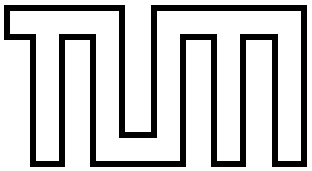


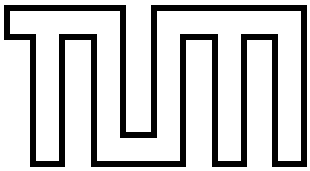
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Exploring Fuzzy Tuning Technique for Molecular Dynamics Simulations in AutoPas

Manuel Lerchner



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Exploring Fuzzy Tuning Technique for Molecular Dynamics Simulations in AutoPas

Remove all TODOS

Untersuchung von Fuzzy Tuning Verfahren für Molekulardynamik-Simulationen in AutoPas

Author:	Manuel Lerchner
Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisors:	Manish Kumar Mishra, M.Sc. &
	Samuel Newcome, M.Sc.
Date:	10.08.2024

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 10.08.2024 Manuel Lerchner

Contents

3 Implementation

This chapter describes the implementation of the Fuzzy Tuning technique in AutoPas. The implementation is divided into three main parts: the generic Fuzzy Tuning framework, the Tuning Strategy, and the Rule Parser. The Fuzzy Tuning framework is the core of this implementation and implements the mathematical foundation of this technique. The Tuning Strategy is the interface between the Fuzzy Tuning framework and the AutoPas simulation. It is responsible for interacting with AutoPas and updating the queue of configurations. The Rule Parser is responsible for parsing the rule base supplied by the user and converting it into the internal representation used by the Fuzzy Tuning framework. The implementation of the Fuzzy Tuning technique in AutoPas is designed to be as generic as possible to allow for easy integration of new types of rule bases.

3.1 Fuzzy Tuning Framework

The Fuzzy Tuning framework implements the mathematical foundation of the Fuzzy Tuning technique. It consists of several components that work together to apply the Fuzzy Rules to the input variables and generate the output variables. The components of the Fuzzy Tuning framework are as follows:

- **Crisp Set:** The Crisp Set is used to model k-cells used as the underlying sets over which the Fuzzy Sets are defined. A k-cell is a hyperrectangle in the k-dimensional space constructed from the Cartesian product of k intervals $I = I_1 \times I_2 \times \ldots \times I_k$ where $I_i = [x_{low}, x_{high}] \subset \mathbb{R}$ is an interval in the real numbers. They are used to define the underlying sets over which the corresponding fuzzy set is defined and can be thought of as the parameter space of the input variable.
- **Fuzzy Set:** A fuzzy set consists of a membership function that assigns a degree of membership to each element of the Crisp Set. There are two types of membership functions: The `BaseMembershipFunction` and the `CompositeMembershipFunction`. The `BaseMembershipFunction` implement a function $f : \mathbb{R} \to [0, 1]$ that directly maps the crisp value to the degree of membership. The `CompositeMembershipFunction` is used to create new fuzzy sets by recursively combining existing fuzzy sets and their membership functions with generic functions. This helps to split up the complex fuzzy sets of the rule base into smaller, more manageable parts. Lets say we have a fuzzy set $\tilde{A}$ defined over the Crisp Set X and a fuzzy set $\tilde{B}$ defined over the Crisp Set Y. Both membership functions are functions mapping the respective Crisp Set to the degree of membership ($\mu_{\tilde{A}} : X \to [0, 1]$ and $\mu_{\tilde{B}} : Y \to [0, 1]$). Since both membership functions directly map the Crisp Set to the degree of membership, they are considered `BaseMembershipFunctions`. When we want to create a new fuzzy set $\tilde{C} = \tilde{A} \cap \tilde{B}$ this new fuzzy set is defined over the Crisp Set $X \times Y$ and thus needs to provide a

add chcpter

membership function $\mu_{\tilde{C}} : X \times Y \to [0, 1]$. As described in this membership function is defined as $\mu_{\tilde{C}}(x, y) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y))$, which can be thought of as recursively combining the membership functions of $\tilde{A}$ and $\tilde{B}$ with the minimum function.

This way of combining fuzzy sets builds a tree structure where the leafs calculate a direct membership value and the inner nodes combine the membership values of their children and pass them up to their parent.

add image

- **Linguistic Variable:** A linguistic variable is a variable whose values are terms in a natural language. Each term is associated with a fuzzy set that defines its concept using the language of fuzzy logic. The linguistic variables can then be used to express rules in a human-readable way.
- **Fuzzy Rule:** A fuzzy rule is a conditional statement that describes the relationship between input- and output variables. It consists of an antecedent and a consequent both of which are fuzzy sets. During the evaluation of the rule, the antecedent is evaluated to determine the degree to which the rule is satisfied and thus the effect of the rule can be reduced accordingly.
- **Fuzzy Control System:** The Fuzzy Control System combines all the concepts described above to create a system that can evaluate a set of fuzzy rules and generate an output based on the input variables. Such a control system acts like a black box $f : \mathbb{R}^n \to \mathbb{R}$ that takes a set of crisp input variables and returns the predicted value.

Add UML Diagram

3.2 Rule Parser

The Rule Parser is responsible for parsing the rule base supplied by the user and converting it into the internal representation used by the Fuzzy Tuning framework. It makes use of the ANTLR4[1] parser generator to create a parser for a domain-specific language tailored to the needs of the Fuzzy Tuning. The language is inspired by common standards such as the Fuzzy Control Language (FCL)[2] but is more lightweight and also allows for the connection of the Fuzzy Tuning framework to the AutoPas simulation. The transformation of the parsed rules into the internal representation is done by a visitor pattern that traverses the parse tree generated by ANTLR4 and internally builds the corresponding objects of the Fuzzy Tuning framework.

Appart from the obvious grammar rules for the fuzzy sets and rules, the language encompasses a few additional constructs to allow for the connection to the AutoPas simulation. This is a crucial part of the tuning framework as the continuous outputs of the Fuzzy Control Systems need to be mapped to the discrete configuration space of the AutoPas simulation. We chose a approach inspired by Mohammed et al. [MKEC22]'s work on scheduling algorithms and used a ranking system that embeds all available configurations in the continuous output space of the Fuzzy Control System. The Tuning Strategy then selects the configuration closest to the predicted value.

Add example image

An example of a rule file making use of this construct is described in Listing 3.2.

The full grammar specification of the language can be found in Listing A.1.

[1] https://www.antlr.org/

[2] https://www.fuzzylite.com/

3.3 Tuning Strategy

The Tuning Strategy implements the interface between the Fuzzy Tuning framework and the AutoPas simulation. It is responsible for interacting with AutoPas and updating the queue of configurations.

It does this by evaluating all Fuzzy Systems present in the rule file with the current LiveInformation of the simulation. The LiveInformation is a snapshot of the current state of the simulation and contains summary statistics about various aspects of the simulation. This evaulation yields a list of all configurations selected by the Fuzzy Control Systems and those replace the current queue of configurations which need to be tested.

Currently two different modes of interpreting the rules are implementet:

3.3.1 Configuration Suitability Approach

The Suitability Approach is designed to tackle rule bases that try to directly predict a suitability value for *each* configurations. This means that the rule file needs to define $\#Containers \cdot \#Traversals \cdot \#DataLayouts \cdot \#Newton3\_options$ different Fuzzy Control Systems. One for each possible configuration of those parameters. As a result, we receive a direct ranking (lets say from 0 to 100% applicability) for each configuration. The Tuning Strategy then selects all configurations within a certain threshold of the highest ranking configuration and rewrites the queue of configurations with these selected configurations.

This method is a natural choice for rule bases as its style closely relates to the idea of Fuzzy Control Systems. It is possible to make full use of the continuous output space of the Fuzzy Control Systems and the method provides a straight forward mapping to the discrete AutoPas configuration space.

The huge downside of this approach is the need to define massive amount of Fuzzy Control Systems and rules in the rule file. This leads to a very big rule file which quickly becomes infeasible to maintain by hand. The code snippet in Listing 3.1 shows a small excerpt of a rule file using the Suitability Approach.

```
FuzzySystemSettings:
    defuzzificationMethod: "CoG"
    interpretOutputAs: "Suitability"

// Input Variables
FuzzyVariable: domain: "threadCount" range: (-20.34, 49.34)
    "higher than 18.0":   SigmoidFinite(39.38, 18.0, -3.23)
    ...

FuzzyVariable: domain: "particlesPerCellStdDev" range: (-0.017, 0.07)
    "lower than 0.029":   SigmoidFinite(0.055, 0.029, 0.004)
    ...

//Define Suitability Variable for a specific configuration
FuzzyVariable: domain: "Suitability LinkedCells_AoS_lc_c01_disabled" range: (0, 1)
    "terrible":           Gaussian(0.125, 0.018)
    "rubbish":            Gaussian(0.25,  0.018)
    "bad":                Gaussian(0.375, 0.018)
    "medium":             Gaussian(0.5,   0.018)
    "ok":                 Gaussian(0.625, 0.018)
    "good":               Gaussian(0.75,  0.018)
    "excellent":          Gaussian(0.875, 0.018)

...

// A rule describing the suitability of a specific configuration under certain conditions
if ("threadCount" == "higher than 18.0") && ("particlesPerCellStdDev" == "lower than 0.029")
  then ("Suitability_LinkedCells_AoS_lc_c01_disabled" == "bad")

...
```

Listing 3.1: Rule snippet depicting the Suitability Approach

3.3.2 Parameter Tuning Approach

A more lightweight approach is the Parameter Tuning Approach. This approach creates a single Fuzzy Control System for each tunable parameter. All Fuzzy Control Systems then attempt to independently predict the optimal value for their respective parameter. Using this approach, the rule file only needs to define $\#Parameters$ Fuzzy Control Systems which is much more manageable. The drawback of this approach is that the continuous output space of the Fuzzy Control Systems needs to be directly translated to discrete values, which is a non-trivial task. The main problem lies in the fact that the tunable parameters are nominal values and thus have no natural order. This means that the interpolation between different linguistic terms, which is naturally performed by the Fuzzy Control Systems, is not meaningful in this context. To avoid this problem, we can however use a defuzzification method that does not rely on performing some kind of interpolation. One such method is the SOM method which selects the smallest x-value for which the membership function is maximal. By using this method there is only a discrete set of possible outputs which can be directly mapped to the nominal values of the tunable parameters by making use of the OutputMapping construct of the Rule Parser. Since there is no interpolation between the linguistic terms, the order of the terms in the OutputMapping can be chosen arbitrarily.

The code snippet in Listing 3.2 shows a small excerpt of a rule file using the Parameter Tuning Approach. In this rule file both the *container* and the *dataLayout* parameter are predicted together as they are very related anyway.

find citations for fuzzy rules for nominal values

maybe rename individual systems

```
FuzzySystemSettings:
    defuzzificationMethod: "SoM"
    interpretOutputAs: "IndividualSystems"

// Input Variables
FuzzyVariable: domain: "threadCount" range: (-20.34, 49.34)
    "lower than 18.0":    SigmoidFinite(39.34, 18.0, -3.34)
    ...

FuzzyVariable: domain: "homogeneity" range: (-0.024, 0.156)
    "lower than 0.049":   SigmoidFinite(0.093, 0.049, 0.005)
    ...

//Define a variable with nominal values
FuzzyVariable: domain: "Container_DataLayout" range: (0, 4)
  "LinkedCells_SoA, VerletClusterLists_SoA, VerletListsCells_AoS":                    Gaussian(0.666, 0.133)
  "LinkedCells_SoA, VerletClusterLists_SoA, VerletListsCells_AoS, VerletListsCells_SoA": Gaussian(1.333, 0.133)

...

// Define how the defuzzified output should be mapped to AutoPas configurations
OutputMapping:
  "Container_DataLayout":
    0.666 =>  [container="LinkedCells", dataLayout="SoA"],
              [container="VerletClusterLists", dataLayout="SoA"],
              [container="VerletListsCells", dataLayout="AoS"]
    1.333 =>  [container="LinkedCells", dataLayout="SoA"],
              [container="VerletClusterLists", dataLayout="SoA"],

    ...

// A rule describing the optimal configuration under certain conditions
if ("homogeneity" == "lower than 0.049") && ("threadCount" == "lower than 18.0")
  then ("Container_DataLayout" == "VerletClusterLists_SoA, VerletListsCells_AoS")

...
```

Listing 3.2: Rule snippet depicting the Parameter Tuning Approach

List of Figures

List of Tables

Bibliography

[BMK96] Bernadette Bouchon-Meunier and Vladik Kreinovich. Axiomatic description of implication leads to a classical formula with logical modifiers: (in particular, mamdani's choice of "and" as implication is not so weird after all). 1996.

[CBMO06] Keeley Crockett, Zuhair Bandar, David Mclean, and James O'Shea. On constructing a fuzzy inference framework using crisp decision trees. *Fuzzy Sets and Systems*, 157(21):2809–2832, 2006.

[GSBN21] Fabio Alexander Gratl, Steffen Seckler, Hans-Joachim Bungartz, and Philipp Neumann. N ways to simulate short-range particle systems: Automated algorithm selection with the node-level library autopas. *Computer Physics Communications*, 273:108262, 2021.

[GST+19] Fabio Alexander Gratl, Steffen Seckler, Nikola Tchipev, Hans-Joachim Bungartz, and Philipp Neumann. Autopas: Auto-tuning for particle simulations. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 748–757, 2019.

[LM15] Benedict Leimkuhler and Charles Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Interdisciplinary Applied Mathematics. Springer, May 2015.

[MKEC22] Ali Mohammed, Jonas H. Müller Korndörfer, 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.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[SGH+21] Steffen Seckler, Fabio Gratl, Matthias Heinen, Jadran Vrabec, Hans-Joachim Bungartz, and Philipp Neumann. Autopas in ls1 mardyn: Massively parallel particle simulations with node-level auto-tuning. *Journal of Computational Science*, 50:101296, 2021.

[VBC08] G. Viccione, V. Bovolin, and E. Pugliese Carratelli. Defining and optimizing algorithms for neighbouring particle identification in sph fluid simulations. *International Journal for Numerical Methods in Fluids*, 58(6):625–638, 2008.