A Non-Iterative Quantum Computation for Karnik-Mendel Algorithms

Amir Pourabdollah
School of Science and Technology
Nottingham Trent University
Nottingham, UK
amir.pourabdollah@ntu.ac.uk

Jerry M. Mendel
Life Fellow, IEEE
University of Southern California
Los Angeles, CA 90089-2564, USA
jmmprof@me.com

***Abstract*—Karnik-Mendel method (KM) is widely known for type-reduction of interval type-2 fuzzy sets. The original and enhanced solutions of KM use various forms of iteration to converge onto the result, leading to different levels of computational complexities. Based on the algorithmic advancements in quantum computing, this paper proposes a non-iterative quantum computing solution to KM. We map the KM problem to the problem of minimising an objective function in the form of a binary constrained quadratic model (CQM) which can then be solved in a single run using "quantum annealing". The algorithms used in each step, together with a numerical example and the results from real quantum computer are provided. Quantum technology permitting in the future, the proposed solution can potentially remove the known defuzzification bottleneck of designing type-2 fuzzy systems.**

***Index Terms*—quantum computing, Karnik-Mendel, type-2 fuzzy.**

I. Introduction

In designing Interval Type-2 Fuzzy Sets (IT2-FSs) [1], Karnik-Mendel (KM) method [2] is well-known for type-reduction and defuzzification. This is implemented as various iterative algorithms with proven convergence, and there are some modifications and enhancements of the original KM method in order to reduce its computational complexities (e.g., in [3]–[5]), all of which being iterative and/or approximative.

In this paper we propose a fundamentally different, non-classical approach to implement the KM type-reduction method, which does not involve any iteration, but rather by solving it using a class of quantum computation known as *quantum annealing*.

Thanks to its built-in massive parallelism, quantum computing is increasingly becoming a useful non-classical approach for the improvement of computational intelligence algorithms. However, only in recent years has quantum computation started to be limitedly applied in the context of fuzzy logic (e.g., [6]–[10]). The main direction of research has been on the development of quantum fuzzy inference systems, so that the complex fuzzy rule-based systems in the future can take advantage of quantum computing for speeding up the process. This would be especially significant for those with large rule-bases or for the higher-order systems such as type-2 and non-singleton. However, to the best of our knowledge, no previous work has focused on quantum algorithms for type-2 systems.

We aim at formalising the KM method as a quantum-ready problem, and developing/testing an algorithm for solving it with a real quantum annealer. The proposed algorithm has a potential to addressing a computationally intensive bottleneck [11], [12] towards wide-spreading the development of type-2 fuzzy systems in the future.

The limiting factor, as the case for many other quantum computing algorithms, is the scalability of the current quantum hardware technology. This includes the hardware limitations in maintaining large quantum computers, and the existence of noise and imperfections in such physical systems. It is therefore important to notice that the majority of current research works in quantum algorithms in different domains, including this paper, are still at the proof-of-concept level, rather than showcasing an advantage in real-world applications. Therefore, in this paper, no space/time complexity measure is given to compare our proposed algorithm with the previous approaches.

In the following sections, we first provide some background information. Then, we will review the underlying concepts in both KM method and quantum annealing. We will then provide the details of our algorithm, followed by testing the algorithm for a numerical example.

II. Background

In quantum computing [13], the underlying idea is that instead of considering the binary variables as classical 0/1 bits, they are represented by the *superpositioned* and *entangled* quantum bits (qubits) in which each qubit can simultaneously take values 0 or 1, with different probabilities. This theoretically allows massive computation parallelism, particularly useful for solving problems with large search spaces.

Through a quantum computing algorithm, the state of an initiated system of qubits is evolved through time. By tuning the physical state of the qubits, the system is evolved in a way that the final measurement probabilities of the qubits collectively represent a desired computation [14]. Currently, two quantum computing paradigms are available: the first, being the mainstream, is the quantum circuital model [15], and the second is the quantum annealing (or adiabatic) model [16].

In the circuit model, the qubits are evolved through a sequence of quantum 'gates', i.e., unitary linear operations [15]. Due to its design flexibility, the circuit model can undertake a wider range of computations, including those required in fuzzy logic. For example, a Quantum Fuzzy Inference Engine (QFIE) has been developed in [6], [7] based on the quantum circuit model, which is able to achieve an exponential speed-up in computing fuzzy rules over classical methods.

On the other hand, quantum annealing is a more simplified and more scalable approach [17], [18], but its limitation is that it is specialised for optimising only a certain form of objective functions, i.e., being in quadratic polynomials with binary variables (Binary Quadratic Model - BQM [19], [20]).

Briefly, through the quantum annealing process, the qubits' system evolve adiabatically in a controlled way that the observed binary values of the measured qubits at the final state (minimum energy) correspond to the optimum solution to the BQM. This is due to the fact that the linear terms are in fact the qubits' local field magnitudes, and the quadratic terms are their coupling strengths, roughly corresponding to the qubits' superposition and entanglement attributes.

The primary challenge of applying quantum annealing for an optimisation problem is converting the problem to the required form of BQM. This challenge for fuzzy logic system would mean mapping different required fuzzy logic operations into BQM optimisation problems. In our previous works, we started to make this link between fuzzy logic systems and quantum annealing: In [8] we introduced BQM representations of the fuzzy operators, such as fuzzy union, fuzzy intersection, alpha-cut and maximum. This is then followed in [9] by implementing the centroid defuzzification as a quantum annealing algorithm. Finally, in [10] it is shown how a whole fuzzy rule-based system with Mamdani inference can be implemented on quantum annealers.

For the case of the KM algorithm, Kumbasadar in [21] has already formulated the KM problem to a binary linear fractional programming (LFP), then to linear programming (LP), then solved it using iterative processes. For our purpose in this paper, we will partly reuse and simplify his results, in order to arrive in a quantum-ready BQM.

III. Underlying Concepts

In this section, before introducing the quantum algorithm solution for the KM problem, it is necessary to mathematically introduce the basic underpinning concepts, in terms of two problems (A) and (B). In the next section, we will link these concepts to develop our solution.

A. Karnik-Mendel (KM) Defuzzification Method

Without going into details, we directly provide the core of the KM problem for type-reduction of an IT2-FS [5]. Notably, the equivalent problems also appear in other subject domains, as reviewed in [3], [22].

Given an IT2-FS defined by its Lower Membership Function (LMF) and its Upper Membership Function (UMF), the footprint of uncertainty between LMF and UMF contains many *embedded* type-1 fuzzy sets. If all of these embedded sets could be defuzzified, say by centroid defuzzification, two extreme sets would result in the minimum and maximum centroids. The average of these two centroids is considered as the centroid of the IT2-FS.

The key point in solving the problem is that the two extreme sets are either on LMF or on UMF with a single switch-over point, as proven in [23]. Therefore, the general aim in KM algorithms is to find the two switch-over points along the x-axis.

Problem (A): For a fuzzy set discretised in n intervals along the x-axis ($x_1 ... x_n$), find an integer k that minimises/maximises $f(k)$ defined as:

$$f(k) = \frac{\sum_{i=1}^{k} x_i u_i + \sum_{i=k+1}^{n} x_i l_i}{\sum_{i=1}^{k} u_i + \sum_{i=k+1}^{n} l_i} \tag{1}$$

where u_i is the ith upper-membership grade (UMF) and l_i is the ith lower-membership grade (LMF).

We focus on the minimising problem and will see that the maximising problem is quite similar. The aim is to convert the above problem from its fractional form to a binary quadratic form, and to minimise it.

B. Binary Quadratic Model (BQM) Problem

The adiabatic model of quantum computing is able to solve the optimisation problems expressed in the form BQM minimisation, as:

Problem (B): Given n binary (0/1) bits $y_1, ..., y_n$, find assignments of the bits that minimise an objective function $f(y)$ defined as:

$$f(y) = \sum_{i=1}^{n} (p_i y_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (p_{ij} y_i y_j) \tag{2}$$

where p_i and p_{ij} are configurable (linear and quadratic) coefficients. More details can be found in [19].

C. Constrained Quadratic Model (CQM) Problem

Many real-world optimisation problems can be converted to optimising an objective function that must also satisfy a set of constraints. In these cases, it is common to convert the constraint(s) to extra linear and/or quadratic terms (known as penalty terms), that are weightedly added to the original objective function. The result would be considered as a single unconstrained objective function.

Similarly, the unconstrained problem (B) might be the result of adding some penalty terms to an original constrained BQM (known as Constrained Quadratic Model - CQM). Therefore, it is possible to redefine problem (B) as:

Problem (B)-redefined as CQM: Given n binary (0/1) bits $y_1, ..., y_n$, find assignments of the bits that minimise an objective function $f(y)$ defined as (2) *having a set of constraints, each in the form of:*

$$\sum_{i=1}^{n} (q_i y_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (q_{ij} y_i y_j) \quad operator \quad c \tag{3}$$

where p_i, p_{ij}, q_i and q_{ij} are configurable linear/quadratic coefficients, $operator$ is one of $\{" < ", " > ", " <= ", " >= ", " = "\}$, and c is a constant.

IV. The Algorithm

In this section, we first show how to reformulate the problem A (KM) into problem B (CQM) defined in the previous section, then will explain how to find the answers of the formulated CQM on a quantum annealer.

A. Converting KM to Binary Linear Fractional Form

For this conversion, we follow the method proposed in [21]. The notations, however, are slightly different. Consider Problem A (1). First, let us define binary values y_i that act as a binary switch between UMF and LMF:

$$y_i = \begin{cases} 1 & \textit{if } i < k \\ 0 & \textit{otherwise} \end{cases} \qquad y_i \in \{0, 1\}, \quad i \in \{0, ..., n\} \tag{4}$$

We define a binary matrix Y in the form of:

$$Y = [y_1, ..., y_n]^T = [1\ 1\ ...\ 1\ 0\ 0\ ...\ 0]^T \tag{5}$$

As shown in [5], the KM problem can be reformulated from minimising (1) to minimising $f(Y)$ in:

$$f(Y) = \frac{\sum_{i=1}^{n} x_i u_i y_i + \sum_{i=1}^{n} x_i l_i (1 - y_i)}{\sum_{i=1}^{n} u_i y_i + \sum_{i=1}^{n} l_i (1 - y_i)} \tag{6}$$

or:

$$f(Y) = \frac{\sum_{i=1}^{n} x_i y_i (u_i - l_i) + \sum_{i=1}^{n} x_i l_i}{\sum_{i=1}^{n} y_i (u_i - l_i) + \sum_{i=1}^{n} l_i} \tag{7}$$

If we define:

$$X = [x_i; i = 1...n]^T \tag{8}$$

$$U = [u_i; i = 1...n] \tag{9}$$

$$L = [l_i; i = 1...n] \tag{10}$$

$$A = [x_i(u_i - l_i); i = 1...n] \tag{11}$$

$$B = [u_i - l_i; i = 1...n] \tag{12}$$

$$\alpha = LX \quad , \quad \beta = \sum_{i=1}^{n} l_i \tag{13}$$

then $f(Y)$ can be rewritten as:

$$f(Y) = \frac{AY + \alpha}{BY + \beta} \tag{14}$$

Minimising $f(Y)$ in (14) is in the form of an unconstrained Linear Fractional Programming (LFP) problem with binary variable.

B. Converting LFP to LP

Using the Charnes Cooper Transformation (introduced in [24], also formulated in [25] and [21]), a constrained LFP problem can be transformed to an equivalent Linear Programming (LP) problem with transformed variables and constraints. Without reviewing the transformation details, the LFP problem (14) can be transformed as follows:

The variable matrix Y is transformed to a new variable matrix $Z = [z_1, ..., z_n]^T$, so that the transformed problem is now finding Z that can minimise a linear function $g(Z)$, defined as:

$$g(Z) = CZ + \gamma \tag{15}$$

where:

$$C = A - \frac{\alpha}{\beta} B \tag{16}$$

$$\gamma = \frac{\alpha}{\beta} \tag{17}$$

and;

$$Z = \frac{1}{BY + \beta} Y; \quad or \quad Y = \frac{\beta}{1 - BZ} Z \tag{18}$$

Notice that the constant γ in (15) can be ignored for minimising of the objective function. Therefore, the problem is reduced to a linear programming in the form of:

$$g(Z) = CZ \tag{19}$$

Fortunately, once the LP is solved, we do not need to transform any result back to LFP. The reason being that the ultimate goal is to find the switch-over position, which is, as will be shown in the next subsection, same for the LP and the LFP.

C. Defining Constraints

Since the LFP (14) has no explicit constraint, the LFP transformation also does not provide an explicit constraint. However, the Charnes Cooper Transformation is for general variable type and does not consider the implications of binary variables. Importantly here, (18) implies a constraint: Y is binary but Z is not, therefore the only acceptable Z is the one that corresponds to a binary Y (according to (18)).

It is also noticeable that since Y is in a specific form of $[1\ 1\ ...\ 1\ 0\ 0\ ...\ 0]^T$ with a single switch-over position, Z becomes in the following specific form with the same single switch-over position:

$$Z = zY = [z\ z\ ...\ z\ 0\ 0\ ...\ 0]^T \tag{20}$$

where z is a single non-negative decimal number, defined in (18), as:

$$z = \frac{1}{BY + \beta} \tag{21}$$

Accordingly, we add a constraint that implies the strong dependency between Z and Y. Based on (21), this constraint can be formulated as:

$$z(BY + \beta) - 1 = 0 \tag{22}$$

Practically, the above equality cannot always hold. This is due to different imperfections such as discretisation errors, errors in decimal number operations, and the noise existing on the quantum annealer hardware. Therefore, we define a precision threshold p that is to be tuned when the algorithm is tested. The constraint is then to be redefined as:

$$z(BY + \beta) - 1 \leq p \tag{23}$$

Finally, there is another constraint, for which there should be a single switch-over point in Z. However, this constraint is automatically satisfied when the objective function is minimised. It is shown in [23] that minimising/maximising the centroid necessarily guarantees a single switch-over point between UMF and LMF. Therefore, we do not need to separately formulate this constraint.

D. Reformulating the KM Problem

Although Z is the variable for the LP problem (15), the existing constraint makes a circular relationship between Y of the LFP and Z of the LP. To get around this, we merge Y and Z to become the variable vector of a new merged LP, so that both Y and Z can be found simultaneously through a single CQM optimisation algorithm.

In making the marged variable vector for the new LP problem, we notice that Z members are either z or 0, and that the zero members of Z match those of Y (see the previous subsection). Therefore,

$$z_i = zy_i \quad ; \quad i = 1...n \tag{24}$$

In other words, all information needed from Z in the new problem is a single decimal variable z, and that the new variable vector can be reduced from $[y_1, ..., y_n, z_1, ..., z_n]$ to a more simplified form of $[y_1, ..., y_n, z]$.

However, this vector consists of a mix of binary and decimal variables. If the quantum annealer works on binary variables only, it will be necessary to expand z to its binary format with m bits, in which case the variable vector will have $n+m$ binary values. Some quantum annealers, however, provide libraries for internal variable type conversion, so that the decimal variable z can still be kept as a single variable in the new variable vector.

Given (25), the new LP problem can now be defined as a CQM with mixed variable types, and with the following quadratic polynomial objective function:

$$g(Z) = CZ = C.(zY) = z.(CY) = z(c_1y_1 + ... + c_ny_n) \tag{25}$$

It is important to note that since z is dependent on Y, we do not consider z as a constant in (25) - otherwise it would have been ignored for minimising $g(Z)$.

Finally, given (23), the new CQM has a quadratic polynomial constraint, defined as:

$$z(b_1y_1 + ... + b_ny_n + \beta) - 1 \leq p \tag{26}$$

E. Summary

In summary, our non-iterative algorithm is proposed as:

Step 1	Calculate B, C, α, β according to (13,14,17)
Step 2	Define a variable vector in the form of $[y_1, y_2, ..., y_n, z]$ with binary y_i and decimal z.
Step 3	Define a CQM objective function in the form of: $g(Z) = z(c_1y_1 + ... + c_ny_n)$
Step 4	Define the CQM's constraint in the form of: $z(b_1y_1 + ... + b_ny_n + \beta) - 1 \leq p$
Step 5	Run quantum annealing for the defined CQM
Step 6	Find the 0/1 switch-over position in the result
Step 7	Replace $g(Z)$ with $-g(Z)$ and rerun Steps 5-6

V. Implementation and Test

The numerical example that we consider is an IT2-FS with a non-symmetrical Gaussian UMF and a non-symmetrical triangular LMF, as defined below. This example is repeatedly used in the literature (e.g. in [23], [26], [27]).

$$UMF = \begin{cases} exp\left(\frac{1}{2}\left(\frac{x-2}{5}\right)^2\right) & x \in [-5, 7] \\ exp\left(\frac{1}{2}\left(\frac{x-9}{1.75}\right)^2\right) & x \in [8, 14] \end{cases} \tag{27}$$

$$LMF = \begin{cases} \frac{0.6(x+4)}{19} & x \in [-5, 2] \\ \frac{0.4(14-x)}{19} & x \in [3, 14] \end{cases} \tag{28}$$

in which $-5 \leq x \leq 14$ with n=20 discretisation levels. The graphs of the defined UMF and LMF are shown in Fig. 1.

It can be numerically shown that the two switch-over points between the UMF and the LMF are between $x = 0$ and $x = 1$ (for the minimum centroid) and between $x = 7$ and $x = 8$ (for the maximum centroid). The switch-over points are also shown in Fig. 1. The aim is to develop a non-iterative quantum algorithm to determine these two points.

The implementation is based on D-Wave System (https://docs.dwavesys.com), a cloud-based real quantum computing platform. D-Wave also provides some Python libraries for programming using web-based and desktop IDEs that connect to the same platform.

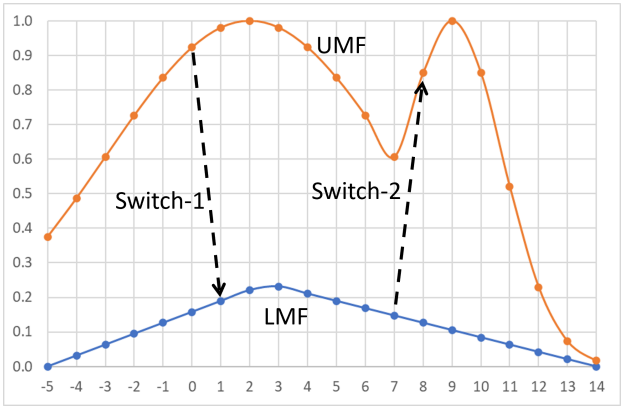

Fig. 1. LMF, UMF and the two switch-over points of the numerical example

Listing 1. Python program for quantum KM algorithm

```
from dimod import Integer, ConstrainedQuadraticModel
from dwave.system import LeapHybridCQMSampler

# x and LMF/UMF grades in an example IT2 FLS
x = [-5.0000 ,-4.0000 ,-3.0000 ,-2.0000 ,-1.0000 , 0.0000 , 1.0000 , 2.0000 , 3.0000 , 4.0000,
    5.0000 , 6.0000 , 7.0000 , 8.0000 , 9.0000 ,10.0000 ,11.0000 ,12.0000 ,13.0000 ,14.0000]
lmf= [ 0.0000 , 0.0316 , 0.0632 , 0.0947 , 0.1263 , 0.1579 , 0.1895 , 0.2211 , 0.2316 , 0.2105,
    0.1895 , 0.1684 , 0.1474 , 0.1263 , 0.1053 , 0.0842 , 0.0632 , 0.0421 , 0.0211 , 0.0000]
umf= [ 0.3753 , 0.4868 , 0.6065 , 0.7261 , 0.8353 , 0.9231 , 0.9802 , 1.0000 , 0.9802 , 0.9231,
    0.8353 , 0.7261 , 0.6065 , 0.8494 , 1.0000 , 0.8494 , 0.5205 , 0.2301 , 0.0734 , 0.0169]
n=len(x)
prec=10000

# Step 1: Calculate the required values/vectors - A, B, C, alpha, beta
beta =sum(lmf)
alpha=sum(x[i]*lmf[i] for i in range(n))
b  =[umf[i]-lmf[i] for i in range(n)]
a  =[x[i]*b[i] for i in range(n)]
c  =[a[i]-alpha/beta*b[i] for i in range(n)]
c  =[c[i]/prec for i in range(n)]  #to compensate integer conversion

# Step 2: Initialise a variable vector consisting Y (binary) and z (integer)
y=[Integer('y'+ str(i).zfill(2), upper_bound=1) for i in range(n)]
z=Integer('z', upper_bound=prec)

# Step 3: Define a CQM objective function
cqm=ConstrainedQuadraticModel()
for i in range(n): cqm.set_objective(cqm.objective+z*c[i]*y[i]) #to get the maximum, change +z to -z

# Step 4: Define a constraint to relate integer z to binary Y
lhs=[('z', 'y'+str(i).zfill(2), b[i]) for i in range(n)]
lhs.append(('z', beta))
cqm.add_constraint(lhs, '<=', rhs=prec)

# Step 5: Run the CQM on the cloud-based quantum annealer to find the solution of the variable vector
sampleset = LeapHybridCQMSampler().sample_cqm(cqm)

# Step 6: Print the first feasible result (minimum energy) and find the single switch-over point
answer=sampleset.filter((lambda d: d.is_feasible)).first[0]
for key, value in answer.items(): print(f"{key:3}{int(value):10}")
for i in range(1,n):
  if answer.get('y'+str(i).zfill(2))!=answer.get('y'+str(i-1).zfill(2)):
    print ('Switch-over found at position',i)
```

The steps listed in section IV-E are implemented as the Python program shown in Listing 1. In reference to this Listing, after importing the required D-Wave libraries for CQM, the vectors X, LMF and UMF are enumerated in the code - based on (27) and (28), till 4 decimal points. In Step 1, the vectors and scalar values for A, B, C, α and β are calculated, according to (11), (12), (17), (13). Also, a constant for precision is defined for the following reason:

The DWave's CQM library currently supports binary and integer variables but is yet to support decimal variables. To keep z in the variable vector, given that its value is in $[0, 1]$, we scale up z to an integer between 0 and a $prec$ factor (here, $prec$=10000 for 4 decimal point precision). To compensate this, we scale-down the values in the C vector by the same factor in order to cancel each other out in the objective function $g(Z) = CZ$. The imported CQM library supports floating point precision, therefore scaling down C should not have a significant effect on the results. $prec$ is also used for defining the constraint between Y and z (see section IV-C).

In Step 2, a variable vector is defined consisting of n=20 qubits for the binary variables $y_1...y_{20}$ and some additional qubits for the integer variable z (the number of which would be internally set based on the upper bound of z). In Step 3, the CQM objective function $c_1y_1z+...+c_{20}y_{20}z$ is encoded in the CQM library's required format. Also in Step 4, the constraint $b_1y_1z + ... + b_{20}y_{20}z + \beta z - 1 \leq p$ is encoded (see (23)). The left hand side of the constraint is a quadratic polynomial in itself. Fortunately, the D-Wave's CQM library internally converts the constraints to some linear and quadratic penalty terms with adjusted weights, all to be added to the objective function in order to make a final unconstrained BQM.

Having all CQM attributes set, Step 5 is where the program calls the quantum computing service on the cloud with the CQM object. The cloud returns the answer through a 'sample set', which is the final states of the variable vector after quantum annealing. The first answer would be the state with the lowest energy - corresponding to the minimum value of the objective function that satisfies the given constraint.

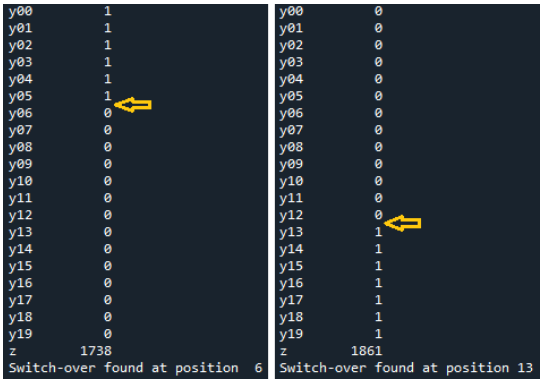

Fig. 2. The outputs of the program in Listing 1: The switch over point for minimising (left) and maximising (right) the objective function are determined.

Finally, the returned solution is to be analysed. The first 20 items of the solution are binary (0/1) values which should have a single switch-over point, the position of which is determined by the program in the final step.

If the sign of the objective function is reversed in step 3, the algorithm would find the maximum value of the objective function, and another switch-over point is determined. The outputs of the program for both tasks are shown in Fig. 2.

Fig. 2 shows that the switch-over points are at positions 6 (of the UMF) and 13 (of the LMF) - matching to the numerical calculation. Using (1), it can be shown that the left-most and the right-most centroids of the IT2-FS are at x =0.2609 and x=7.1663, and that their average (x=3.7136) would be the centroid of the IT2-FS.

VI. Conclusion and Future Direction

We approached the KM problem from a novel non-classical angle, and developed a quantum annealing algorithm to solve it non-iteratively. The main advantage of our approach in comparison to all other KM implementation is the non-iterative nature of the algorithm, thanks to the built-in parallelism of quantum computing. The proposed algorithm includes a number of steps to map the KM problem to a quantum-ready objective function. A quantum cloud service (D-Wave Systems) has then been used to test the algorithm for defuzzifying IT2-FSs.

The current limitation is the scalability of the algorithm to finer discretisation intervals, due to the limited number of controllable low-noise qubits in the current quantum annealers. However, this technological issue for the current era of Noisy Intermediate-Scale Quantum (NISQ), which is yet to be resolved in the future, should not stop the research on developing potentially-advantageous algorithms for the future real-world applications. Accordingly, a future direction of this research is to develop quantum algorithms for other complex elements of developing type-2 and non-singleton fuzzy rule-based systems. This particularly includes defuzzification/type-reduction of general-type-2 fuzzy systems (GTFSs) - as suggested in [3].

References

[1] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Transactions on Fuzzy systems*, vol. 8, no. 5, pp. 535–550, 2000.

[2] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *information SCiences*, vol. 132, no. 1-4, pp. 195–220, 2001.

[3] J. M. Mendel, "On km algorithms for solving type-2 fuzzy set problems," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 3, pp. 426–446, 2013.

[4] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: overview and comparisons," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2012.

[5] J. M. Mendel, *Uncertain rule-based fuzzy systems: introduction and new directions*, 2nd ed. Springer, Cham, Switzerland, 2017.

[6] G. Acampora, R. Schiattarella, and A. Vitiello, "On the implementation of fuzzy inference engines on quantum computers," *IEEE Transactions on Fuzzy Systems*, 2022.

[7] G. Acampora, A. Massa, R. Schiattarella, and A. Vitiello, "Distributing fuzzy inference engines on quantum computers," in *2023 IEEE International Conference on Fuzzy Systems (FUZZ)*. IEEE, 2023, pp. 1–6.

[8] A. Pourabdollah, G. Acampora, and R. Schiattarella, "Fuzzy logic on quantum annealers," *IEEE Transactions on Fuzzy Systems*, 2021.

[9] ——, "Implementing defuzzification operators on quantum annealers," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2022, pp. 1–6.

[10] A. Pourabdollah, C. Wilmott, R. Schiattarella, and G. Acampora, "Fuzzy inference on quantum annealers," in *2023 IEEE International Conference on Fuzzy Systems (FUZZ)*. IEEE, 2023, pp. 1–6.

[11] S. Greenfield, "Type-2 fuzzy logic: Circumventing the defuzzification bottleneck," 2012.

[12] S. Greenfield and F. Chiclana, "Accuracy and complexity evaluation of defuzzification strategies for the discretised interval type-2 fuzzy set," *International Journal of Approximate Reasoning*, vol. 54, no. 8, pp. 1013–1033, 2013.

[13] D. McMahon, *Quantum computing explained*. John Wiley & Sons, 2007.

[14] A. O. Pittenger, *An introduction to quantum computing algorithms*. Springer Science & Business Media, 2012, vol. 19.

[15] F. Tacchino, A. Chiesa, S. Carretta, and D. Gerace, "Quantum computers as universal quantum simulators: state-of-the-art and perspectives," *Advanced Quantum Technologies*, vol. 3, no. 3, p. 1900052, 2020.

[16] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018.

[17] D. Aharonov, W. van Dam, J. Kempe, and Landau, "Adiabatic quantum computation is equivalent to standard quantum computation," in *IEEE Symp. on Found. of Comp. Sci.* IEEE, 2004, pp. 42–51.

[18] A. Das and B. K. Chakrabarti, *Quantum annealing related optimization methods*. Springer Science & Business Media, 2005, vol. 679.

[19] M. Lewis and F. Glover, "Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis," *Networks*, vol. 70, no. 2, pp. 79–97, 2017.

[20] C. C. McGeoch, "Adiabatic quantum computation and quantum annealing: Theory and practice," *Synthesis Lectures on Quantum Computing*, vol. 5, no. 2, pp. 1–93, 2014.

[21] T. Kumbasar, "Revisiting karnik–mendel algorithms in the framework of linear fractional programming," *International Journal of Approximate Reasoning*, vol. 82, pp. 1–21, 2017.

[22] D. Wu and J. M. Mendel, "Enhanced karnik–mendel algorithms," *IEEE transactions on fuzzy systems*, vol. 17, no. 4, pp. 923–934, 2008.

[23] J. M. Mendel and F. Liu, "Super-exponential convergence of the karnik–mendel algorithms for computing the centroid of an interval type-2 fuzzy set," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 2, pp. 309–320, 2007.

[24] A. Charnes and W. W. Cooper, "Programming with linear fractional functionals," *Naval Research logistics quarterly*, vol. 9, no. 3-4, pp. 181–186, 1962.

[25] M. B. Hasan and S. Acharjee, "Solving lfp by converting it into a single lp," 2010.

[26] X. Liu and J. M. Mendel, "Connect karnik-mendel algorithms to root-finding for computing the centroid of an interval type-2 fuzzy set," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 4, pp. 652–665, 2011.

[27] J. M. Mendel and X. Liu, "Simplified interval type-2 fuzzy logic systems," *IEEE transactions on fuzzy systems*, vol. 21, no. 6, pp. 1056–1069, 2013.