Chair of Scientific Computing
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. (hons)
Samuel James Newcome, M.Sc.

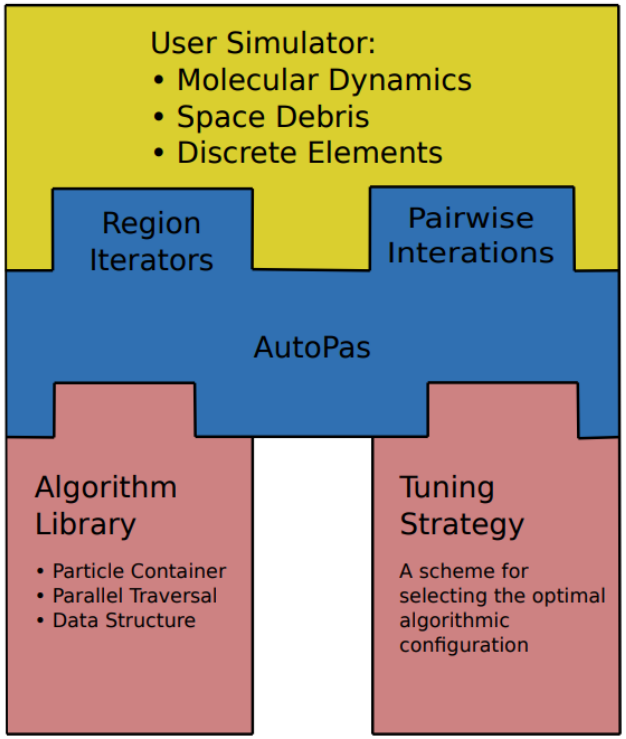
Table of Contents

What is AutoPas?

- Library for optimal node-level performance in N-body simulations
- Many different implementations for the N-body problem
- AutoTuning: Automatically switch between implementations
  - **Container:** How to find neighboring particles?
  - **Traversal:** How to handle multi-threading?
  - **Data Layout:** How to store particles in memory?
  - **Newton 3:** Can we exploit Newton's 3rd law?
  - ...
- Official Example applications:
  - `md_flexible` (Molecular Dynamics)
  - `sph` (Smoothed Particle Hydrodynamics)

Structure of AutoPas

- Three main components:
  - 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

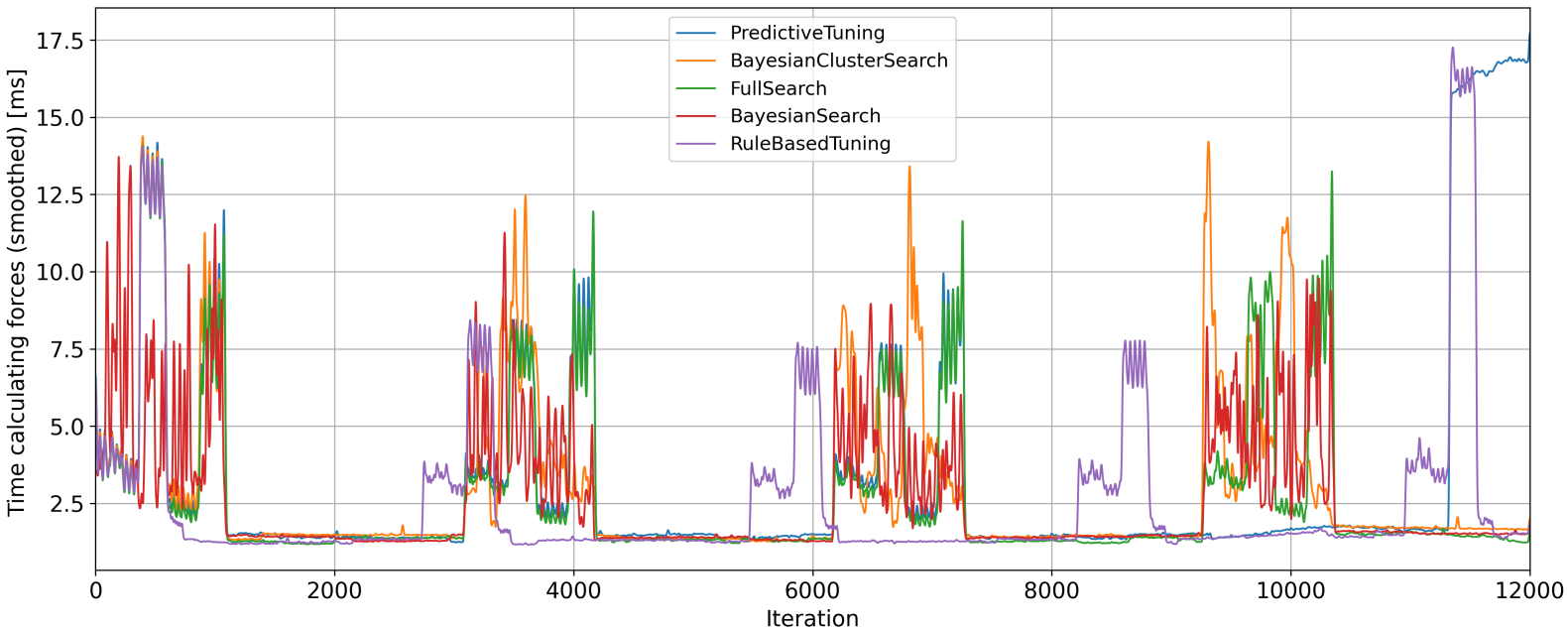


Source: [3]

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

Auto-Tuning

- Tuning Phase: Find the best configuration
  - Tuning Strategies select configurations to evaluate
  - Expensive, Time consuming
- Simulation Phase: Use the best configuration



Fuzzy Logic Systems

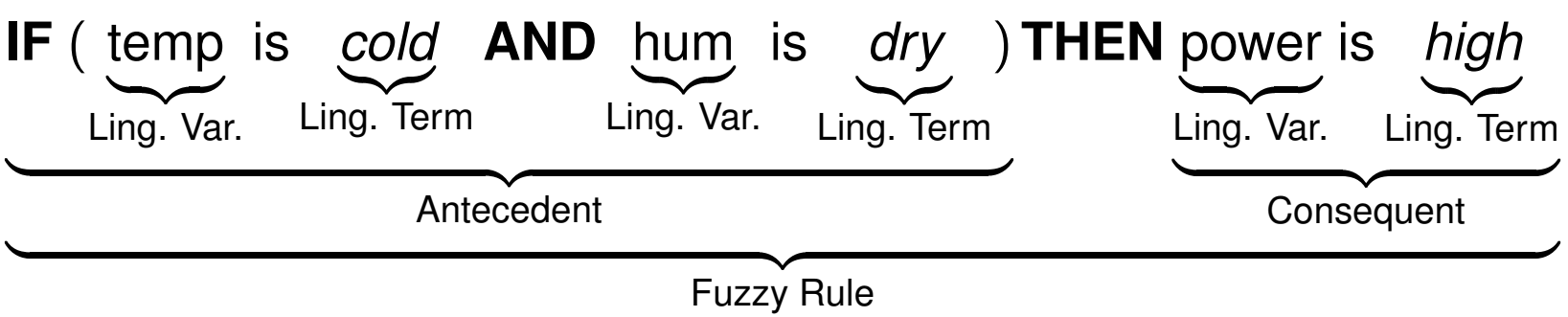
- Use human-like reasoning to model complex systems
- Example: Heater Control
  - Input: temperature (e.g. 20°C), humidity (e.g. 50%)
  - Output: heater power (e.g. 50%)
  - Rules:

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*

- Very easy to understand and interpret
- Can handle uncertainty and imprecise information
- Complexity is abstracted away in the linguistic terms (e.g. *cold, warm, hot*)
- Fuzzy System $f : \mathbb{R}^n \to \mathbb{R}$

Mathematical Foundations

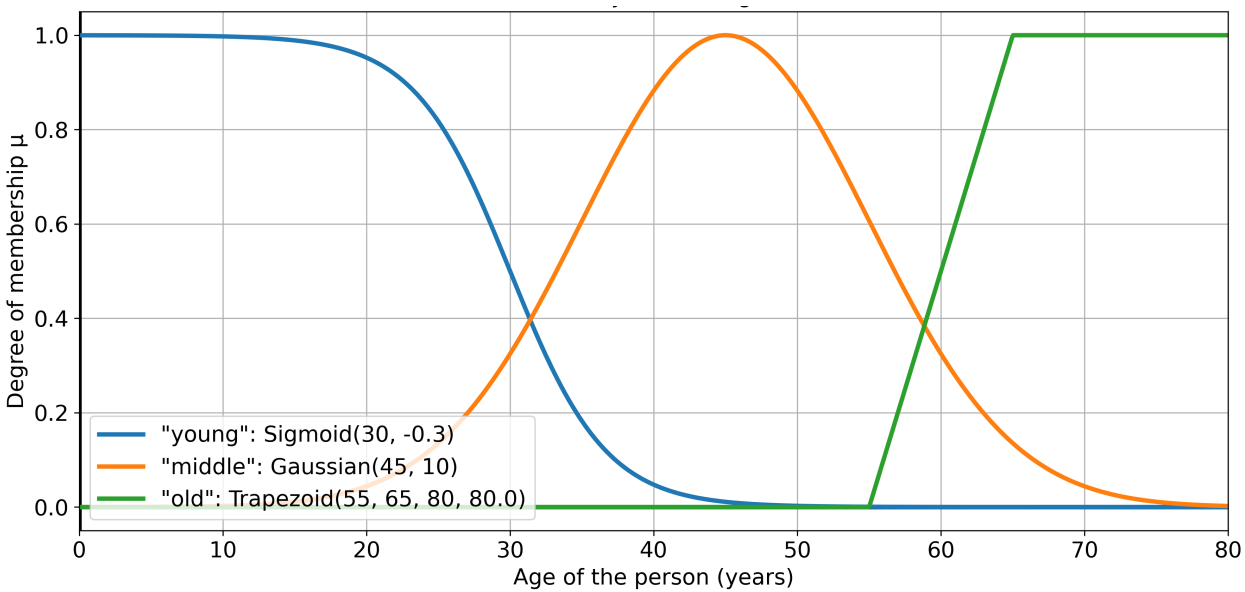
- Consider the Fuzzy Rule:



- Fuzzy Logic Systems consist of:
  - Linguistic Terms / Fuzzy Sets (e.g. *cold, warm, hot*)
  - Linguistic Variables (e.g. temperature, humidity, power)
  - Fuzzy Logic Operators (e.g. **AND, OR, NOT**)
  - Fuzzy Rules (e.g. **IF** *antecedent* **THEN** *consequent*)

Fuzzy Sets

- Fuzzy Sets are generalizations of classical sets
  - Classical Sets: binary membership function $\in_A$: $A \to \{false, true\}$
- Fuzzy Sets are defined by:
  - Underlying Crisp Set X (e.g. $age \subseteq \mathbb{R}$)
  - **Continuous** membership function $\mu_{\tilde{A}} : X \to [0, 1]$
- Allow for uncertainty. When is a person young?



Linguistic Variables

- Linguistic Variables can take on linguistic terms / fuzzy sets
  - E.g. *age* can take *young*, *middle-aged* or *old*
- Instead of using crisp values (35 years), we use a combination of linguistic terms to describe the age (Fuzzyification):

$$35 \text{ years} \implies \begin{cases} 20\% \text{ young} \\ 60\% \text{ middle-aged} \\ 0\% \text{ old} \end{cases}$$

- This allows for non-numerical reasoning:
  - E.g. **IF** age is *young* **THEN** fitness is *high*
  - Use abstract concepts instead of crisp values
- Each linguistic term represents a certain *collection* of values

Fuzzy Logic Operators

- Fuzzy Logic Operators are used to modify/combine fuzzy sets
- Extension of boolean logic operators to real numbers
  - $\wedge : \{false, true\} \times \{false, true\} \rightarrow \{false, true\}$
  - **AND** : $[0, 1] \times [0, 1] \rightarrow [0, 1]$
- Extended operators need to maintain the classical semantics
- Typically, Fuzzy Logic Operators are defined as:
  - **AND:** Corresponds to the intersection of fuzzy sets

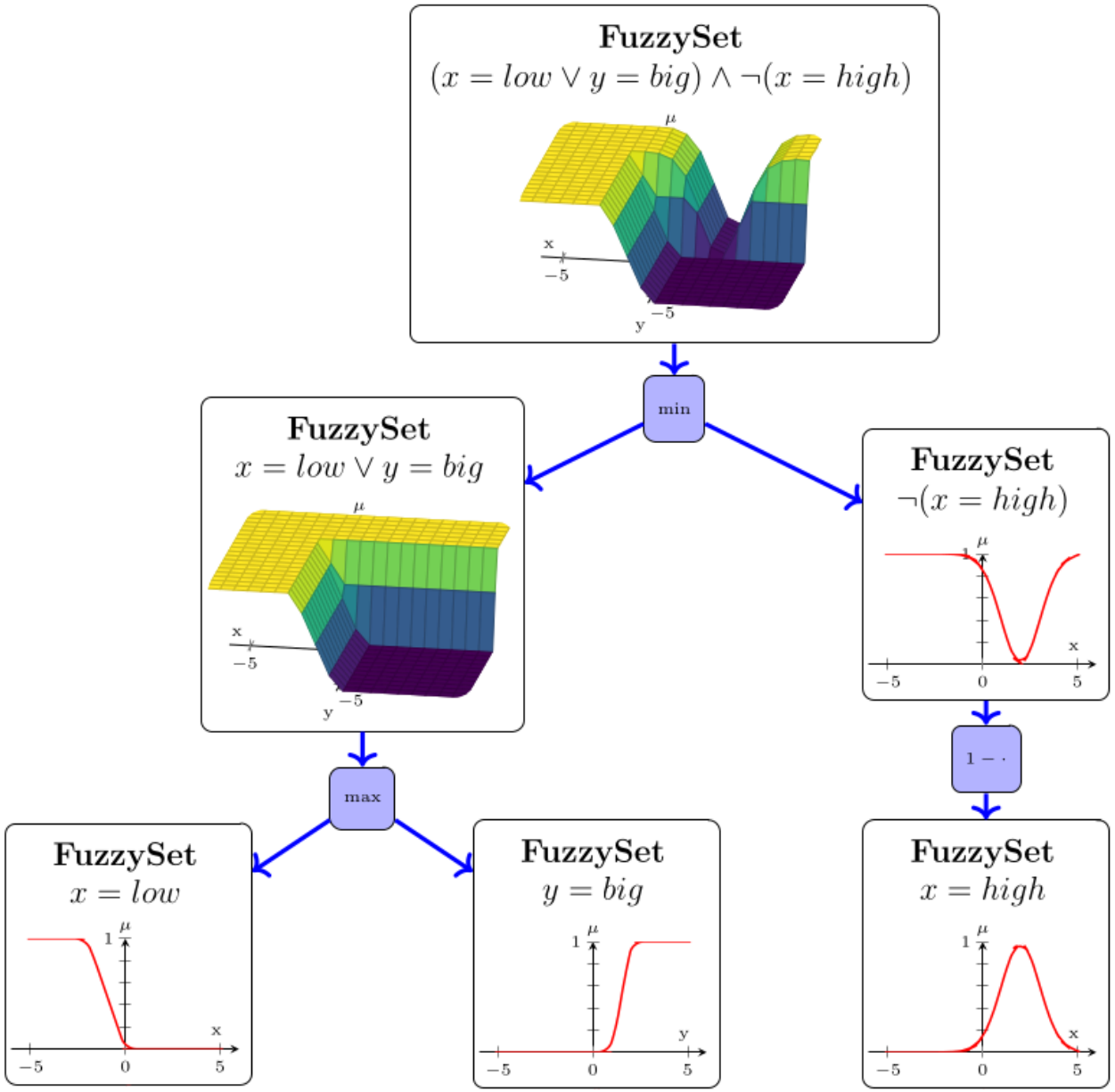
$$\mu_{\tilde{A} \cap \tilde{B}}(x) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

  - **OR:** Corresponds to the union of fuzzy sets

$$\mu_{\tilde{A} \cup \tilde{B}}(x) = \max(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

  - **NOT:** Corresponds to the complement of a fuzzy set

$$\mu_{\neg \tilde{A}}(x) = 1 - \mu_{\tilde{A}}(x)$$


FuzzySet
$(x = low \vee y = big) \wedge \neg(x = high)$
min
FuzzySet
$x = low \vee y = big$
FuzzySet
$\neg(x = high)$
max
1 − ·
FuzzySet
$x = low$
FuzzySet
$y = big$
FuzzySet
$x = high$

Fuzzy Rules

- Each rule is of the form: **IF** *antecedent* **THEN** *consequent*
  - Both antecedent and consequent are fuzzy sets
  - E.g. **IF** age is *young* **THEN** fitness is *high*
- Rules can be interpreted as a logical implication
  - **IF** $\tilde{A}$ **THEN** $\tilde{B}$ $\iff$ $\tilde{A}$ **IMPLIES** $\tilde{B}$
  - Implication is similar to previous operators (**AND, OR, NOT**)
  - Special form of implication: Mamdani Implication
    - $\tilde{R} =$ **IF** $\tilde{A}$ **THEN** $\tilde{B}$
    - $\mu_{\tilde{R}}(x) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$
  - *Effect* of the rule is limited by the *strength* of the antecedent
- A Fuzzy System can consist of multiple rules acting on the same linguistic variable
  - The total *effect* on the output is the combination/union of all individual rule outputs

Defuzzification

- Process of converting arbitrary fuzzy sets to a crisp value
  - Special case: Fuzzy sets resulting from rule application

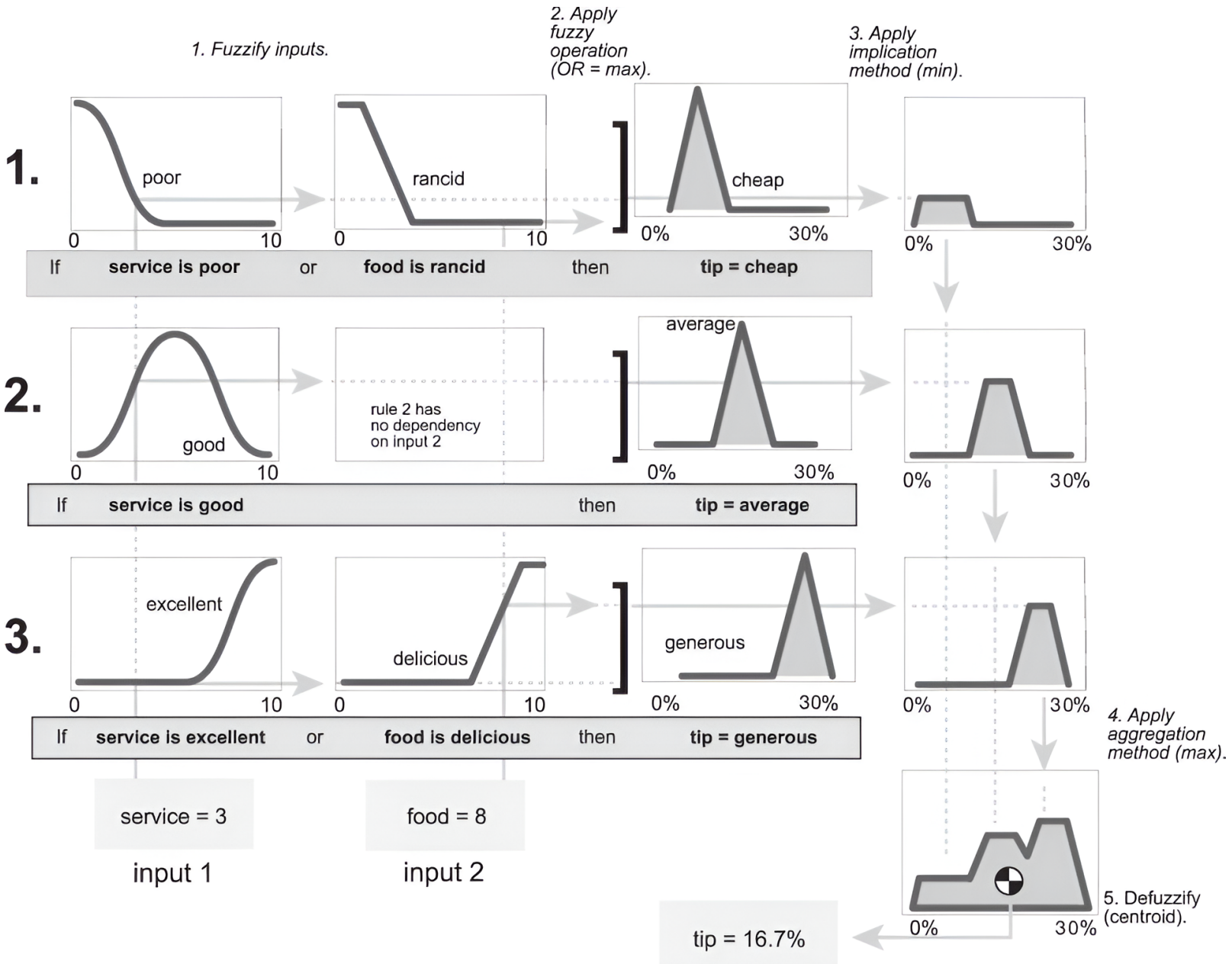
$$\begin{cases} 20\% \text{ young} \\ 60\% \text{ middle-aged} \\ 0\% \text{ old} \end{cases} \Longrightarrow 35 \text{ years}$$

- Multiple methods for defuzzification:
  - **Centroid:** Weighted average of the fuzzy set

$$\text{Centroid} = \frac{\int x \cdot \mu_{\tilde{R}}(x) dx}{\int \mu_{\tilde{R}}(x) dx}$$

  - **Mean of Maxima:** Average of all maxima of the fuzzy set
- Core idea: Represent aspects of the fuzzy set with a crisp value
  - E.g. Weighted average of all possible values (Centroid)
  - E.g. Most likely value (Mean of Maxima)

Source: MathWorks - Fuzzy Inference Process



- TODO: Maybe add decision surfaces here

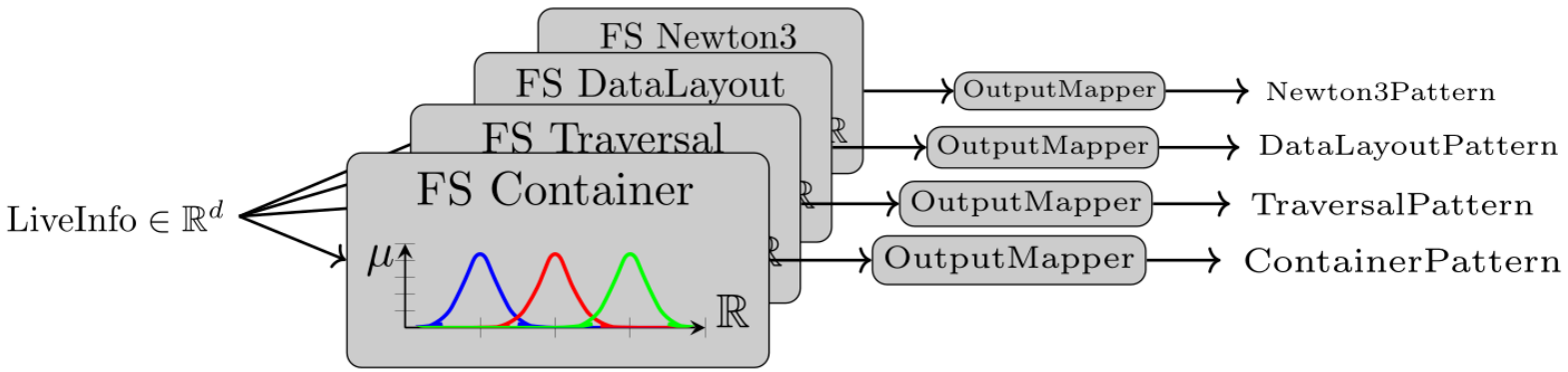
Fuzzy Tuning Strategy

- Main Idea: Use Fuzzy Logic to tune AutoPas
- Make use of LiveInfoData[2] to perform tuning
- Benefits:
  - Similar to Rule-Based Tuning
  - Potentially more expressive and powerful
  - Still easy to understand and interpret
- Challenges and Questions for AutoPas:
  - What are the output variables? How to predict Configurations?
  - How to interpret the result? (Fuzzy System : $f : \mathbb{R}^n \rightarrow \mathbb{R}$)
  - How to create the fuzzy rules? Expert knowledge?
  - How to specify the linguistic terms / fuzzy sets?

[2]Simulation state: `avgParticles/Cell, homogeneity, threadCount ...`

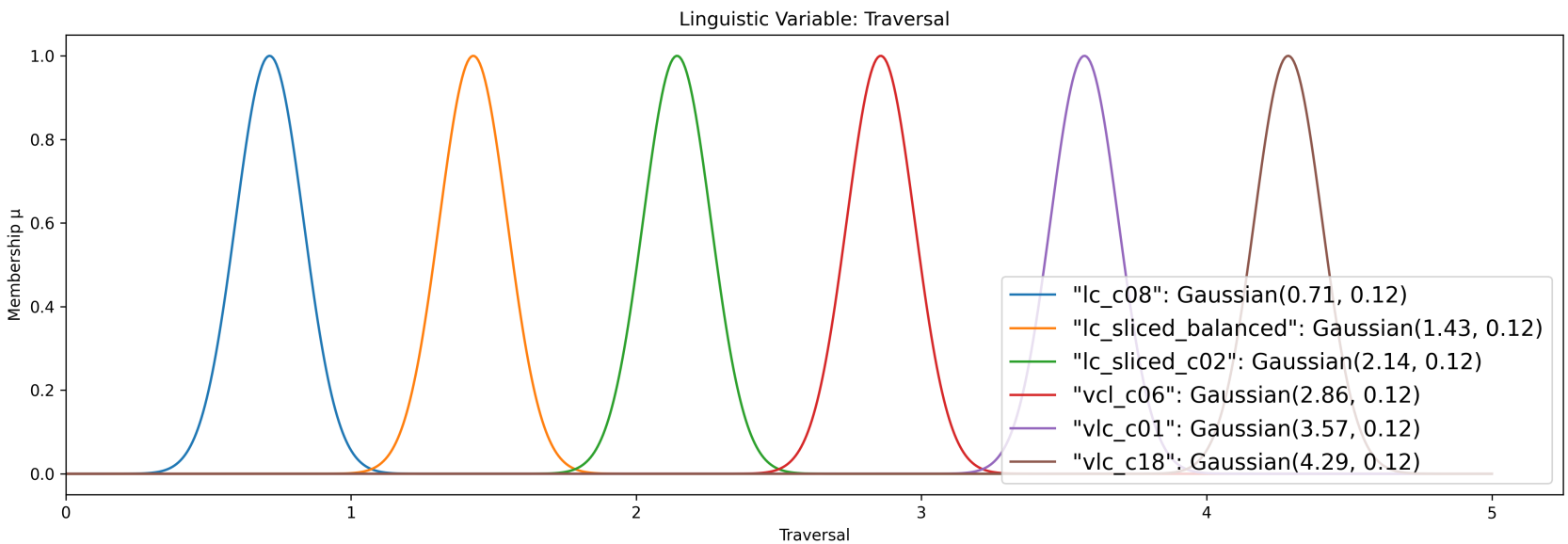
Approach 1: Component Tuning

- Independently predict good values/patterns for each tunable parameter
- Create a Fuzzy System for each tunable parameter
  - Container_DataLayout
  - Traversal
  - Newton 3
- Output Variables: Nominal Representation of parameter values
- Numeric output mapped to the closest value [2]
- Combine the patterns to obtain final configurations



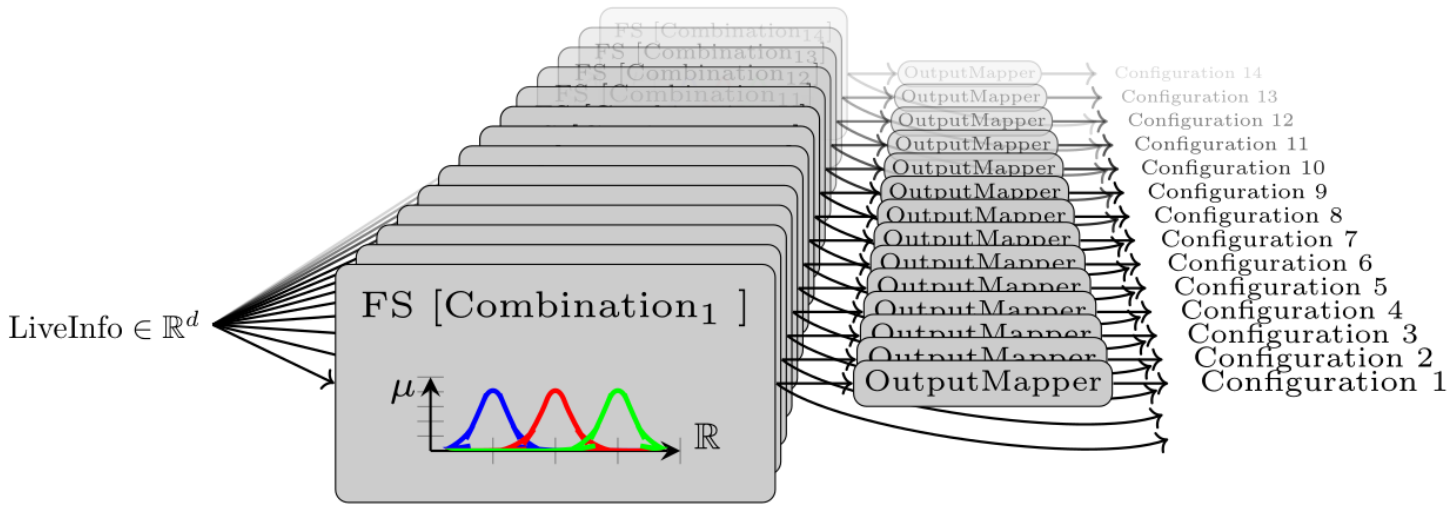
Approach 1: Component Tuning

- Each parameter has a separate Fuzzy System
- Mean of Maxima defuzzification $\Longrightarrow$ no interpolation
- Example rule:
  - **IF** avgParticlesPerCell is *low* **AND** threadCount is *low*
    THEN Traversal is *vcl_c06*
  - Where *low, high, AoS* are appropriate linguistic terms



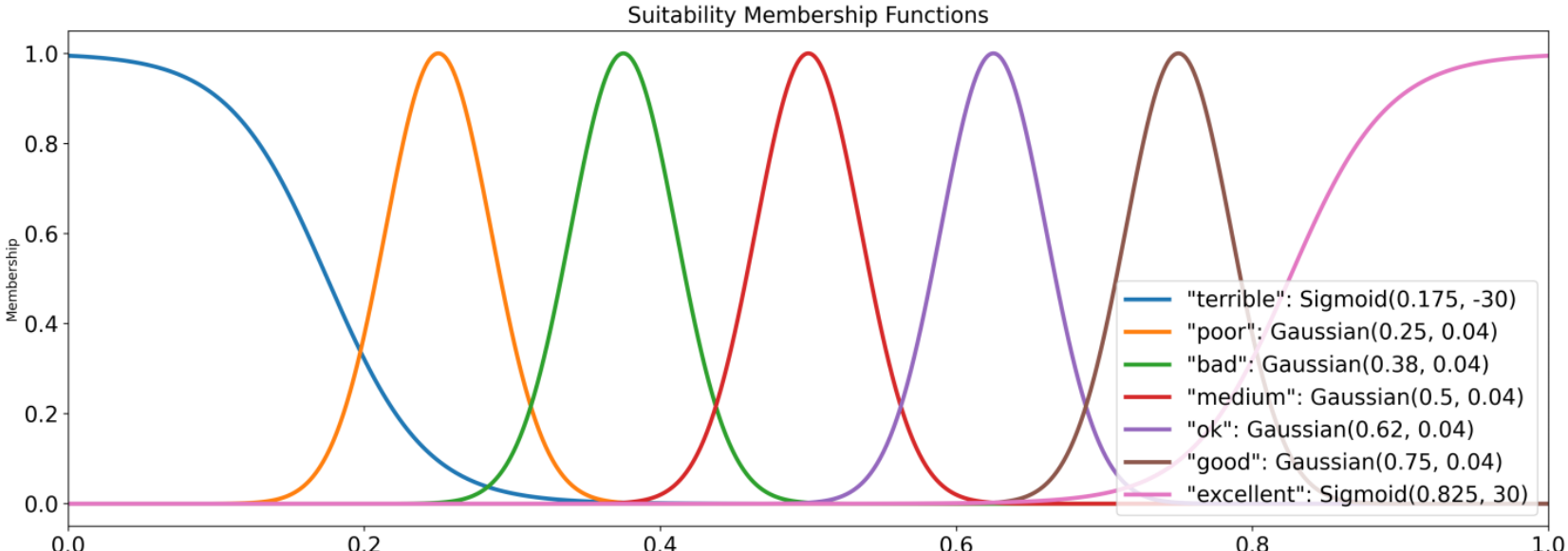
Approach 2: Suitability Tuning

- Predict the suitability of *each* configuration
- Use the suitability values to determine worthwhile configurations
- Create a Fuzzy System for each possible configuration
- Output Variables: Suitability of the configuration
- More complex, but potentially more powerful



Approach 2: Suitability Tuning

- Each configuration has a separate suitability variable
- Center of Gravity defuzzification
- Example rule:
  - **IF** threadCount is *high* **AND** avgParticlesPerCell is *low*
    THEN suitability_LinkedCells_AoS_lc_c18_disabled is *bad*
  - Where *high, low, bad* are appropriate linguistic terms

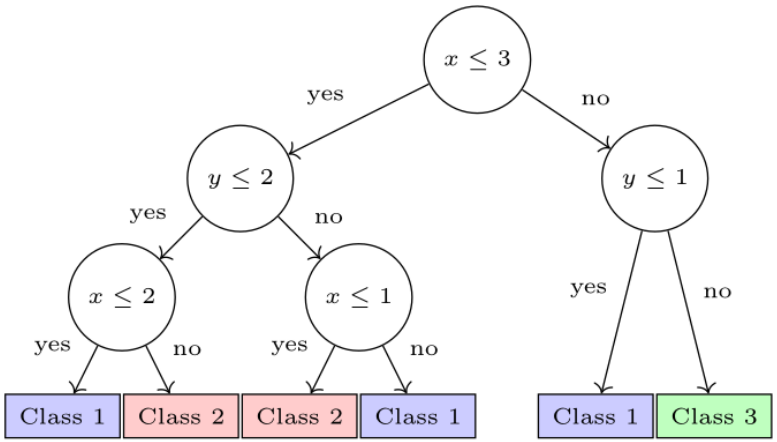


Data-Driven Rule Extraction

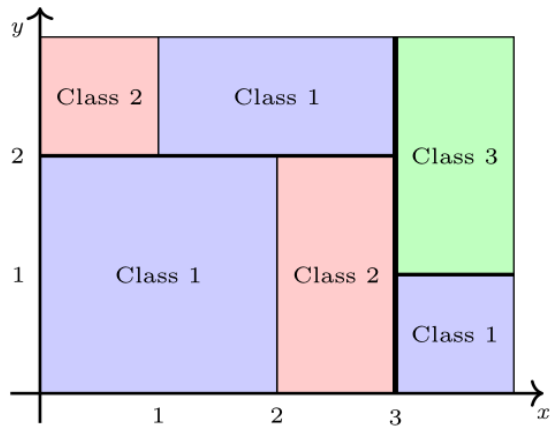
- Creating the Fuzy Rules is hard
  - Expert Knowledge is required
  - Formalization of knowledge is difficult
  - Potentially many rules required
- Use Machine Learning to extract rules from data
- Decision Tree $\rightarrow$ Fuzzy Decision Tree $\rightarrow$ Fuzzy Rules [1]
- Does not require expert knowledge
- Human experts can still validate the rules

Conversion Process

- Decision Trees
  - Try to find the best axis-aligned splits to separate classes
  - Results in *if-then-else* rules
  - Easy to understand and interpret
  - Many libraries available
  - Data $\rightarrow$ Decision Tree (trivial)



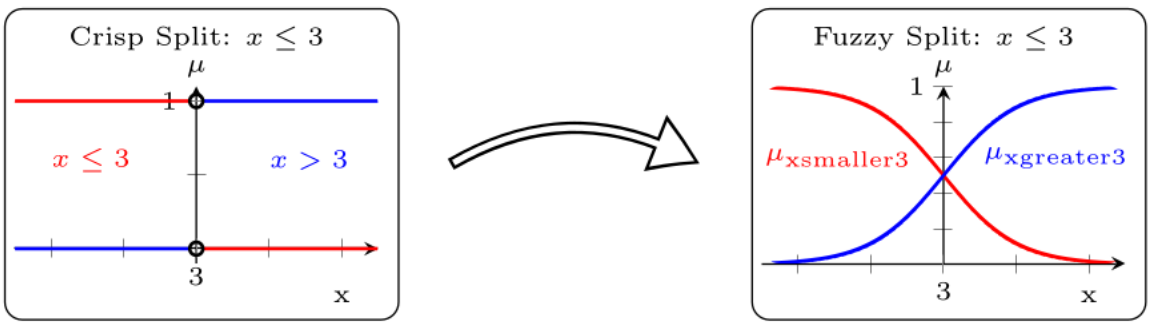
(a) Example decision tree

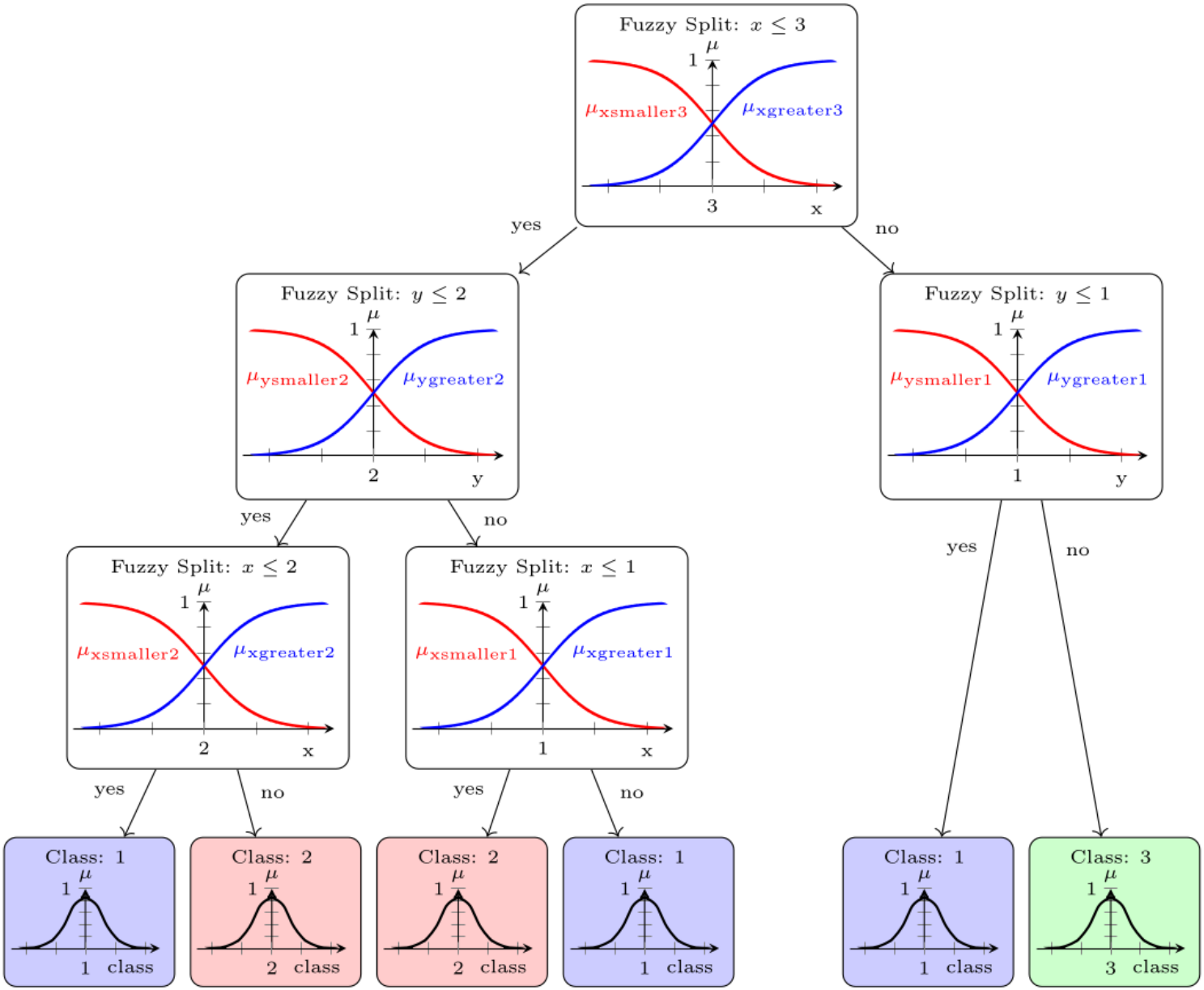


(b) Decision surface over $\mathcal{D} = [0, 4] \times [0, 3]$

Conversion Process

- Fuzzy Decision Trees
  - Convert each (crisp) split into two fuzzy sets
    - Fuzzy sets should maintain the semantics of the split
    - Introduce uncertainty in the split
    - Helpful for generalization, especially if value is close to the split
  - Convert leaf nodes into linguistic terms
- Create linguistic terms for all Leaf Nodes
  - Each class represented by a linguistic term
  - Collect them into a Linguistic Variable




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

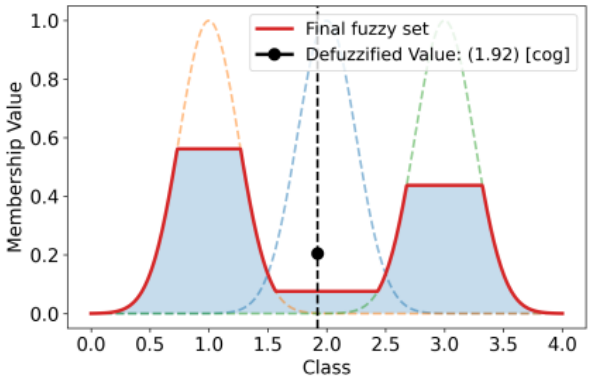
Fuzzy Rule Extraction

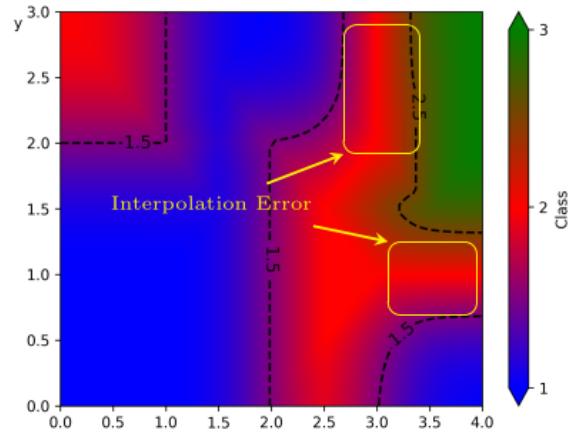
- Depth-First traversal of the Fuzzy Decision Tree
  - Each left-right decision corresponds to a condition
  - Leaf node corresponds to the output
- One rule per path from root to leaf

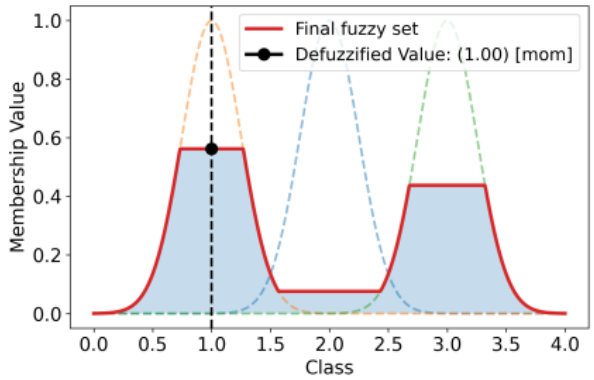
Rule	Antecedent	Consequent
1	$x\ is\ \text{smaller3} \wedge y\ is\ \text{smaller2} \wedge x\ is\ \text{smaller2}$	$class\ is\ 1$
2	$x\ is\ \text{smaller3} \wedge y\ is\ \text{smaller2} \wedge x\ is\ \text{greater2}$	$class\ is\ 2$
3	$x\ is\ \text{smaller3} \wedge y\ is\ \text{greater2} \wedge x\ is\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

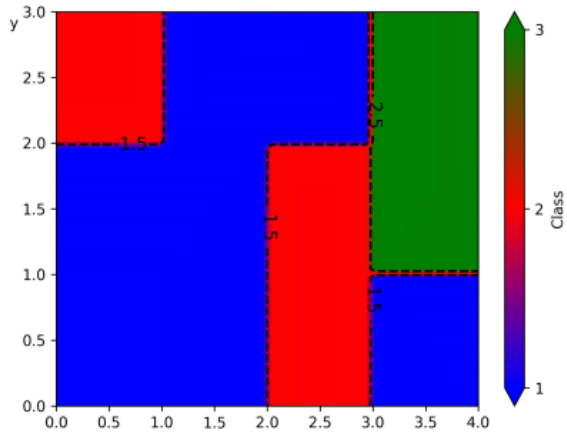
Fuzzy Decision Surfaces - CoG vs. MoM

- CoG: Interpolation effect + errors, smooth boundaries
- MoM: Hard boundaries, similar to Decision Trees









Fuzzy Rule Extraction for `md_flexible`

- Collect huge dataset of `md_flexible` simulations
- LiveInfoData: `maxDensity, homogeneity, threadCount, ...`
- TuningData: `Container, Traversal, Newton3, ..., Time`
- Introduce notion of *relative speed* for each configuration
  - $t_{best}^{(i)}$: Best configuration time in tuning phase i
  - $t_{config}^{(i)}$: Time of configuration in tuning phase i

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

  - Allows for fair comparison between different tuning phases

Resulting Dataset

- Performance of configurations in different environments
- Can be used to extract rules for both approaches
- Goal: Find configurations with high relative speed

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
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Rulefile Approach 1: Component Tuning

- Naively remove bad configurations (relative speed $< 70\%$)
- Group remaining dataset by tuning-phase
- Aggregate present (*good*) values for each parameter
- Apply Rule Extraction algorithm to 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"
⋮	⋮	⋮	⋮	⋮

Rulefile Approach 2: Suitability Tuning

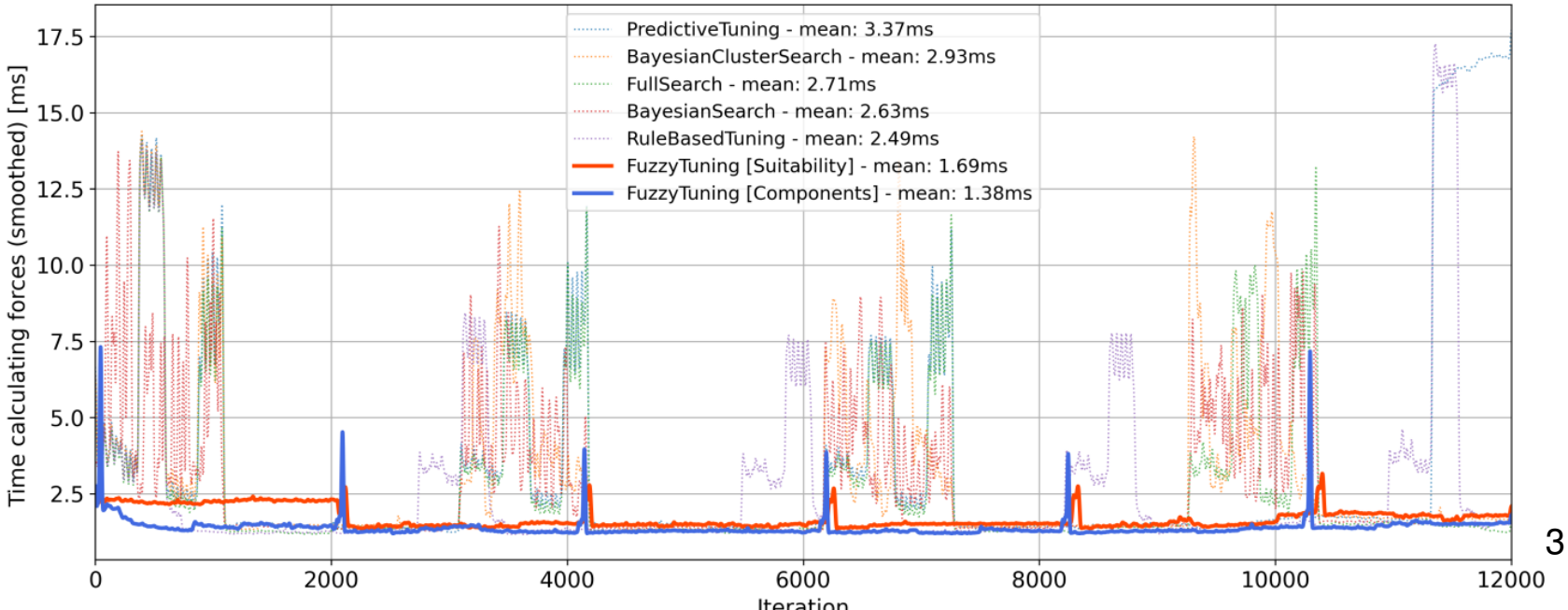
- Assign suitability-classes to each entry in the dataset
- Classes represent ranges of relative speed
- Apply Rule Extraction algorithm for the suitability class

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"
⋮	⋮	⋮	⋮	⋮

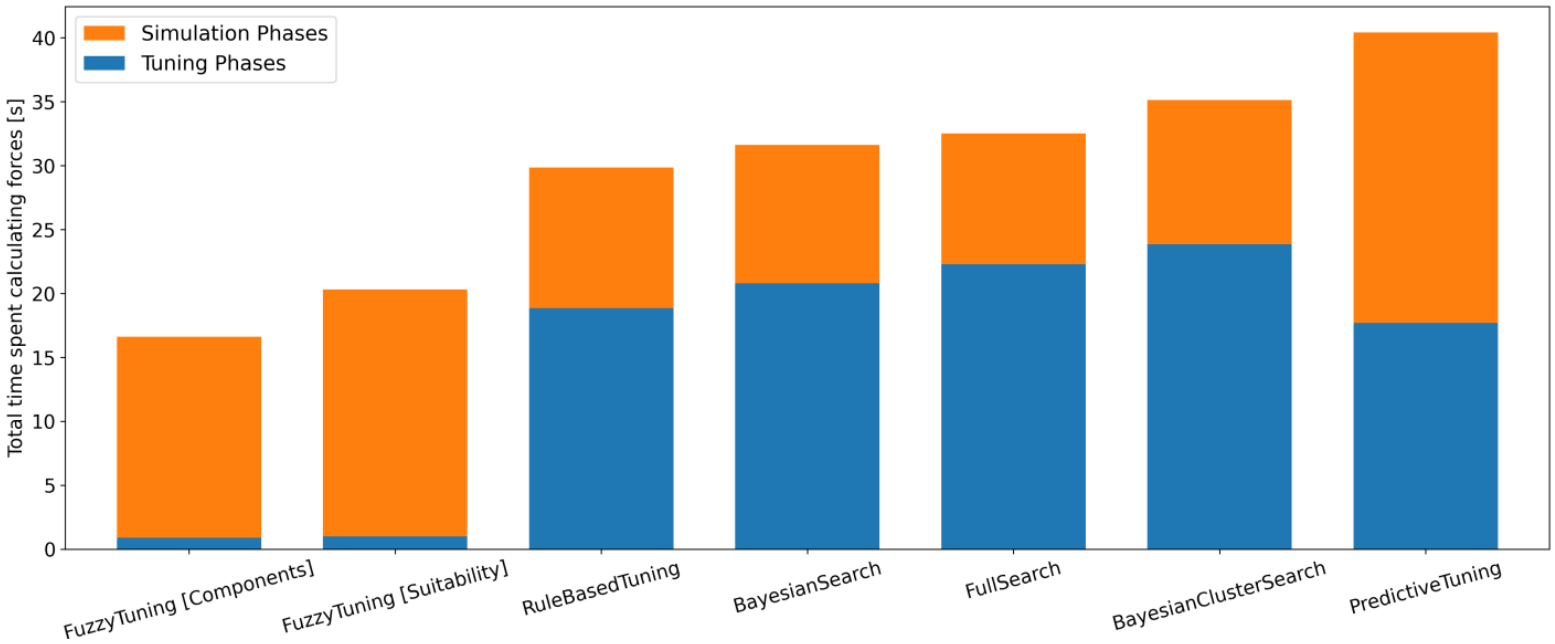
Comparison and Evaluation: Exploding Liquid

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



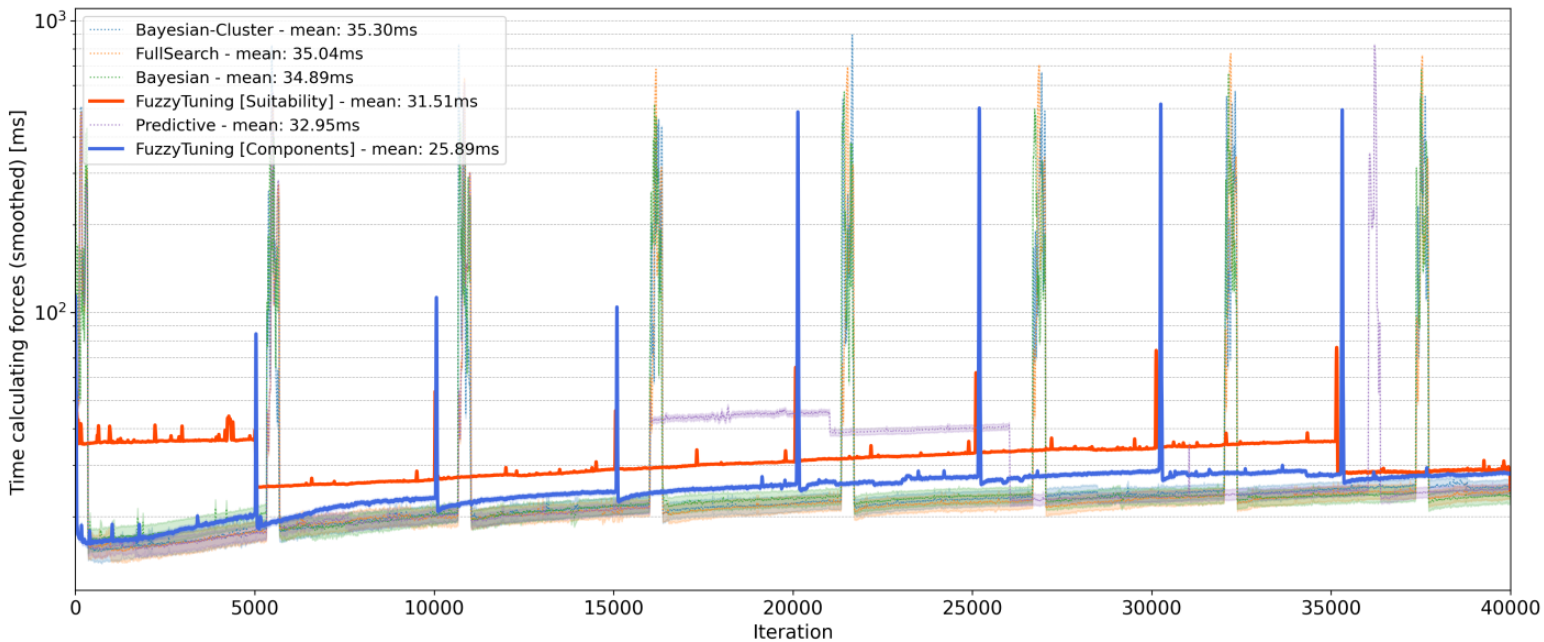
Comparison and Evaluation: Exploding Liquid

- Exploding Liquid Benchmark (Included in Dataset)
  - Fuzzy tuning performs the best by far
  - Mostly due to the very efficient tuning phases
    - Few configurations evaluated
    - Evaluated configurations are expected to perform well
  - Benefits of tuning with tiny overhead



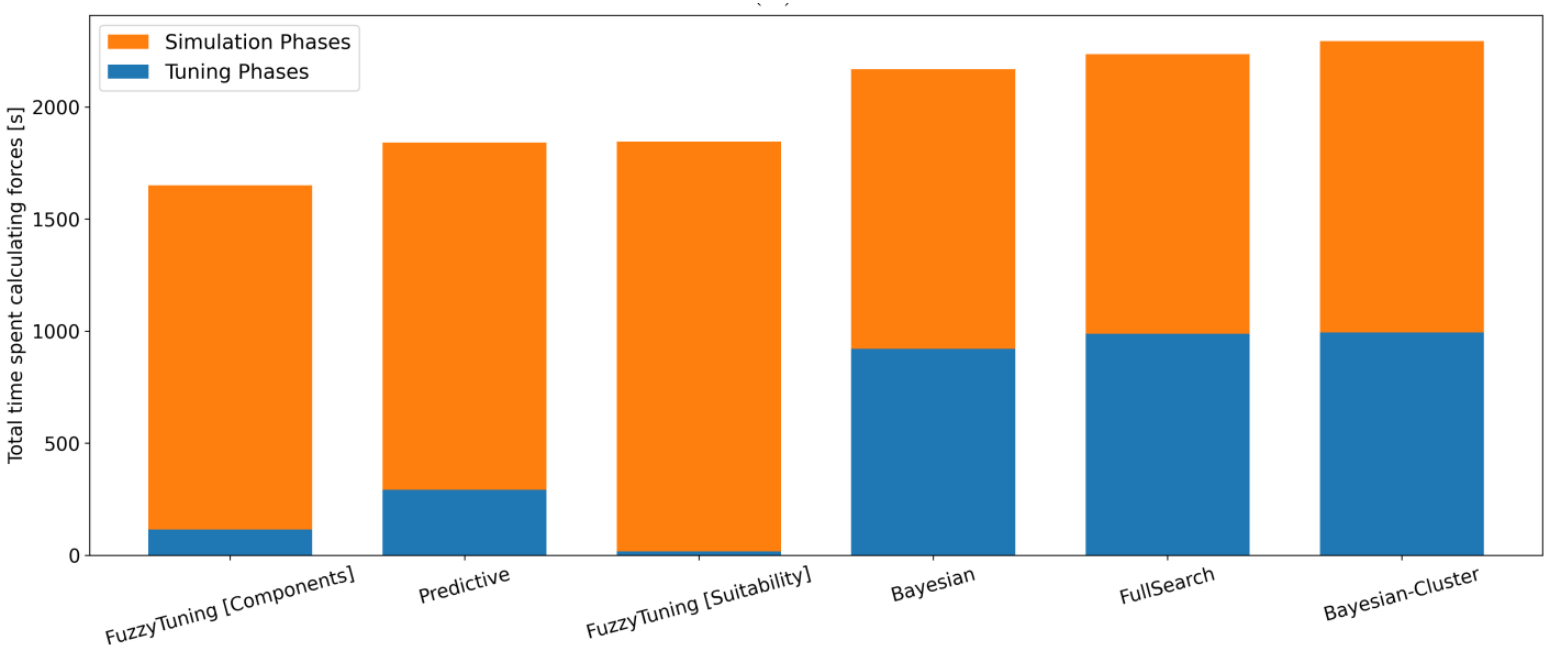
Comparison and Evaluation: Spinodal Decomposition MPI

- Spinodal Decomposition (Included in Dataset)
  - Fuzzy tuning promising again
  - Suitability approach struggles, often misses the best configuration
  - However, both approaches have efficient tuning phases



Comparison and Evaluation: Spinodal Decomposition

- Spinodal Decomposition (Indiretly included in Dataset)
  - Fuzzy tuning performs relatively well
  - Suitability approach still quite good, despite never finding the best configuration
    - Classical tuning strategies perform poorly in tuning phases
    - Fast tuning phases can compensate for suboptimal configurations!



Future Work

- Dynamic Rule Generation
  - Update the expert knowledge on the fly
  - Adapt to new scenarios
  - Share rules between different use cases
- Improving Tuning Strategies
  - Implement early stopping mechanism
  - Avoid evaluating extremely bad configurations
- Simplification of the Fuzzy System to Decision Trees
  - Use Decision Trees instead of Fuzzy Systems
  - Simplify the tuning process
  - Make the tuning process more transparent

Conclusion

- Fuzzy Logic is very promising for tuning AutoPas
- Data-Driven Rule Extraction is a powerful tool
- However:
  - Requires a lot of prior data (current training data is 1.1GB)
  - Users cannot be expected to collect this data beforehand
  - A more user-friendly approach is needed for broader adoption
- Future work could focus on simplifying the process
- Alternatively: Investigate Early Stopping, to solve the tuning overhead once and for all

Thank you for your attention!

Questions?

References I

Keeley Crockett, Zuhair Bandar, David Mclean, and James OShea.

On constructing a fuzzy inference framework using crisp decision trees.

Fuzzy Sets and Systems, 157(21):2809–2832, 2006.

Ali Mohammed, Jonas H. MÃ¼ller KorndÃ¶rfer, Ahmed Eleliemy, and Florina M. Ciorba.

Automated scheduling algorithm selection and chunk parameter calculation in openmp.

IEEE Transactions on Parallel and Distributed Systems, 33(12):4383–4394, 2022.

Samuel James Newcome, Fabio Alexander Gratl, Philipp Neumann, and Hans-Joachim Bungartz.

Towards the smarter tuning of molecular dynamics simulations, Feb 2023.

Amsterdam, The Netherlands.