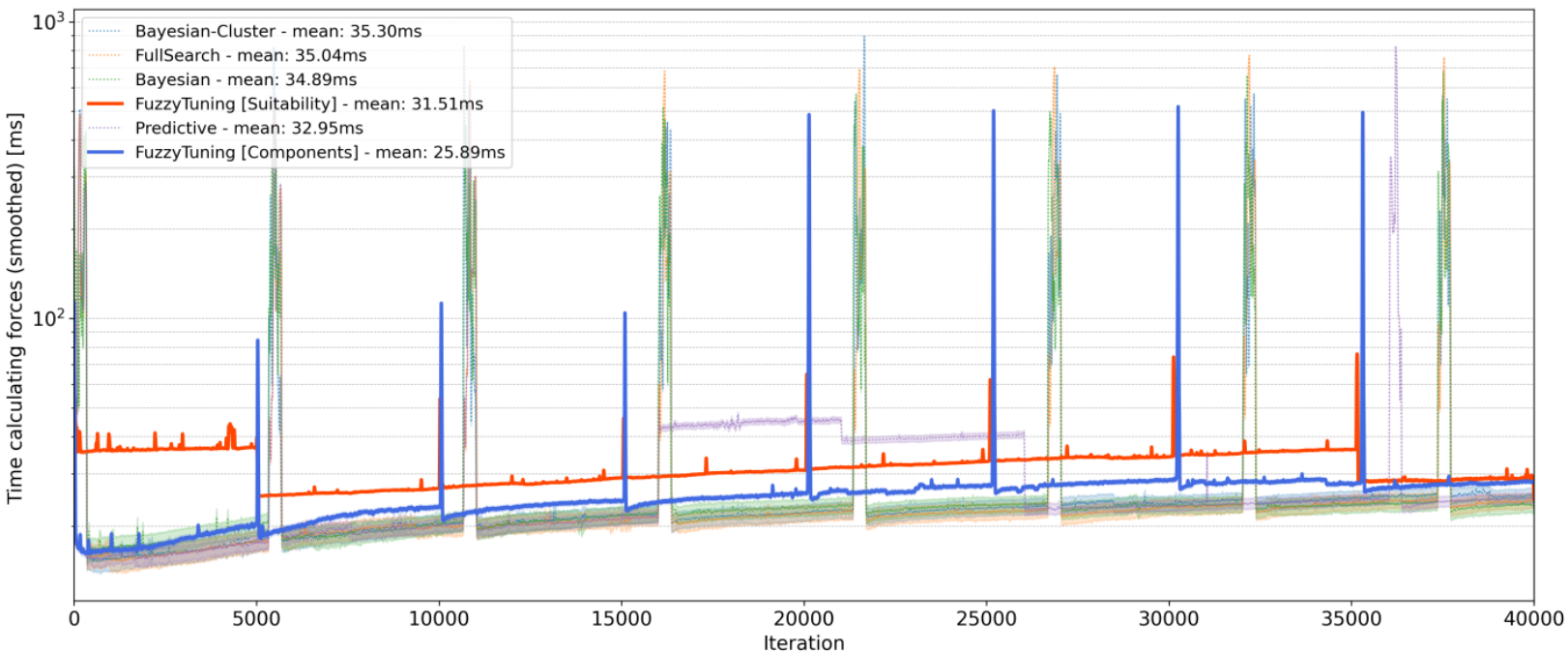
Total Time: Exploding Liquid

- Fuzzy tuning has lowest total time
- Mostly due to the very efficient tuning phases:
  - Few configurations evaluated
  - Evaluated configurations are expected to perform well
- Benefits of tuning with tiny tuning overhead



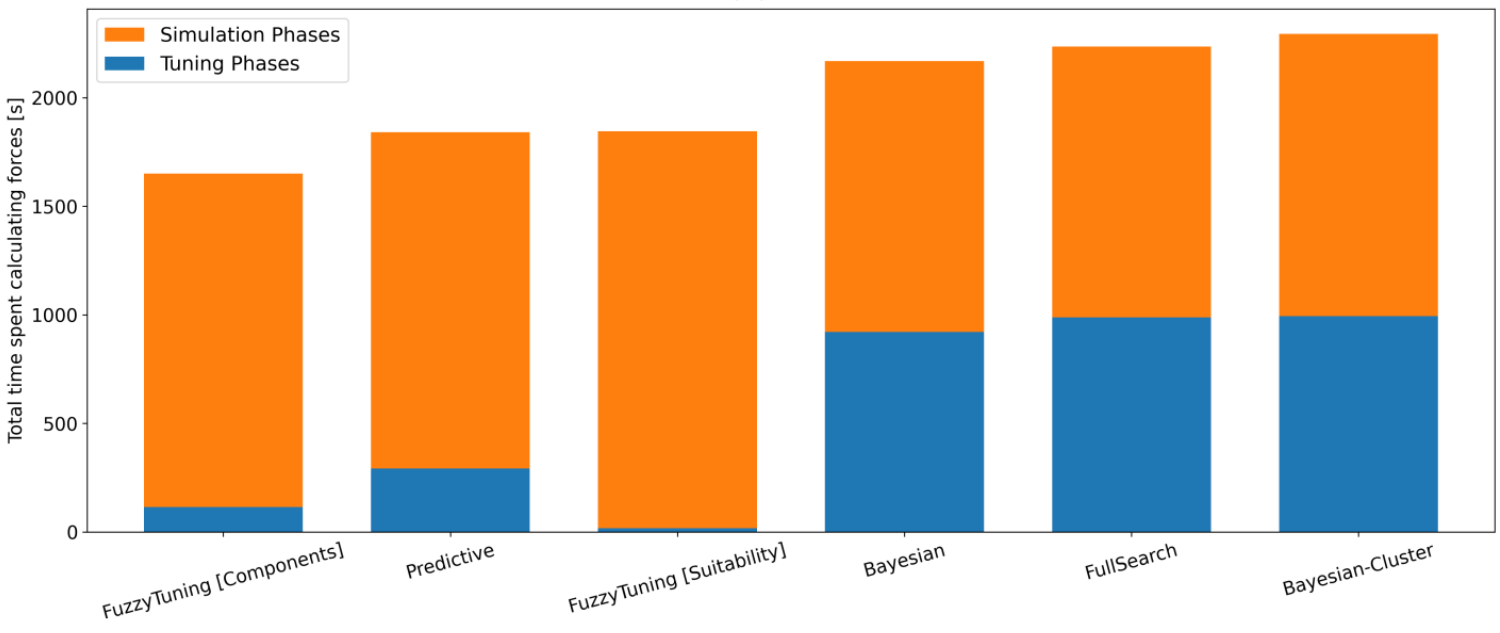
Benchmark 2: Spinodal Decomposition MPI

- Spinodal Decomposition MPI (Indirectly included in Dataset)
  - Component approach performs well
  - Suitability approach struggles
    - However, very fast tuning phases
    - Can this make up for the suboptimal configurations?



Comparison and Evaluation: Spinodal Decomposition

- Similar trends as previously
- Fast tuning phases compensate for suboptimal configurations!
- Improvements to the suitability approach:
  - Increase suitability threshold → encourage longer tuning phases
  - Optimal: Evaluating Top 30% (instead of 10%) of configurations



Conclusion

- Fuzzy Logic is very promising for tuning AutoPas
  - Massive reduction in tuning overhead
  - Still competitive results
- Data-Driven Rule Extraction is a good starting point
  - **Benefits:**
    - ✓ No need for expert knowledge
    - ✓ Semi-automated process
    - ✓ Generalizes to new applications
    - ✓ Interpretabilty is a huge benefit
  - **Drawbacks:**
    - ✗ Requires a lot of data (1.1GB)
    - ✗ Unapealing for users
    - ✗ Maybe too complex for simple problems

Future Work

- Dynamic Rule Generation
  - Update expert knowledge on the fly
  - No need for giant datasets
  - Could automatically adapt to new scenarios / applications
- General Improvements to the Tuning Process
  - Investigate early stopping mechanism
  - Stop evaluating extremely bad configurations early
  - Maybe solve the tuning overhead once and for all
- Simplification of the Model
  - Decision Trees instead of Fuzzy Systems
  - Easier to maintain and understand
  - Maybe comparable results

Thank you for your attention!

Questions?

References I

Crockett, K., Bandar, Z., Mclean, D., and OShea, J. (2006).

On constructing a fuzzy inference framework using crisp decision trees.

Fuzzy Sets and Systems, 157(21):2809–2832.

Mohammed, A., KorndÃ¶rfer, J. H. M., Eleliemy, A., and Ciorba, F. M. (2022).

Automated scheduling algorithm selection and chunk parameter calculation in openmp.

IEEE Transactions on Parallel and Distributed Systems, 33(12):4383–4394.

Newcome, S. J., Gratl, F. A., Muehlhaeusser, M., Neumann, P., and Bungartz, H.-J. (2024).

Autopas: Dynamic algorithm selection in molecular dynamics for optimal time and energy.

In *SIAM Conference on Parallel Processing (PP24)*. SIAM.

References II


Newcome, S. J., Gratl, F. A., Neumann, P., and Bungartz, H.-J. (2023).
Towards the smarter tuning of molecular dynamics simulations.
Amsterdam, The Netherlands.