Constraints Always Satisfied Parameters (CASPs) for Fuzzy Sets Optimisation

Chao Chen, *Member, IEEE,* Jerry M. Mendel, *Life Fellow, IEEE* and Jonathan M. Garibaldi, *Fellow, IEEE*

***Abstract*—The design of membership functions in fuzzy systems often requires satisfying domain, semantic, and relational constraints. Existing methods, while effective at enforcing parameter bounds, often lack flexibility or fail to address complex relational constraints. To overcome these limitations, this paper introduces the Constraints Always Satisfied Parameters (CASPs) framework, which inherently satisfies constraints during optimisation. Three variants of the CASPs are proposed, each balancing design flexibility, performance, and interpretability differently. Experimental evaluations on the Electricity and Laser datasets demonstrate consistent constraint satisfaction across all runs, with CASPs-Single prioritising interpretability, CASPs-Free excelling in RMSE performance, and CASPs-Adapted offering a balanced approach. The results highlight the potential of CASPs to enhance the design and optimisation of fuzzy systems.**

***Index Terms*—CAASPs, CASPs, Constraints Always Satisfied Parameters, Fuzzy set optimisation, Fuzzy system, Membership function design**

I. Introduction

FUZZY systems rely on membership functions (MFs) to model uncertainties and represent linguistic terms. Designing these MFs often involves satisfying constraints that ensure semantic integrity, operational validity, and mathematical correctness. For example, in Type-1 (T1) fuzzy systems, these constraints typically include maintaining specific shapes, overlapping with neighbouring MFs, and preserving positive values for certain parameters. Consequently, optimising T1 MF parameters is inherently a constrained optimisation problem.

Various methodologies have been proposed to handle constraints in MF optimisation. As described in [1], MF constraints can be satisfied using different strategies, which can be broadly categorised as follows.

1) Geometric and Semantic Approaches:
   - Enforcing fixed geometric properties, such as symmetric triangular shapes [2, 3].
   - Requiring T1 MFs to intersect (overlap) only with their nearest neighbours [4, 5], sometimes in a way that forms a strong fuzzy partition [6, 7, 8].
   - Preserving the semantic integrity of MFs [9].
2) Reparameterisation Techniques:
   - Transforming parameters to indirectly enforce constraints, such as ensuring intersection points at a grade of 0.5 [10].
3) Mathematical Transformations:
   - Enforcing positivity and boundedness using techniques such as squaring or sigmoid transformations [11, 12].

While these methods successfully enforce constraints, they often limit the degrees of freedom in MF design, potentially restricting fuzzy system performance. Moreover, traditional methods often encode constraints heuristically or enforce them in an ad-hoc manner within the optimisation process, making it challenging to ensure both optimal performance and consistent implementation. This lack of a systematic approach affects reproducibility and transparency in fuzzy system design, highlighting the need for a more structured methodology to handle constraints explicitly.

Recognising these challenges, Mendel introduced the Constraints Almost Always Satisfied Parameters (CAASPs) as a method to ensure that prespecified T1 MF constraints are satisfied through a reparameterisation [1, pp. 204–209] in which auxiliary parameters are mapped to MF parameters so that shape and overlap constraints are satisfied during the optimisation process. By doing so, CAASPs minimise the need for such manual constraint handling and streamline the optimisation workflow. However, CAASPs do not address "Universe constraints" (e.g., requiring all MF parameters to lie within a bounded region, such as [0, 10]), which necessitates separate handling, and which is why the "almost always" appears in the name of those parameters.

Building on these foundations, this paper presents the Constraints Always Satisfied Parameters (CASPs), an enhanced methodology that guarantees the satisfaction of all prespecified constraints, including Universe constraints.

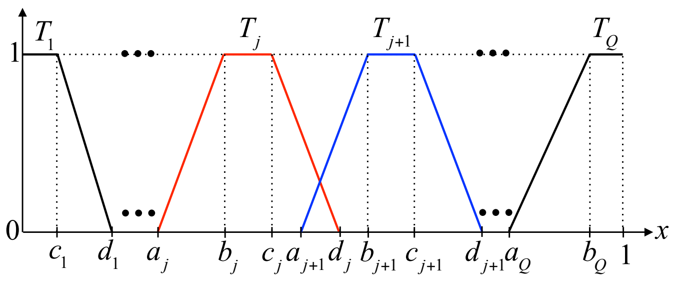
This work explores CASPs in the context of T1 fuzzy systems and demonstrates their effectiveness through detailed simulations. Specifically, this paper provides: 1), a robust reparameterisation method that guarantees the satisfaction of all T1 MF constraints, including Universe constraints; 2), algorithms that integrate CASPs into optimisation workflows; and 3) comparative analyses showcasing the impact of CASPs on design flexibility, system performance, and interpretability.

The remainder of this paper is structured as follows. Section II introduces the CASPs methodology, detailing its formulation and implementation. Section III explains how to use the CASPs during optimisation. Section IV presents the experimental setup, datasets and results, evaluating the effectiveness of CASPs. Section V discusses the results,

C. Chen is with the Laboratory for Uncertainty in Data and Decision Making (LUCID) and the Intelligent Modelling and Analysis (IMA) Research Groups, School of Computer Science, University of Nottingham, Nottingham, Jubilee Campus, NG8 1BB UK (e-mail: chao.chen@nottingham.ac.uk).

J.M. Mendel is with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA (e-mail: mendel@usc.edu).

J.M. Garibaldi is with the University of Nottingham, Ningbo, China (e-mail: jon.garibaldi@nottingham.edu.cn).



T_1
T_j
T_{j+1}
T_Q
1
0
c_1 d_1 a_j b_j c_j a_{j+1} d_j b_{j+1} c_{j+1} d_{j+1} a_Q b_Q 1
x

Fig. 1. T1 trapezoidal MFs, named T1, …, TQ

highlighting the trade-offs between three variants of CASPs in terms of interpretability and optimisation performance. Finally, Section VI concludes the paper with key findings and directions for future research.

II. CASPs Methodology

To illustrate the CASPs methodology, we consider trapezoidal MFs (see Fig. 1), for which the universe of discourse is $[-L, R]$, but is transformed to $[0, 1]$, so that all MF parameters within this range are positive, which makes it much easier to specify and test MF constraints.

A. Notations

The following notation is used.

- Q: The number of MFs. Note that we have $Q \geq 2$, so there will be at least two MFs (left and right shoulders).
- $\boldsymbol{\theta}$: A vector $(\theta_1, \theta_2, \ldots, \theta_{4Q-4})^T$ of the parameters for trapezoidal MFs representing the positions of vertices.
- $\boldsymbol{\psi}$: A vector $(\psi_1, \psi_2, \ldots, \psi_{4Q-4})^T$ of auxiliary CASPs used to reparameterise $\boldsymbol{\theta}$.

For the MFs in Fig. 1, the vector $\boldsymbol{\theta}$ can be explicitly represented as:

$$\boldsymbol{\theta} = \Big(c_1, d_1, (a_j, b_j, c_j, d_j)_{j=2}^{Q-1}, a_Q, b_Q\Big)^T, \tag{1}$$

where:

- c_1, d_1 correspond to θ_1, θ_2, and are parameters of the left-shoulder MF.
- a_j, b_j, c_j, d_j correspond to θ_{4j-5}, θ_{4j-4}, θ_{4j-3}, θ_{4j-2}, and are parameters of the interior MF (applicable only when $Q \geq 3$).
- a_Q, b_Q correspond to $\theta_{4Q-5}, \theta_{4Q-4}$, and are parameters of the right-shoulder MF.

Hence, $\boldsymbol{\theta}$ can be equivalently expressed as:

$$\boldsymbol{\theta} = \Big(\theta_1, \theta_2, (\theta_{4j-5}, \theta_{4j-4}, \theta_{4j-3}, \theta_{4j-2})_{j=2}^{Q-1}, \theta_{4Q-5}, \theta_{4Q-4}\Big)^T \tag{2}$$

B. CASPs Variants

There is no best single set of constraints. Different systems or scenarios may require different types of constraints. In this subsection, we introduce the main types of constraints and present three CASPs variants namely CASPs-Single, CASPs-Adapted, and CASPs-free. These variants have different predefined constraints, which may lead to trade-offs between flexibility, performance, and interpretability.

All CASPs variants share the following fundamental constraints:

1) **Domain constraints:** All MF parameters ($\boldsymbol{\theta}$) must be within a range, typically corresponding to the universe of discourse. In this paper, this range is $(0, 1)$ to ensure positivity and boundedness. Note that the open interval is used here for $\boldsymbol{\theta}$ instead of the closed range $[0, 1]$, which is for the universe of discourse. This will be helpful for later reparameterisation, allowing smooth transformations between $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$, where $\boldsymbol{\psi}$ has an unconstrained range $(-\infty, \infty)$. Since $\boldsymbol{\theta}$ can get arbitrarily close to 0 or 1, such constraints should not have any practical impact on systems. Note that domain constraints may not be necessary in some application scenarios; but, in this paper, we mainly focus on the cases with domain constraints. Hence, such CASPs variants are also named as scaled versions below.
2) **Semantic Integrity:** The vertices of trapezoidal MFs must maintain a specific order to preserve their geometric shape. For an MF with vertices a, b, c, d (as in Fig. 1), the relationship $a < b < c < d$ must always hold. This ensures that the MF remains a valid trapezoid.

The main difference between the three types of CASPs lies in their constraints for MF overlaps[1]:

1) **CASPs-Single:** Enforces the strictest overlap constraints. Each MF is guaranteed to overlap only with its nearest neighbours, with no additional overlaps. This ensures that each trapezoidal MF has exactly one overlap on its left side and one on its right side.
2) **CASPs-Adapted:** Also guarantees overlaps but allows greater flexibility by permitting a single MF to overlap with multiple neighbours. This approach is an adapted version of CAASPs, sharing the same types of constraints but differing in the reparameterisation methodology.
3) **CASPs-Free:** Provides the most freedom because overlaps between MFs are not enforced, allowing for greater flexibility during optimisation while adhering only to domain and semantic integrity constraints.

C. CASPs-Single

The constraints for **CASPs-Single** are (see Fig. 1 for an illustration, $j = 2, 3, \ldots, Q-1$ and $Q \geq 3$.[2]):

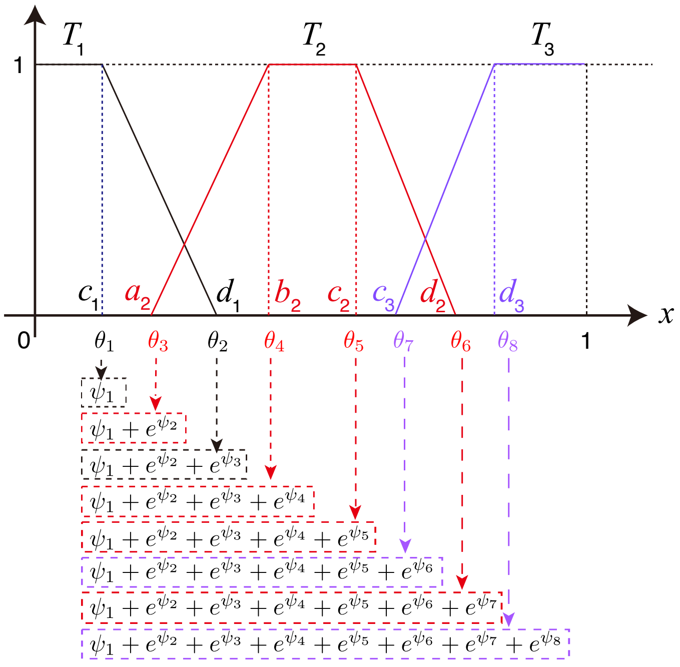
$$T_1 : \left\{ \overbrace{d_1 > c_1}^{T_1\ \#1} \right\} \tag{3}$$

$$T_j : \left\{ \overbrace{a_j > c_{j-1}}^{T_j\ \#1},\ \overbrace{d_{j-1} > a_j}^{T_j\ \#2},\ \overbrace{b_j > d_{j-1}}^{T_j\ \#3},\ \overbrace{c_j > b_j}^{T_j\ \#4} \right\} \tag{4}$$

$$T_Q : \left\{ \overbrace{a_Q > c_{Q-1}}^{T_Q\ \#1},\ \overbrace{d_{Q-1} > a_Q}^{T_Q\ \#2},\ \overbrace{b_Q > d_{Q-1}}^{T_Q\ \#3} \right\} \tag{5}$$

[1]MF overlap guarantees continuity of the output of a type-1 fuzzy system [13].

[2]These values for j and Q also apply to (7), (13), (16), (19) and (22).



T_1
T_2
T_3
1
0
c_1
a_2
d_1
b_2
c_2
c_3
d_2
d_3
x
θ_1
θ_3
θ_2
θ_4
θ_5
θ_7
θ_6
θ_8
1
ψ_1
$\psi_1 + e^{\psi_2}$
$\psi_1 + e^{\psi_2} + e^{\psi_3}$
$\psi_1 + e^{\psi_2} + e^{\psi_3} + e^{\psi_4}$
$\psi_1 + e^{\psi_2} + e^{\psi_3} + e^{\psi_4} + e^{\psi_5}$
$\psi_1 + e^{\psi_2} + e^{\psi_3} + e^{\psi_4} + e^{\psi_5} + e^{\psi_6}$
$\psi_1 + e^{\psi_2} + e^{\psi_3} + e^{\psi_4} + e^{\psi_5} + e^{\psi_6} + e^{\psi_7}$
$\psi_1 + e^{\psi_2} + e^{\psi_3} + e^{\psi_4} + e^{\psi_5} + e^{\psi_6} + e^{\psi_7} + e^{\psi_8}$

Fig. 2. An illustration of how parameters $\boldsymbol{\theta}$ are derived from $\boldsymbol{\psi}$ in Algorithm 1, for the case of three MFs. The correspondence between a, b, c, d and $\boldsymbol{\theta}$ is defined in (1) and (2).

These constraints enforce a strict order of parameters such that a_2 is guaranteed to lie between c_1 and d_1, ensuring one and only one overlap between adjacent MFs. Similarly, b_2 is restricted to be greater than d_1 and less than c_2. This is illustrated in Fig. 2, using the example of 3 MFs. These constraints ensure a clean separation and a well-defined overlapping structure.

Constraints (3) – (5) can also be expressed in terms of the parameters $\boldsymbol{\theta}$ by using (1) and (2), as is done in (6) – (8). This is especially useful for stating the CASPs Algorithms 1 – 6.

$$T_1 : \left\{ \overbrace{\theta_2 > \theta_1}^{T_1\ \#1} \right\} \tag{6}$$

$$T_j : \left\{ \begin{array}{l} \overbrace{\theta_{4j-5} > \theta_{4j-7}}^{T_j\ \#1},\ \overbrace{\theta_{4j-6} > \theta_{4j-5}}^{T_j\ \#2}, \\ \overbrace{\theta_{4j-4} > \theta_{4j-6}}^{T_j\ \#3},\ \overbrace{\theta_{4j-3} > \theta_{4j-4}}^{T_j\ \#4} \end{array} \right\} \tag{7}$$

$$T_Q : \left\{ \begin{array}{l} \overbrace{\theta_{4Q-5} > \theta_{4Q-7}}^{T_Q\ \#1},\ \overbrace{\theta_{4Q-6} > \theta_{4Q-5}}^{T_Q\ \#2}, \\ \overbrace{\theta_{4Q-4} > \theta_{4Q-6}}^{T_Q\ \#3} \end{array} \right\} \tag{8}$$

To implement these constraints by directly updating the parameters $\boldsymbol{\theta}$ during a constrained optimisation is difficult and requires hardcoded adjustments to ensure their compliance. Instead, in Algorithm 1 we introduce a new set of auxiliary parameters $\boldsymbol{\psi}$ which are mapped to $\boldsymbol{\theta}$ in such a way that the relative orderings of parameters required by the constraints are always guaranteed.

Fig. 2 provides an illustration of how the parameters $\boldsymbol{\theta}$ are derived from $\boldsymbol{\psi}$, specifically for the case of three MFs. In this case, Steps 1.2 and 1.3 in Algorithm 1 compute c_1 and d_1, respectively, for the left shoulder[3]. Steps 1.6–1.9 compute a_2, b_2, c_2 and d_2, respectively, for the interior MF. Steps 1.12 and 1.13 compute a_3 and b_3, respectively, for the right shoulder. Steps 1.14 – 1.16 apply the sigmoid transformation, defined in (9), to ensure that all parameters of $\boldsymbol{\theta}$ are within the range $(0, 1)$. This transformation is necessary in cases where domain constraints require that the parameters be within the interval $(0, 1)$. Such situations include, for example, when the universe of discourse is normalised or when the domain is inherently defined in this range.

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad x \in \mathbb{R} \tag{9}$$

Note that the sigmoid function is only one option used in this paper for the range $(0, 1)$; other transformations may be used, particularly for different ranges. For example, the hyperbolic tangent (Tanh) function maps values to the range $(-1, 1)$, and scaled sigmoid variants can generalise these transformations to other ranges (a, b). If no domain constraints are required, these transformation steps can be omitted.

A formal proof that the constraints defined in CASPs are always satisfied during the parameter update and transformation processes is provided in the Appendix A.

It should also be noted that during optimisation, it is sometimes also necessary to perform the reverse mapping, from $\boldsymbol{\theta}$ to $\boldsymbol{\psi}$. This occurs, for example, at the initialisation stage: we may first specify a set of initial values for $\boldsymbol{\theta}$ that already satisfy all constraints, based on prior knowledge, experimental settings, or arbitrary choices. Since the auxiliary parameters $\boldsymbol{\psi}$ are the ones updated during the optimisation process, the initial values of $\boldsymbol{\theta}$ need to be converted into $\boldsymbol{\psi}$. Algorithm 2 provides this mapping.

In Algorithm 2, Steps 2.2 – 2.4 correspond to the inverse of Steps 1.14 – 1.16 in Algorithm 1. Specifically, while Algorithm 1 applies the sigmoid transformation to map each θ_i to the bounded interval $(0, 1)$, here the reverse operation is performed using the logit function in (10), so that θ_i is mapped back to the range $(-\infty, +\infty)$. In other words, the logit function is the inverse function of $\sigma(x)$. The relationship between x and p in (9) and (10) can be summarised as the equivalence in (11).

$$\text{logit}(p) = \sigma^{-1}(p) = \ln\left(\frac{p}{1-p}\right), \quad p \in (0, 1) \tag{10}$$

$$p = \sigma(x) \iff x = \text{logit}(p), \quad x \in \mathbb{R},\ p \in (0, 1) \tag{11}$$

The subsequent steps of Algorithm 2 compute each $\boldsymbol{\psi}$ from the differences between the specific values of $\boldsymbol{\theta}$. The basic idea is that the increments that were originally introduced through exponential functions in Algorithm 1 can now be recovered by taking the logarithm of the corresponding differences. For example, the difference $\theta_3 - \theta_1$ corresponds to $\exp(\psi_2)$, and therefore ψ_2 can be obtained as shown in Step 2.6.

[3]Notation "Step $k.\ell$" refers to Step ℓ in Algorithm k.

Algorithm 1 Compute θ from ψ for CASPs-Single (Scaled)

```
1: $n \leftarrow 4(Q-1)$
2: $\theta_1 \leftarrow \psi_1$
3: $\theta_2 \leftarrow \theta_1 + \exp(\psi_2) + \exp(\psi_3)$
4: if $Q > 2$ then
5:     for $j = 2$ to $\frac{n}{4}$ do
6:         $\theta_{4j-5} \leftarrow \theta_{4j-7} + \exp(\psi_{4j-6})$
7:         $\theta_{4j-4} \leftarrow \theta_{4j-6} + \exp(\psi_{4j-4})$
8:         $\theta_{4j-3} \leftarrow \theta_{4j-4} + \exp(\psi_{4j-3})$
9:         $\theta_{4j-2} \leftarrow \theta_{4j-3} + \exp(\psi_{4j-2}) + \exp(\psi_{4j-1})$
10:    end for
11: end if
12: $\theta_{n-1} \leftarrow \theta_{n-3} + \exp(\psi_{n-2})$
13: $\theta_n \leftarrow \theta_{n-2} + \exp(\psi_n)$
14: for $i = 1$ to $n$ do
15:     $\theta_i \leftarrow \sigma(\theta_i)$
16: end for
```

Algorithm 2 Calculate ψ from θ for CASPs-Single (Scaled)

```
1: $n \leftarrow 4(Q-1)$
2: for $i = 1$ to $n$ do
3:     $\theta_i \leftarrow \text{logit}(\theta_i)$
4: end for
5: $\psi_1 \leftarrow \theta_1$
6: $\psi_2 \leftarrow \ln(\theta_3 - \theta_1)$
7: if $Q > 2$ then
8:     for $j = 2$ to $\frac{n}{4}$ do
9:         $\psi_{4j-5} \leftarrow \ln(\theta_{4j-6} - \theta_{4j-5})$
10:        $\psi_{4j-4} \leftarrow \ln(\theta_{4j-4} - \theta_{4j-6})$
11:        $\psi_{4j-3} \leftarrow \ln(\theta_{4j-3} - \theta_{4j-4})$
12:        $\psi_{4j-2} \leftarrow \ln(\theta_{4j-1} - \theta_{4j-3})$
13:    end for
14: end if
15: $\psi_{n-1} \leftarrow \ln(\theta_{n-2} - \theta_{n-1})$
16: $\psi_n \leftarrow \ln(\theta_n - \theta_{n-2})$
```

D. CASPs-Adapted

The constraints for **CASPs-Adapted** are:

$$T_1 : \left\{ \overbrace{d_1 > c_1}^{T_1\ \#1} \right\} \tag{12}$$

$$T_j : \left\{ \overbrace{d_{j-1} > a_j}^{T_j\ \#1},\ \overbrace{b_j > a_j}^{T_j\ \#2},\ \overbrace{c_j > b_j}^{T_j\ \#3},\ \overbrace{d_j > c_j}^{T_j\ \#4} \right\} \tag{13}$$

$$T_Q : \left\{ \overbrace{d_{Q-1} > a_Q}^{T_Q\ \#1},\ \overbrace{b_Q > a_Q}^{T_Q\ \#2} \right\} \tag{14}$$

Compared to the constraints for CASPs-Single, the constraints in (12) – (14) are more relaxed. To illustrate the difference, consider again the example with three MFs shown in Fig. 2, which was used earlier to explain CASPs-Single. In that case, a_2 is restricted to lie strictly between c_1 and d_1, ensuring exactly one overlap with MF T_1. In contrast, under CASPs-Adapted, a_2 only needs to be less than d_1, without the requirement of being greater than c_1. Similarly, b_2 does not have to exceed d_1. This relaxation introduces more freedom, enabling a single MF to overlap not only with its immediate neighbours, but also with multiple adjacent or even more distant MFs across the domain. This characteristic makes CASPs-Adapted better suited for applications requiring more flexibility in parameter distributions.

The CASPs-Adapted constraints in (12) – (14) can be expressed using θ_j as:

$$T_1 : \left\{ \overbrace{\theta_2 > \theta_1}^{T_1\ \#1} \right\} \tag{15}$$

$$T_j : \left\{ \begin{array}{l} \overbrace{\theta_{4j-6} > \theta_{4j-5}}^{T_j\ \#1},\ \overbrace{\theta_{4j-4} > \theta_{4j-5}}^{T_j\ \#2}, \\ \overbrace{\theta_{4j-3} > \theta_{4j-4}}^{T_j\ \#3},\ \overbrace{\theta_{4j-2} > \theta_{4j-3}}^{T_j\ \#4} \end{array} \right\} \tag{16}$$

$$T_Q : \left\{ \overbrace{\theta_{4Q-6} > \theta_{4Q-5}}^{T_Q\ \#1},\ \overbrace{\theta_{4Q-4} > \theta_{4Q-5}}^{T_Q\ \#2} \right\} \tag{17}$$

The procedure for computing $\boldsymbol{\theta}$ from $\boldsymbol{\psi}$ in the CASPs-Adapted case is presented in Algorithm 3. Although its overall structure is the same as the structure of Algorithm 1, Steps 3.3–3.13 are different than Steps 1.3–1.13.

A formal proof that the MF constraints in (12) – (14) are always satisfied follows the same reasoning and structure as the proof for CASPs-Single. Each constraint is guaranteed by the construction rules in Algorithm 3, which involve adding or subtracting strictly positive exponential terms to the preceding parameter. The detailed proof is omitted for brevity and is left to the reader.

The procedure for mapping $\boldsymbol{\theta}$ to $\boldsymbol{\psi}$ is presented in Algorithm 4. As for Algorithm 2, this is achieved through a series of reverse operations, e.g., by taking the logarithm of the corresponding differences between specific values of $\boldsymbol{\theta}$.

E. CASPs-Free

The constraints for **CASPs-Free** are:

$$T_1 : \left\{ \overbrace{d_1 > c_1}^{T_1\ \#1} \right\} \tag{18}$$

$$T_j : \left\{ \overbrace{b_j > a_j}^{T_j\ \#1},\ \overbrace{c_j > b_j}^{T_j\ \#2},\ \overbrace{d_j > c_j}^{T_j\ \#3} \right\} \tag{19}$$

$$T_Q : \left\{ \overbrace{b_Q > a_Q}^{T_Q\ \#1} \right\} \tag{20}$$

In this case, the constraints are further relaxed so that only the internal ordering of parameters within each MF is enforced, e.g., $a_2 < b_2 < c_2 < d_2$, as illustrated in Fig. 2. There are no restrictions on the relationships between adjacent MFs, meaning a_2 does not need to be greater than c_1, and b_2 does not need to be less than d_1. This allows complete freedom

Algorithm 3 Compute θ from ψ for CASPs-Adapted (Scaled)

```
1:  $n \leftarrow 4(Q-1)$
2:  $\theta_1 \leftarrow \psi_1$
3:  $\theta_2 \leftarrow \theta_1 + \exp(\psi_2)$
4:  if $Q > 2$ then
5:      for $j = 2$ to $\frac{n}{4}$ do
6:          $\theta_{4j-5} \leftarrow \theta_{4j-6} - \exp(\psi_{4j-5})$
7:          $\theta_{4j-4} \leftarrow \theta_{4j-5} + \exp(\psi_{4j-4})$
8:          $\theta_{4j-3} \leftarrow \theta_{4j-4} + \exp(\psi_{4j-3})$
9:          $\theta_{4j-2} \leftarrow \theta_{4j-3} + \exp(\psi_{4j-2})$
10:     end for
11: end if
12: $\theta_{n-1} \leftarrow \theta_{n-2} - \exp(\psi_{n-1})$
13: $\theta_n \leftarrow \theta_{n-1} + \exp(\psi_n)$
14: for $i = 1$ to $n$ do
15:     $\theta_i \leftarrow \sigma(\theta_i)$
16: end for
```

Algorithm 4 Calculate ψ from θ for CASPs-Adapted (Scaled)

```
1:  $n \leftarrow 4(Q-1)$
2:  for $i = 1$ to $n$ do
3:      $\theta_i \leftarrow \mathrm{logit}(\theta_i)$
4:  end for
5:  $\psi_1 \leftarrow \theta_1$
6:  $\psi_2 \leftarrow \ln(\theta_2 - \theta_1)$
7:  if $Q > 2$ then
8:      for $j = 2$ to $\frac{n}{4}$ do
9:          $\psi_{4j-5} \leftarrow \ln(\theta_{4j-6} - \theta_{4j-5})$
10:         $\psi_{4j-4} \leftarrow \ln(\theta_{4j-4} - \theta_{4j-5})$
11:         $\psi_{4j-3} \leftarrow \ln(\theta_{4j-3} - \theta_{4j-4})$
12:         $\psi_{4j-2} \leftarrow \ln(\theta_{4j-2} - \theta_{4j-3})$
13:     end for
14: end if
15: $\psi_{n-1} \leftarrow \ln(\theta_{n-2} - \theta_{n-1})$
16: $\psi_n \leftarrow \ln(\theta_n - \theta_{n-1})$
```

Algorithm 5 Compute θ from ψ for CASPs-Free (Scaled)

```
1:  $n \leftarrow 4(Q-1)$
2:  $\theta_1 \leftarrow \psi_1$
3:  $\theta_2 \leftarrow \theta_1 + \exp(\psi_2)$
4:  if $Q > 2$ then
5:      for $j = 2$ to $\frac{n}{4}$ do
6:          $\theta_{4j-5} \leftarrow \psi_{4j-5}$
7:          $\theta_{4j-4} \leftarrow \theta_{4j-5} + \exp(\psi_{4j-4})$
8:          $\theta_{4j-3} \leftarrow \theta_{4j-4} + \exp(\psi_{4j-3})$
9:          $\theta_{4j-2} \leftarrow \theta_{4j-3} + \exp(\psi_{4j-2})$
10:     end for
11: end if
12: $\theta_{n-1} \leftarrow \psi_{n-1}$
13: $\theta_n \leftarrow \theta_{n-1} + \exp(\psi_n)$
14: for $i = 1$ to $n$ do
15:     $\theta_i \leftarrow \sigma(\theta_i)$
16: end for
```

Algorithm 6 Calculate ψ from θ for CASPs-Free (Scaled)

```
1:  $n \leftarrow 4(Q-1)$
2:  for $i = 1$ to $n$ do
3:      $\theta_i \leftarrow \mathrm{logit}(\theta_i)$
4:  end for
5:  $\psi_1 \leftarrow \theta_1$
6:  $\psi_2 \leftarrow \ln(\theta_2 - \theta_1)$
7:  if $Q > 2$ then
8:      for $j = 2$ to $\frac{n}{4}$ do
9:          $\psi_{4j-5} \leftarrow \theta_{4j-5}$
10:         $\psi_{4j-4} \leftarrow \ln(\theta_{4j-4} - \theta_{4j-5})$
11:         $\psi_{4j-3} \leftarrow \ln(\theta_{4j-3} - \theta_{4j-4})$
12:         $\psi_{4j-2} \leftarrow \ln(\theta_{4j-2} - \theta_{4j-3})$
13:     end for
14: end if
15: $\psi_{n-1} \leftarrow \theta_{n-1}$
16: $\psi_n \leftarrow \ln(\theta_n - \theta_{n-1})$
```

for the MFs to overlap in any configuration or even have gaps between them, making CASPs-Free highly flexible for exploring global solutions or scenarios where interpretability is secondary to optimisation.

The constraints in (18) – (20), expressed in terms of $\boldsymbol{\theta}$ are in (21) – (23).

$$T_1 : \left\{ \overbrace{\theta_2 > \theta_1}^{T_1\ \#1} \right\} \tag{21}$$

$$T_j : \left\{ \begin{array}{l} \overbrace{\theta_{4j-4} > \theta_{4j-5}}^{T_j\ \#1}, \ \overbrace{\theta_{4j-3} > \theta_{4j-4}}^{T_j\ \#2}, \\ \overbrace{\theta_{4j-2} > \theta_{4j-3}}^{T_j\ \#3} \end{array} \right\} \tag{22}$$

$$T_Q : \left\{ \overbrace{\theta_{4Q-4} > \theta_{4Q-5}}^{T_Q\ \#1} \right\} \tag{23}$$

Algorithm 5 presents the procedure for mapping $\boldsymbol{\psi}$ to $\boldsymbol{\theta}$ in the CASPs-Free case. Steps 5.2, 5.6, and 5.12 construct the values of θ directly from ψ. The remaining Steps 5.3 – 5.13 establish the internal ordering within each MF.

A formal proof that the MF constraints in (18) – (20) are always satisfied for CASPs-Free is very similar to that for CASPs-Single and CASPs-Adapted, and is left to the reader.

Algorithm 6 presents the procedure for mapping $\boldsymbol{\theta}$ to $\boldsymbol{\psi}$. Similar to previous algorithms, this is achieved through reverse operations, such as applying the logit transformation and taking logarithms of the relevant differences, in order to recover the internal ordering within each MF.

F. Comments

By progressively relaxing the constraints from CASPs-Single to CASPs-Free, our proposed framework offers increasing flexibility to accommodate diverse application requirements. Importantly, the CASPs methodology is not limited to the three variants introduced in this paper. Users may define their own types of constraints, such as range constraints (e.g., restricting parameters to intervals like $[0,1]$ or $[0,\infty)$), order constraints (e.g., enforcing a predefined ordering or hierarchy among parameters), or other constraints (e.g., ensuring continuity or preventing abrupt changes in overlap regions, slopes, or distribution patterns).

Since these constraints are incorporated through the same reparameterisation principles, they are guaranteed to be satisfied during optimisation. This makes CASPs a highly flexible methodology that can be tailored to a wide range of application scenarios.

III. How to Use CASPs During Optimization

The following proposed optimisation procedure, which is stated for the trapezoidal MFs in Fig. 1, views an objective function J as a function of the CASPs, i.e., $J \to J(\boldsymbol{\psi})$. Usually, J is stated in terms of the $\boldsymbol{\theta}$ parameters, and so to obtain $J(\boldsymbol{\psi})$ from $J(\boldsymbol{\theta})$, one can either: (1) express all MFs as explicit functions of the CASPs using the formulas in Algorithm 1, 3, or 5, or (2) switch back and forth between the CASP and original parameters during optimisation, i.e., the CASPs are updated by the optimisation algorithm, after which they are converted to the original parameters using Algorithm 2, 4, or 6, so as to compute the value of J.

The steps of the CASP optimisation procedure are:

- *Step 1.* Specify initial values for the CASPs $\boldsymbol{\psi}$, $\boldsymbol{\psi}^{(0)}$. These initial values are not arbitrary but should be computed by first choosing non-random initial values for the original parameters $\boldsymbol{\theta}^{(0)}$, in such a way that all MF constraints are satisfied, e.g., once the number of MFs, Q, is chosen for a variable, then a figure like Fig. 2 can be sketched, after which its $\boldsymbol{\theta}$ parameters can be read off of it. Those parameters are $\boldsymbol{\theta}^{(0)}$. $\boldsymbol{\psi}^{(0)}$ is then computed from $\boldsymbol{\theta}^{(0)}$ by using Algorithms 2, 4, or 6.
- *Step 2.* Compute $y(\boldsymbol{\theta}^{(0)})$ *and* $J(\boldsymbol{\theta}^{(0)})$. Do Steps 3–5 until a Stopping Rule is satisfied ($m = 0, 1, \ldots, m^*$). One stopping rule is to fix the number of iterations of Steps 3–5, m^*, ahead of time. Another stopping rule is to stop when the change in the objective function from the previous iteration falls below a pre-specified threshold value, i.e., when
$$|J(\boldsymbol{\psi}^{(m+1)}) - J(\boldsymbol{\psi}^{(m)})| \leq \varepsilon,$$
then $m = m^*$.
- *Step 3.* Use any preferred optimisation method (e.g., gradient descent based on derivatives $\partial J/\partial \boldsymbol{\psi}$, evolutionary, PSO, etc.) to compute $\boldsymbol{\psi}^{(m+1)}$. It will use $\boldsymbol{\psi}^{(m)}$, and either $J(\boldsymbol{\psi}^{(m)})$ or $J(\boldsymbol{\theta}(\boldsymbol{\psi}^{(m)}))$.
- *Step 4.* Compute $J(\boldsymbol{\psi}^{(m+1)})$. This will require computing the output of the T1 fuzzy system, $y(\boldsymbol{\theta}^{(m+1)}) = y(\boldsymbol{\theta}(\boldsymbol{\psi}^{(m+1)}))$, where $\boldsymbol{\theta}^{(m+1)}$ is computed using Algorithms 1, 3, or 5.
- *Step 5.* STOP? If YES, then set $\boldsymbol{\theta}^{(m^*)} \equiv \boldsymbol{\theta}^{(m+1)}$, and END. If NO, then go to Step 3 with $m \to m+1$.

IV. Simulations and Results

This section presents experiments whose purpose is to explore and analyse the characteristics of the three CASPs variants after optimisation, highlighting their differences and strengths.

A. Datasets

1) Electricity Dataset: The Electricity dataset is used for the estimation of the medium voltage maintenance cost of the electrical line [14]. It contains data related to operational and environmental factors that affect the maintenance costs of electrical distribution networks. This data set has been widely applied in machine learning research for regression tasks, particularly for evaluating models' ability to handle cost estimation under dynamic and noisy conditions.

The data set consists of 1056 entries, with 4 inputs and 1 output. The number of trapezoidal MFs used for the simulations was configured as follows, based on the settings in [15]:

- Inputs: 3, 3, 9, and 7 MFs, respectively.
- Output: 9 MFs.

Each input and output includes a left-shoulder and a right-shoulder, both defined by two parameters. All other internal trapezoidal MFs have four parameters. Hence, the number of parameters ($\boldsymbol{\psi}$) for the inputs and output is 8, 8, 32, 24, and 32, respectively, for a total of 104 parameters. The rule base consisted of 95 rules, generated using a self-implemented rule generation approach based on the k-means clustering method [16]. The corresponding code will be made available on Code Ocean for reproducibility [17]. This method partitions the data set into clusters by grouping similar input-output data pairs. Each cluster represents a fuzzy rule, where the cluster centre is used to define the membership functions for the input variables, and the corresponding output value is associated with the consequent of the rule.

2) Laser Dataset: The Laser dataset is a benchmark for evaluating time series prediction methods [18]. It consists of time-series data generated by laser scanning technology and is widely used in time series analysis, prediction, and regression tasks.

The data set consists of 993 entries, with 4 inputs and 1 output. For simplicity, the number of trapezoidal MFs was arbitrarily set to ensure manageable rule complexity and balanced input-output configurations. i.e.,

- Inputs: 7, 7, 7, and 7 MFs, respectively.
- Output: 9 MFs.

Each input and output includes a left-shoulder and a right-shoulder, both defined by two parameters. All other internal trapezoidal MFs have four parameters. Hence, the number of parameters ($\boldsymbol{\psi}$) for the inputs and output is 24, 24, 24, 24, and 32, respectively, for a total of 128 parameters. The rule base consisted of 158 rules, generated using the same self-implemented rule generation approach as described for the Electricity data set.

B. Experimental Setup

The experiments used Mamdani-type fuzzy inference systems with centroid defuzzification. Details of this approach can be found in [1, Section 3.6.1, pp. 89–90]. The modelling and optimisation process was carried out using the FuzzyR framework [19], which leverages automatic differentiation for gradient-based optimisation [20] to update the parameters ψ, as described in Section III. The following settings were used for all experiments:

- Optimiser: Adam.
- Learning Rate: 0.03.
- Epochs: 300.
- Batch Size: 128.
- AMSGrad: Enabled.

Note that batch size refers to the number of data samples processed together before the model updates its parameters. For the Electricity data set, each epoch consists of 9 batches ($1056/128$). For the Laser data set, each epoch consists of 8 batches ($993/128$). A batch is formed by randomly sampling 128 entries from the dataset, and the data is shuffled between epochs to improve generalisation. AMSGrad is a variant of the Adam optimiser that improves stability during optimisation. It ensures a more stable learning rate by keeping past adjustments consistent, helping the model to converge more reliably.

Each variant of CASPs was evaluated in 30 runs for each data set.

C. Results

The RMSE statistics for the Electricity and Laser datasets, including the mean, standard deviation, minimum, and maximum values, are summarised in Tables I and II, respectively. The average MAE and correlation coefficient (CC) are also included. For each dataset, the statistics were calculated over 30 optimisation runs, each performed for 300 epochs using AMSGrad, providing insight into the performance and variability of the three CASPs variants. To account for potential outliers caused by random initialisations or rare optimisation anomalies, additional statistics are presented in the right-hand portion of each table. These exclude the best and worst results from the 30 runs, offering a more robust estimate of typical performance. This approach ensures that the reported results are not overly influenced by extreme cases, providing a fairer comparison of the three CASPs variants.

The initial and optimised MF plots for the best runs, determined based on testing RMSEs, are shown in Figs. 3 and S-1[4] for the Electricity and Laser datasets, respectively. The top row in each figure shows the initial MFs before optimisation, serving as the baseline configuration. The subsequent rows represent the optimised MFs under the three CASPs variants. In each row, the results of the RMSE, MAE, and CC for the training and testing sets are presented on the right, summarising the performance of the corresponding initial and optimised MFs on the left.

V. Discussion

The results, summarised in Tables I and II, together with Figs. 3 and S-1, highlight the trade-offs between the three variants of CASPs. Each variant offers a different level of strictness in managing MF constraints, balancing interpretability and optimisation performance. CASPs-Single prioritises interpretability, CASPs-Adapted achieves a balance between flexibility and performance, and CASPs-Free delivers superior RMSE performance at the expense of interpretability. The following discussion explores these trade-offs in more detail, examining the unique characteristics and implications of each variant based on the experimental results.

a) CASPs-Single: This imposes the strictest constraints, ensuring that MFs are nicely distributed and follow well-defined geometric rules. While these constraints help maintain a structured distribution of MFs, they also limit significant adjustments during optimisation compared to the other two variants of CASPs. This restriction enhances interpretability but may hinder adaptability, potentially leading to suboptimal solutions trapped in local minima. Consequently, while the RMSE values tend to be large, they are relatively consistent across runs, leading to moderate standard deviations that fall between those of CASPs-Adapted and CASPs-Free. Hence, CASPs-Single is particularly suited for applications where interpretability is prioritised over achieving the lowest RMSE.

b) CASPs-Adapted: This introduces some flexibility by relaxing certain overlapping constraints, allowing for more dynamic MF behaviour during optimisation. This additional freedom provides the potential for achieving very good performance. However, relaxation of constraints can lead to extreme cases, such as an MF overlapping with multiple others across the entire domain. As a result, the performance of CASPs-Adapted may vary significantly, with larger standard deviations observed in experimental results.

c) CASPs-Free: This imposes only minimal constraints on individual MFs, without enforcing overlap or continuity requirements. This approach is similar to global search algorithms like simulated annealing, as it allows MFs to explore a wide solution space. CASPs-Free has shown effectiveness in approximating global optimal solutions, achieving the smallest standard deviations among the three approaches. It is particularly well-suited for applications that prioritise accuracy and exploration over interpretability, such as data-driven regression or adaptive control.

More broadly, the CASPs framework serves as a general methodology that can be applied to a wide range of optimisation problems involving fuzzy sets, ensuring constraint satisfaction irrespective of the specific application domain. Although CASPs-Free provides the highest flexibility among the proposed variants, it still preserves the basic MF constraints, ensuring that the resulting MFs remain valid and preventing the generation of irregular or distorted shapes that often occur in other optimisation-based or data-driven approaches without explicit constraint enforcement.

It is also worth noting that the main advantage of the proposed CASPs framework is that all predefined constraints are always satisfied, regardless of the optimisation method used. For example, even when operations such as mutation or perturbation are involved in non-gradient-based optimisation methods (e.g., genetic algorithms or particle swarm optimisation), these operations are applied to the auxiliary

[4]The results on additional datasets exhibit similar patterns; therefore, to save space, corresponding results (e.g., Fig. S-1) are presented in the supplemental material.

TABLE I
SUMMARY OF PERFORMANCE METRICS, OBTAINED ON THE **ELECTRICITY** DATASET USING THE ADAM OPTIMISER.

Scaled

	CASPs−Single		CASPs−Adapted		CASPs−Free	
	Training	Testing	Training	Testing	Training	Testing
No. of Runs	30	30	30	30	30	30
Mean RMSE	179.56	203.31	141.64	160.85	**130.91**	**151.44**
Std. Dev.	23.60	26.33	27.99	26.40	**5.81**	**5.68**
Min RMSE	155.51	175.93	115.84	**134.31**	**115.06**	140.36
Max RMSE	238.85	289.48	243.50	271.60	**137.45**	**172.49**
Mean MAE	128.26	144.22	103.58	114.81	**86.70**	**100.30**
Mean CC	0.9956	0.9920	0.9972	0.9949	**0.9977**	**0.9955**

Scaled (Excluding Best and Worst)

	CASPs−Single		CASPs−Adapted		CASPs−Free	
	Training	Testing	Training	Testing	Training	Testing
No. of Runs	28	28	28	28	28	28
Mean RMSE	178.30	200.93	138.93	157.69	**131.24**	**151.07**
Std. Dev.	21.10	21.20	20.62	16.30	**5.03**	**5.70**
Min RMSE	155.63	175.93	117.52	**134.31**	**116.36**	140.36
Max RMSE	224.70	258.44	223.25	217.40	**137.22**	**172.49**
Mean MAE	127.61	143.14	102.40	112.92	**86.84**	**99.98**
Mean CC	0.9956	0.9922	0.9973	0.9952	**0.9976**	**0.9956**

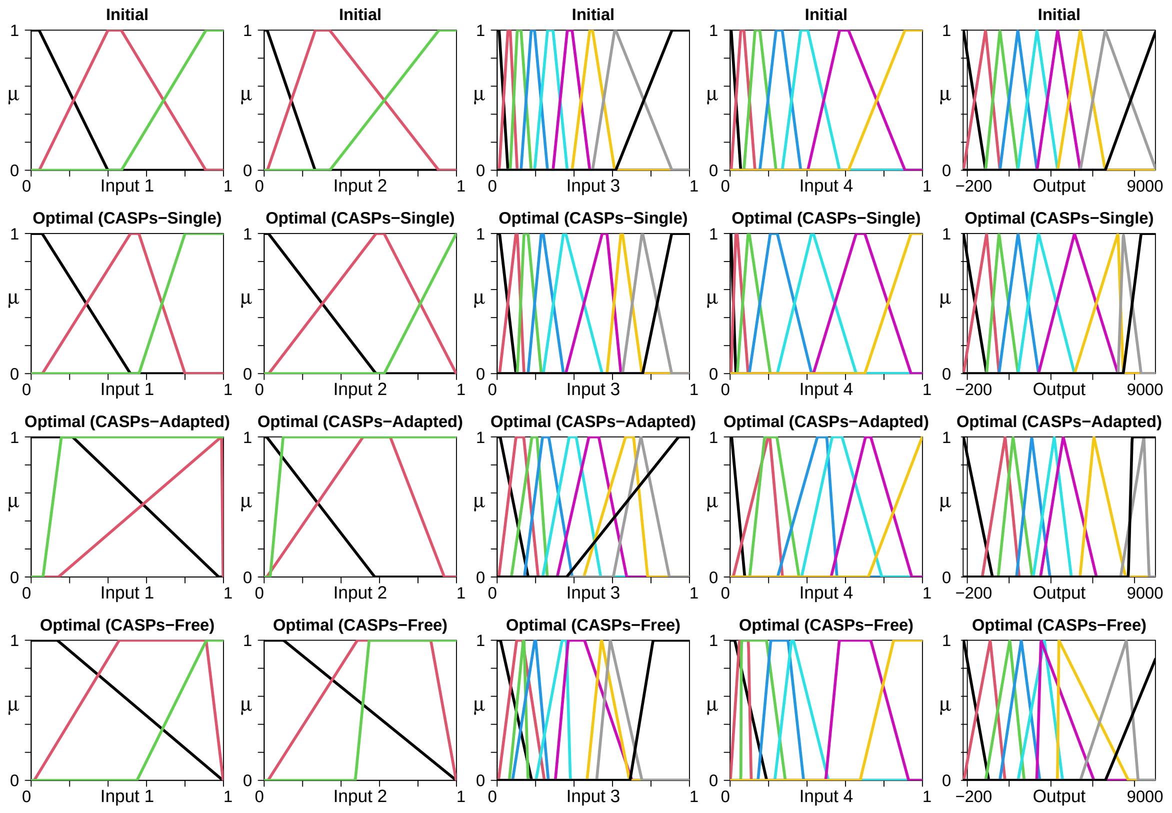
TABLE II
SUMMARY OF PERFORMANCE METRICS, OBTAINED ON THE **LASER** DATASET USING THE ADAM OPTIMISER

Scaled

	CASPs−Single		CASPs−Adapted		CASPs−Free	
	Training	Testing	Training	Testing	Training	Testing
No. of Runs	30	30	30	30	30	30
Mean RMSE	6.90	6.63	6.32	6.05	**4.99**	**4.68**
Std. Dev.	0.37	0.90	1.29	1.59	**0.14**	**0.26**
Min RMSE	6.57	6.01	5.42	4.83	**4.55**	**4.16**
Max RMSE	7.93	9.42	12.47	11.38	**5.28**	**5.62**
Mean MAE	4.30	4.59	3.87	3.94	**2.48**	**2.76**
Mean CC	0.9897	0.9874	0.9913	0.9893	**0.9946**	**0.9939**

Scaled (Excluding Best and Worst)

	CASPs−Single		CASPs−Adapted		CASPs−Free	
	Training	Testing	Training	Testing	Training	Testing
No. of Runs	28	28	28	28	28	28
Mean RMSE	6.87	6.65	6.14	5.90	**5.00**	**4.68**
Std. Dev.	0.32	0.92	0.57	1.26	**0.11**	**0.27**
Min RMSE	6.57	6.01	5.72	4.83	**4.79**	**4.16**
Max RMSE	7.71	9.42	8.14	10.96	**5.26**	**5.62**
Mean MAE	4.30	4.60	3.73	3.81	**2.48**	**2.76**
Mean CC	0.9898	0.9873	0.9919	0.9900	**0.9946**	**0.9939**



	Training	Testing
RMSE	321.68	271.86
MAE	217.04	189.39
CC	0.9897	0.9892

	Training	Testing
RMSE	177.48	175.93
MAE	128.32	129.01
CC	0.9957	0.9940

	Training	Testing
RMSE	125.44	134.31
MAE	95.73	97.16
CC	0.9978	0.9965

	Training	Testing
RMSE	116.36	140.36
MAE	79.55	94.58
CC	0.9981	0.9962

Fig. 3. The initial and optimised MF plots for the best runs, obtained on the **Electricity** dataset using the Adam optimiser.

parameters $\boldsymbol{\psi}$, which are subsequently mapped to $\boldsymbol{\theta}$. Hence, all predefined constraints on $\boldsymbol{\theta}$ remain satisfied throughout the optimisation process. To demonstrate this, additional experiments with non-gradient-based optimisers are presented in Section S-I-B. Also, although the focus of this work is not on achieving the best possible performance, a more comprehensive comparison, including both existing fuzzy and non-fuzzy methods from the literature, is provided in Section S-I-C. The results show that even with simply defined membership functions and rule bases, the proposed approach can still achieve reasonably competitive performance. These findings further support the consistent pattern previously observed that a higher degree of freedom in parameterisation potentially yields better performance.

VI. Conclusions

In this paper, we introduced the CASPs framework as a systematic and robust approach to handling MF constraints in fuzzy systems. By inherently satisfying domain, semantic, and relational constraints during optimisation, CASPs eliminate the need for manual adjustments, offering a seamless and reliable framework for MF parameter design. Importantly, CASPs provide a methodology that explicitly defines and enforces constraints, addressing a critical limitation of traditional approaches, which often rely on implicit or ad hoc implementations. This explicitness enhances reproducibility and ensures consistency across different implementations.

Through exploration of three variants, this study highlighted the trade-offs between constraint strictness, interpretability, and optimisation performance. CASPs-Single ensures well-structured MFs, prioritising interpretability, while CASPs-Free provides maximum flexibility, excelling in RMSE performance but with reduced semantic integrity. CASPs-Adapted strikes a balance between these extremes, offering a versatile solution for diverse applications.

The experimental results, validated on the Electricity and Laser datasets, demonstrated the effectiveness of CASPs in achieving consistent performance across multiple runs. These findings underscore the potential of CASPs to enhance optimisation capabilities and design flexibility, paving the way for their application in more complex fuzzy systems. The generality of CASPs is further supported by additional experiments using alternative optimisation methods and more comprehensive comparisons across multiple datasets, which are included in the supplemental material.

Future work will focus on a further investigation of different types of constraint, exploring their impact on MF design and optimisation. Additionally, CASPs will be extended to address more complex fuzzy systems, such as Type-2 fuzzy logic systems, further broadening their applicability and impact in the field.

Appendix
Proof that CASPs always satisfy all constraints

This appendix provides proof that the constraints defined in the CASPs approach are always satisfied during the parameter update and transformation processes in the core algorithms. Specifically, we focus on demonstrating that the reparameterised variables maintain the required constraints throughout optimisation.

Note that Algorithms 2, 4, and 6 only serve to transform the parameters $\boldsymbol{\theta}$ into $\boldsymbol{\psi}$. Since $\boldsymbol{\psi}$ has no restrictions, no proof is required for these algorithms. Only transformations from $\boldsymbol{\psi}$ to $\boldsymbol{\theta}$ need to ensure that all constraints are satisfied.

In the following, we present the proofs for Algorithm 1 (CASPs-Single). Proofs for Algorithms 3 (CASPs-Adapted) and 5 (CASPs-Free) follow the same reasoning and structure as Algorithm 1, and hence are omitted for brevity.

A. Proofs for Algorithm 1 (CASPs-Single)

That the constraints for CASPs-Single, as expressed in (6) – (8) are satisfied is proved.

1) Proof of $\theta_2 > \theta_1$ (Constraint T_1): In line 3 of Algorithm 1, θ_2 is computed as:

$$\theta_2 = \theta_1 + \exp(\psi_2) + \exp(\psi_3)$$

Since exponential terms are always positive, we have $\exp(\psi_2) > 0$ and $\exp(\psi_3) > 0$, and hence

$$\theta_2 = \theta_1 + \exp(\psi_2) + \exp(\psi_3) > \theta_1.$$

Therefore, the constraint $T_1 (\theta_2 > \theta_1)$ holds.

2) Proof of $\theta_{4j-6} > \theta_{4j-5}$ (Constraint $T_j\#2$) in (7)*:* When $j = 2$, this corresponds to proving $\theta_2 > \theta_3$. From line 3 of Algorithm 1, we have

$$\theta_2 = \theta_1 + \exp(\psi_2) + \exp(\psi_3),$$

and from line 6,

$$\theta_3 = \theta_1 + \exp(\psi_2).$$

Hence,

$$\theta_2 = \theta_3 + \exp(\psi_3) > \theta_3.$$

Therefore, the constraint $T_j\#2$ holds for $j = 2$.

For $j > 2$, let j be replaced by $j-1$ in line 9 of Algorithm 1, to obtain

$$\theta_{4(j-1)-2} = \theta_{4(j-1)-3} + \exp(\psi_{4(j-1)-2}) + \exp(\psi_{4(j-1)-1}).$$

Simplifying the indices gives

$$\theta_{4j-6} = \theta_{4j-7} + \exp(\psi_{4j-6}) + \exp(\psi_{4j-5}).$$

From line 6 of Algorithm 1, we also have

$$\theta_{4j-5} = \theta_{4j-7} + \exp(\psi_{4j-6}).$$

Therefore,

$$\theta_{4j-6} = \theta_{4j-5} + \exp(\psi_{4j-5}) > \theta_{4j-5}.$$

Thus, the constraint $T_j\#2$ $(\theta_{4j-6} > \theta_{4j-5})$ holds for all $j \geq 2$.

3) Proof of $\theta_{4j-5} > \theta_{4j-7}$ (Constraint $T_j\#1$) and Others in (7) *and* (8)*:* In line 6 of Algorithm 1, we have:

$$\theta_{4j-5} = \theta_{4j-7} + \exp(\psi_{4j-6})$$

Since the exponential term $\exp(\psi_{4j-6})$ is always positive, it can be concluded that:

$$\theta_{4j-5} = \theta_{4j-7} + \exp(\psi_{4j-6}) > \theta_{4j-7}.$$

Therefore, the constraint $T_j\#1$ ($\theta_{4j-5} > \theta_{4j-7}$) holds.

The proofs for the remaining constraints defined in T_j and T_Q follow the same structure. Each involves adding/subtracting one or more strictly positive exponential term to the previous parameter, ensuring that the constraints are satisfied. Detailed proofs are left to the reader. Note that the total number of parameters in Algorithm 1 is given by $n = 4(Q-1)$. Therefore, in the constraints defined under T_Q, terms such as θ_{4Q-5}, θ_{4Q-7}, and θ_{4Q-4} correspond to θ_{n-1}, θ_{n-3}, and θ_n, respectively.

4) Proof of Domain Constraints: In Steps 1.14–1.16, the sigmoid function $\sigma(\theta_i)$ is applied to all parameters θ_i, i.e.:

$$\theta_i \leftarrow \sigma(\theta_i).$$

The sigmoid function $\sigma(x)$ in (9) has the following property:

$$0 < \sigma(x) < 1, \quad \forall x \in \mathbb{R}.$$

By applying the sigmoid function to each parameter θ_i, it is guaranteed that all θ_i are mapped into the range $(0, 1)$. Therefore, the domain constraints are always satisfied as a result of the sigmoid transformation.

In summary, the proofs are mainly based on the positive nature of the exponential function and the properties of the sigmoid function. These properties ensure that all constraints—whether for ordering, overlaps, or domains—are always satisfied throughout the parameter update and transformation processes.

References

[1] J. M. Mendel, *Explainable Uncertain Rule-Based Fuzzy Systems*, 3rd ed. Springer Cham, 2024.

[2] K. Pal, R. K. Mudi, and N. R. Pal, "A new scheme for fuzzy rule-based system identification and its application to self-tuning fuzzy controllers," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 32, no. 4, pp. 470–482, 2002.

[3] A. Sebastião, C. Lucena, L. Palma, A. Cardoso, and P. Gil, "Optimal tuning of scaling factors and membership functions for mamdani type PID fuzzy controllers," in *Proceedings of International Conference on Control, Automation and Robotics*, 2015, pp. 92–96.

[4] M. B. Trabia and W. E. McCarthy, "Design of fuzzy logic controllers for optimal performance," *Journal of Intelligent and Fuzzy Systems*, vol. 6, no. 4, pp. 459–470, 1998.

[5] S. E. Woodward and D. P. Garg, "A numerical optimization approach for tuning fuzzy logic controllers," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 4, pp. 565–569, 1999.

[6] E. H. Ruspini, "A new approach to clustering," *Information and Control*, vol. 15, no. 1, pp. 22–32, 1969.

[7] O. Cordón, F. Herrera, L. Magdalena, and P. Villar, "A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base," *Information Sciences*, vol. 136, no. 1-4, pp. 85–107, 2001.

[8] D. A. N. Simon, "Sum normal optimization of fuzzy membership functions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 04, pp. 363–384, 2002.

[9] J. V. de Oliveira, "Semantic constraints for membership function optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 29, no. 1, pp. 128–138, 1999.

[10] P. P. Bonissone, X. Hu, and R. Subbu, "A Systematic PHM Approach for Anomaly Resolution: A Hybrid Neural Fuzzy System for Model Construction," in *Proceedings of Annual Conference of the PHM Society*, 2009, pp. 1–13.

[11] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[12] R. Alcalá, J. Casillas, O. Cordón, F. Herrera, and I. Zwir, "Techniques for learning and tuning fuzzy rule-based systems for linguistic modeling and their application," *Knowledge Engineering Systems, Techniques and Applications*, vol. 3, pp. 889–941, 1999.

[13] D. Wu and J. M. Mendel, "On the Continuity of Type-1 and Interval Type-2 Fuzzy Logic Systems," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 179–192, 2011.

[14] O. Cordón, F. Herrera, and L. Sánchez, "Solving Electrical Distribution Problems Using Hybrid Evolutionary Data Analysis Techniques," *Applied Intelligence*, vol. 10, no. 1, pp. 5–24, 1999.

[15] O. Cordón, F. Herrera, and P. Villar, "Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 667–674, 2001.

[16] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[17] C. Chen, J. M. Mendel, and J. M. Garibaldi, "Source Code for 'Constraints Always Satisfied Parameters for Fuzzy Sets'," https://www.codeocean.com/, 2025.

[18] A. S. Weigend and N. A. Gershenfeld, *Time series prediction: Forecasting the future and understanding the past.* Santa Fe: Addison-Wesley, 1993.

[19] C. Chen, T. R. Razak, and J. M. Garibaldi, "FuzzyR: An Extended Fuzzy Logic Toolbox for the R Programming Language," in *Proceedings of IEEE International Conference on Fuzzy Systems*, Glasgow, UK, 2020, pp. 1–8.

[20] C. Chen, C. Wagner, and J. M. Garibaldi, "Gradient-based Fuzzy System Optimisation via Automatic Differentiation – FuzzyR as a Use Case," *arXiv:2403.12308 [cs.AI]*, 2024.