An Effective Approach for Genetic-Fuzzy Mining Using the Graphics Processing Unit

Chun-Hao Chen[1], **Yu-Qi Huang**[2] and Tzung-Pei Hong[2, 3]

[1]Department of Information and Finance Management, National Taipei University of Technology, Taipei, Taiwan

[2]Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan

[3]Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan

Email: chchen@ntut.edu.tw, **cream08111230@gmail.com**, tphong@nuk.edu.tw

Yu-Qi Huang

- A graduate student in National University of Kaohsiung

Outline

01.

Fuzzy data mining

Fuzzy Data Mining

- Traditional data mining
  - binary data
  - consider whether to buy or not
- Real application
  - Should consider the purchased quantity
  - Fuzzy Data Mining
  - Which membership function is good?
- Membership function finding
  - Genetic Algorithm

02.

motivation

Motivation

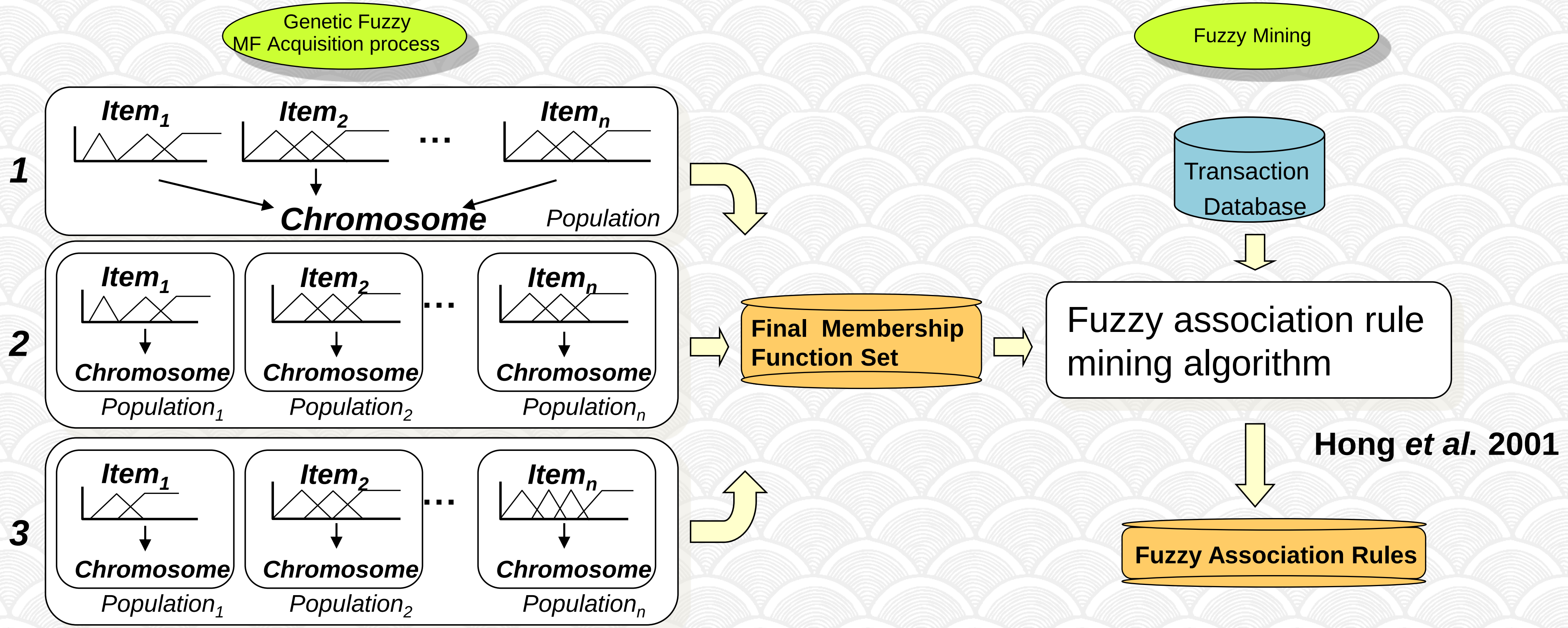
- The Crossover and Fitness Function has a hefty time cost of the previous method
- The time cost is so heavy that only the membership function of one candidate itemsets can be mined

03.

Proposed method

Proposed Method

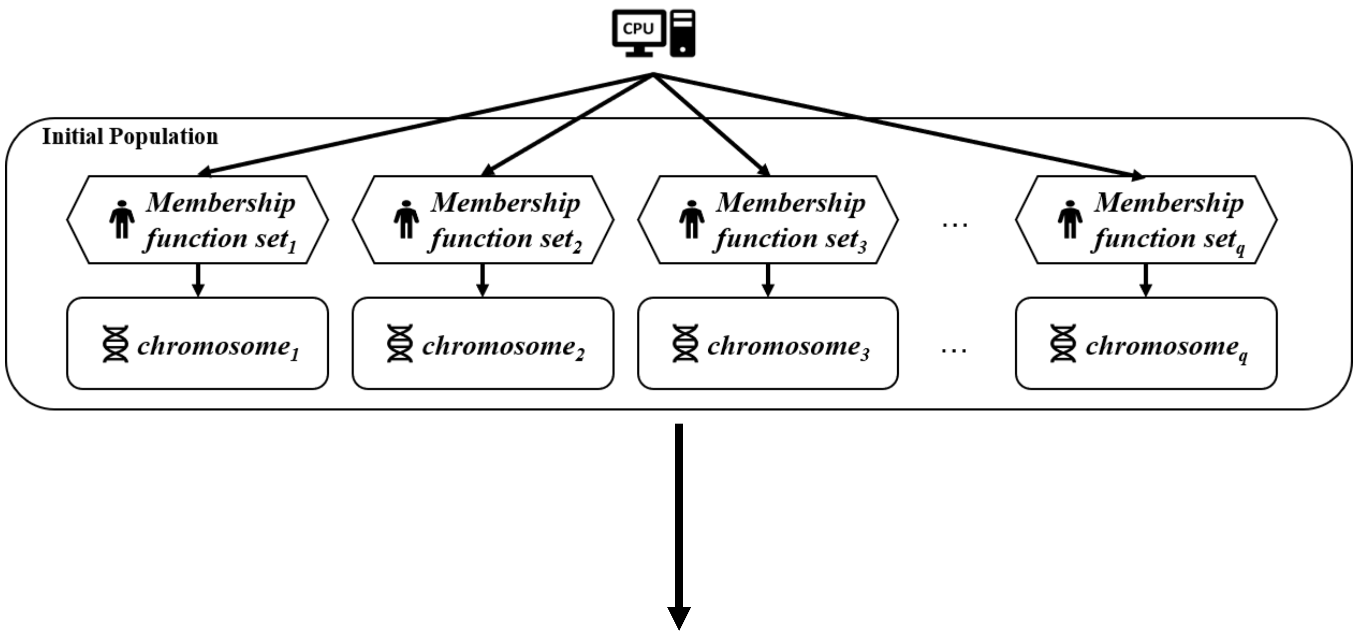
Phase1	Phase2
Mining Membership Functions	Mining Fuzzy Association Rules



Framework

1. Initial Population
2. MMA Crossover (GPU)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

Initial Population



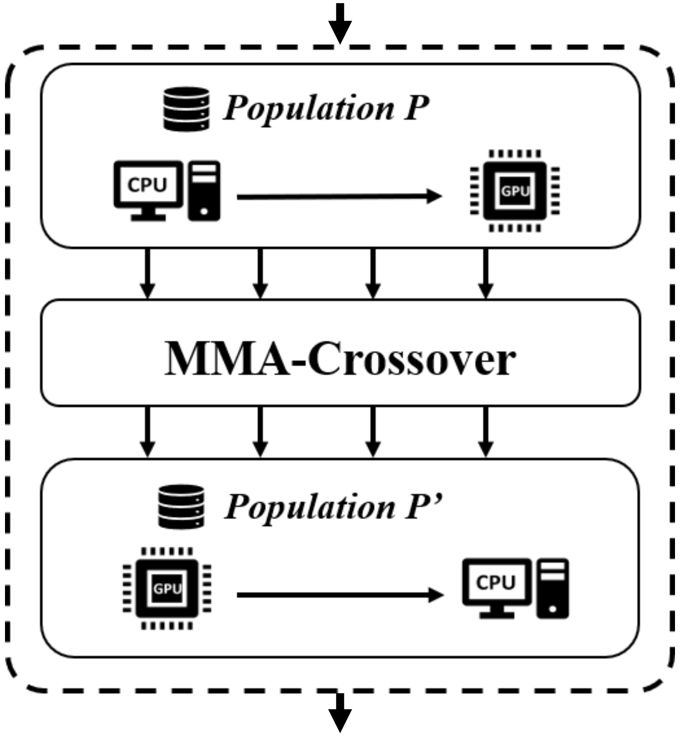
Output: chromosomes

Framework

1. Initial Population
2. MMA Crossover (GPU)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

MMA Crossover (GPU)

Input: initial chromosomes



Output: chromosomes after Crossover

Framework

1. Initial Population
2. MMA Crossover (GPU)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

Mutation

Input: initial chromosomes

Mutation

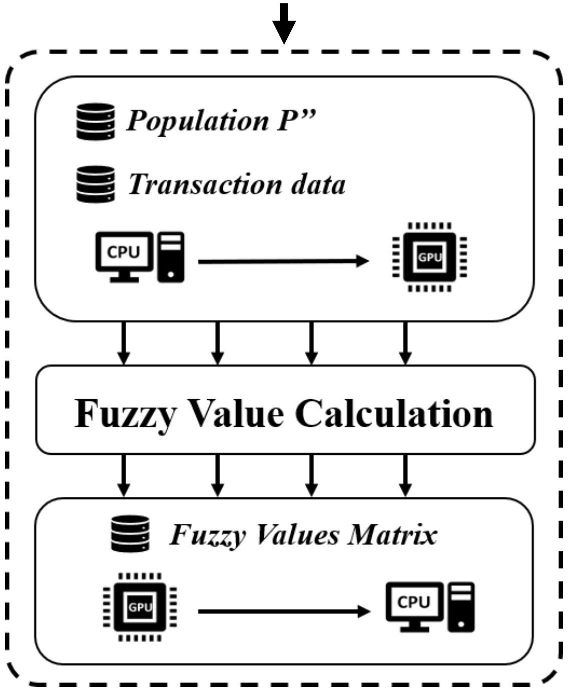
Output: chromosomes after Crossover

Framework

1. Initial Population
2. MMA Crossover (parallel)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

Fuzzy Value Calculation

Input: chromosomes, transactions



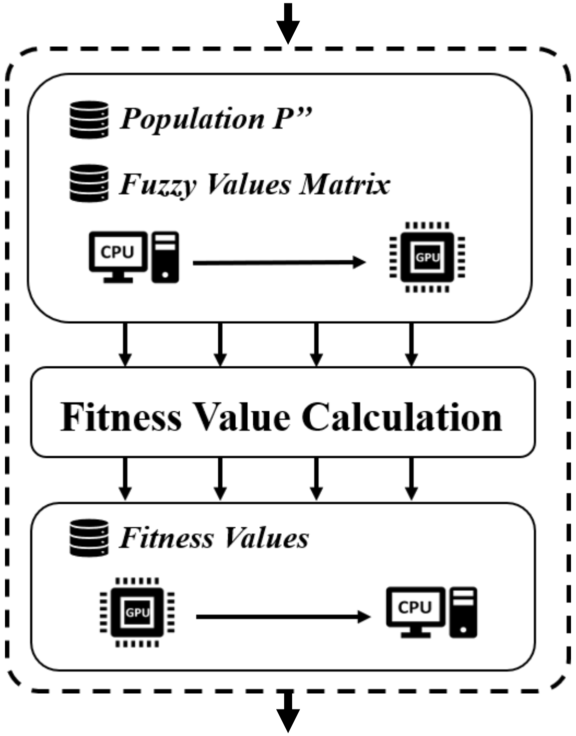
Output: Fuzzy Value Matrix

Framework

1. Initial Population
2. MMA Crossover (GPU)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

Fitness Value Calculation

Input: chromosomes, Fuzzy Value Matrix



Output: Fitness Values

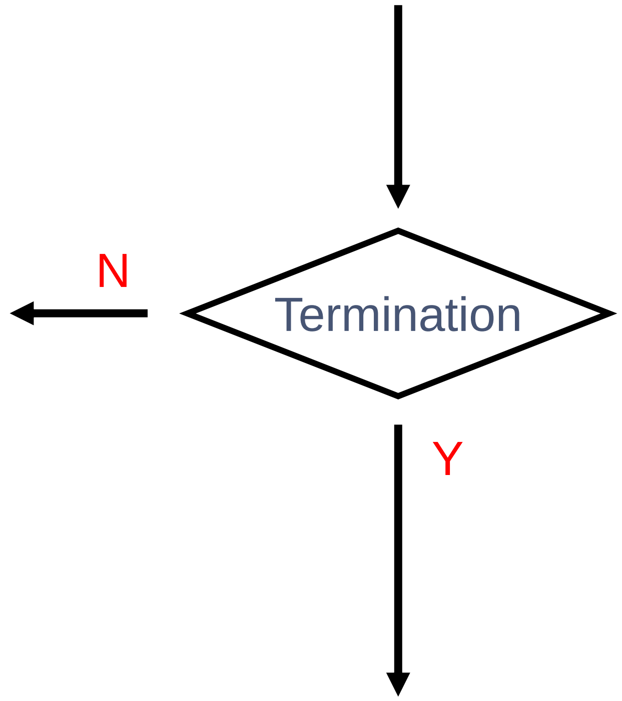
Framework

1. Initial Population
2. MMA Crossover (GPU)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

Termination

Input: Fitness Values

Repeat
- Step1
- Step2
- Step4
- Step4
- Step5



Output: Membership Function, Large One Itemset

Framework

1. Initial Population
2. MMA Crossover (GPU)
3. Mutation
4. Fuzzy Value Calculation (GPU)
5. Fitness Value Calculation (GPU)
6. Termination
7. Output

Output

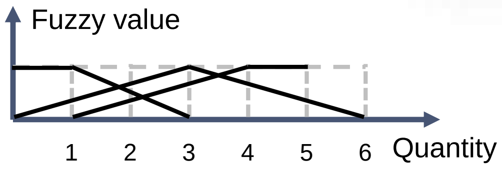
```
the best chromosome:
[[1. 2. 4. 5. 3. 3.]
 [1. 5. 5. 4. 3. 2.]
 [1. 5. 4. 4. 2. 5.]
 [1. 5. 4. 4. 3. 3.]]

frequent itemsets is:
◆ 1 itemsets:
000.low        000.mid        000.high       001.low        002.low
002.mid        002.high       003.low        003.mid        003.high
```

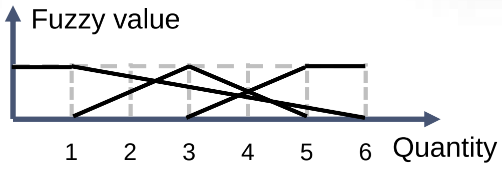
Proposed Method

TID	Items
T1	(bread, 1).
T2	(milk, 1); (bread, 5).
T3	(Milk, 3); (candy, 3).
T4	(milk, 1); (candy, 1); (bread, 3).
T5	(milk, 2); (jam, 2); (bread, 2)

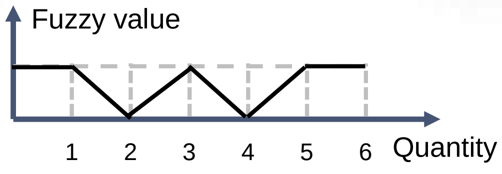
Milk



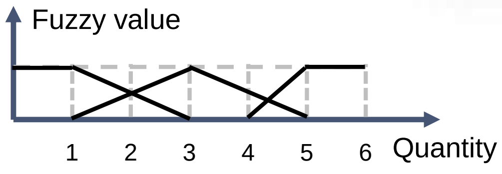
Jam



Candy

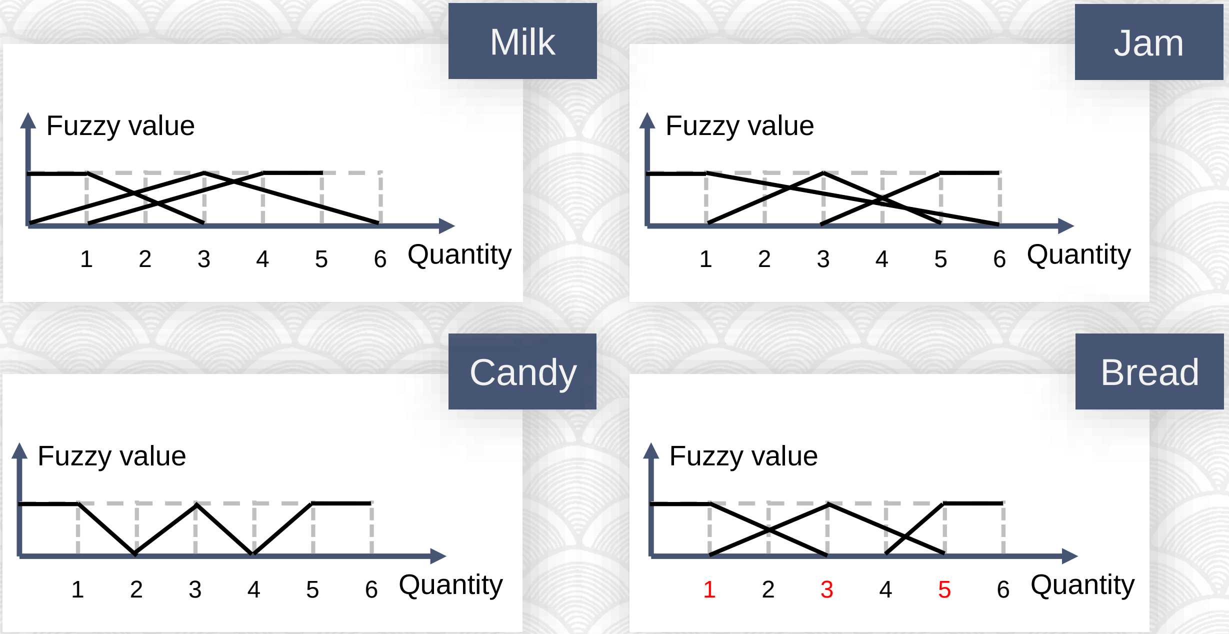


Bread



Chromosome

- **Example**



$$C1 = \overbrace{1, 2,\ 3, 3,\ 4, 3,}^{MF_1}\ \overbrace{1, 5,\ 3, 2,\ 5, 2,}^{MF_2}\ \overbrace{1, 1,\ 3, 1,\ 5, 1,}^{MF_3}\ \overbrace{1, 2,\ 3, 2,\ 5, 1}^{MF_4}$$

	MF_1			MF_2			MF_3			MF_4		
	Middle, range			Middle, range			Middle, range			Middle, range		
Fuzzy set	low	mid	high	low	mid	high	low	mid	high	low	mid	high

Chromosome (C_q) = a set of membership functions

MMA Crossover

arithmetical1	$C_1^{t+1} = \left(c_{11}^{t+1}, \dots, c_{1h}^{t+1}, \dots, c_{1z}^{t+1}\right)$ *where* $c_{1h}^{t+1} = dch + (1-d)c_h{}'$
arithmetical2	$C_2^{t+1} = \left(c_{21}^{t+1}, \dots, c_{2h}^{t+1}, \dots, c_{2z}^{t+1}\right)$ *where* $c_{1h}^{t+1} = dch' + (1-d)\mathrm{c_h}$
Maximum	$C_3^{t+1} = \left(c_{31}^{t+1}, \dots, c_{3h}^{t+1}, \dots, c_{3z}^{t+1}\right)$ *where* $min\ \{ch, ch'\}$
Minimum	$C_4^{t+1} = \left(c_{41}^{t+1}, \dots, c_{4h}^{t+1}, \dots, c_{4z}^{t+1}\right)$ *where* $min\ \{ch, ch'\}$

MMAC Example

- Each pair of genes calculate independent
- The number of genes is enormous
- We can do MMA crossover parallelly based on genes on the GPU

Membership Function $_1$						
C_1	1	2	2	4	4	1
C_2	2	1	3	3	5	2
Minimum	1	1	2	3	4	1
Maximum	2	2	3	4	5	2
Arithmetical1	1.65	1.35	2.65	3.35	4.65	1.65
Arithmetical2	1.35	1.65	2.35	3.65	4.35	1.35

…

Membership Function $_z$						
C_z	2	2	2	4	6	1
C_z	2	1	3	1	4	2
Minimum	2	1	2	1	4	1
Maximum	2	2	3	4	6	2
Arithmetical1	2	1.65	2.65	2.95	5.3	1.65
Arithmetical2	2	1.35	2.35	2.05	4.7	1.35

New chromosomes
generated by
MMA crossover

Mutation

- Single point mutation with random number

$$C1 = \overbrace{1, 2,\ 3, 3,\ 4, 3}^{MF_1},\ \overbrace{1, 5,\ 3, 2,\ 5, 2}^{MF_2},\ \overbrace{1, 1,\ 3, 1,\ 5, 1}^{MF_3},\ \overbrace{1, 2,\ 3, 2,\ 5, 1}^{MF_4}$$

$$C1 = \overbrace{1, 2,\ 3, 3,\ 4, 3}^{MF_1},\ \overbrace{1, 5,\ 3, 2,\ 5, ?}^{MF_2},\ \overbrace{1, 1,\ 3, 1,\ 5, 1}^{MF_3},\ \overbrace{1, 2,\ 3, 2,\ 5, 1}^{MF_4}$$

Fuzzy Value Calculation

TID	Items
T1	(bread, 1).
T2	(milk, 1); (bread, 5).
T3	(Milk, 3); (candy, 3).
T4	(milk, 1); (candy, 1); (bread, 3).
T5	(milk, 2); (jam, 2); (bread, 2)

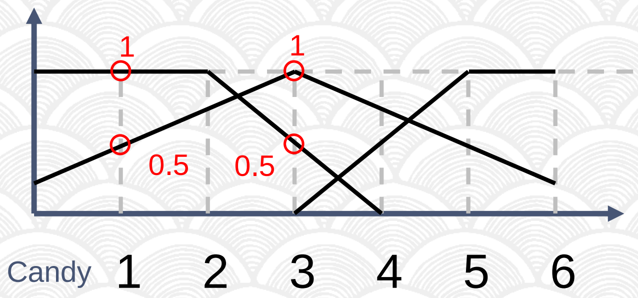
Input transactions

Item	Item index	The number of purchase in all transactions			
Milk	0	1	3	1	2
Jam	1	2	x	x	x
Candy	2	3	1	x	x
Bread	3	1	5	3	2

Input chromosome

index	Low middle	Low range	Mid middle	Mid range	High middle	High range
0	1	2	2	4	4	1
1	2	1	3	3	5	2
2	2	2	3	4	5	2
3	1.65	1.35	2.65	3.35	4.65	1.65

- Each item → three fuzzy value
- Sum all the fuzzy value as support
- Compare with minimum support
- With highly repetitive work



Fuzzy Value Calculation

Output			
Milk	Low	Mid	High
1	1	0.75	0
3	0	0.75	0
1	1	0.75	0
2	0.5	1	0
Support	2.5 / 5 = 0.5	3.25 / 5 = 0.65	0 / 5 = 0
Jam	Low	Mid	High
1	1	0.67	0
Support	1 / 5 = 0.2	0.67 / 5 = 0.134	0 / 5 = 0
Candy	Low	Mid	High
3	0.5	1	0
1	1	0.5	0
Support	1.5 / 5 = 0.3	1.5 / 5 = 0.3	0 / 5 = 0
Bread	Low	Mid	High
1	1	0.51	0
3	0	0.3	1
1	0	0.9	0
2	0.74	0.81	0
Support	1.75 / 5 = 0.348	2.51 / 5 = 0.501	1 / 5 = 0.2

- Milk, Jam, Candy, Bread are calculated in different threads
- Compare with minimum support
- Get the large one

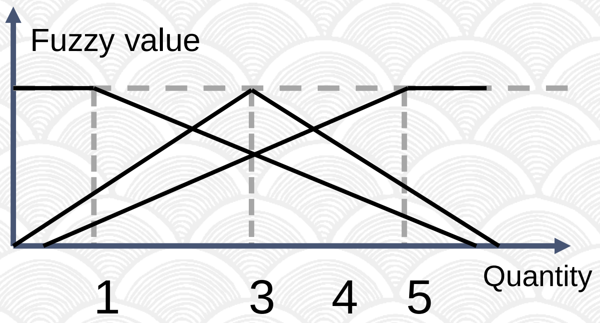
Fitness Function

- **Definition**
  - $f(C_q) = \frac{|L_1|}{suitability(Cq)}$, where $|L_1|$ is the number of large itemset ↑ better
    - $suitability(C_q) = overlap(C_q) + coverage(Cq)$

- **The two bad kinds of membership function**

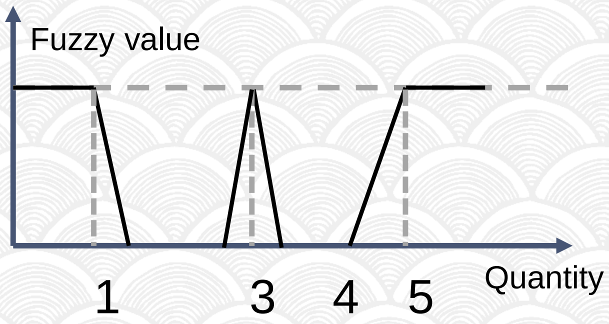
Overlap

(a) **Redundant** membership functions



Coverage

(b) **Separate** membership functions



Suitability

- **Definition**
  - $suitability(C_q)$

$$\sum_{j=1}^{m}[overlap(C_{qj}) + coverage(Cqj)]$$

  - **Overlap**

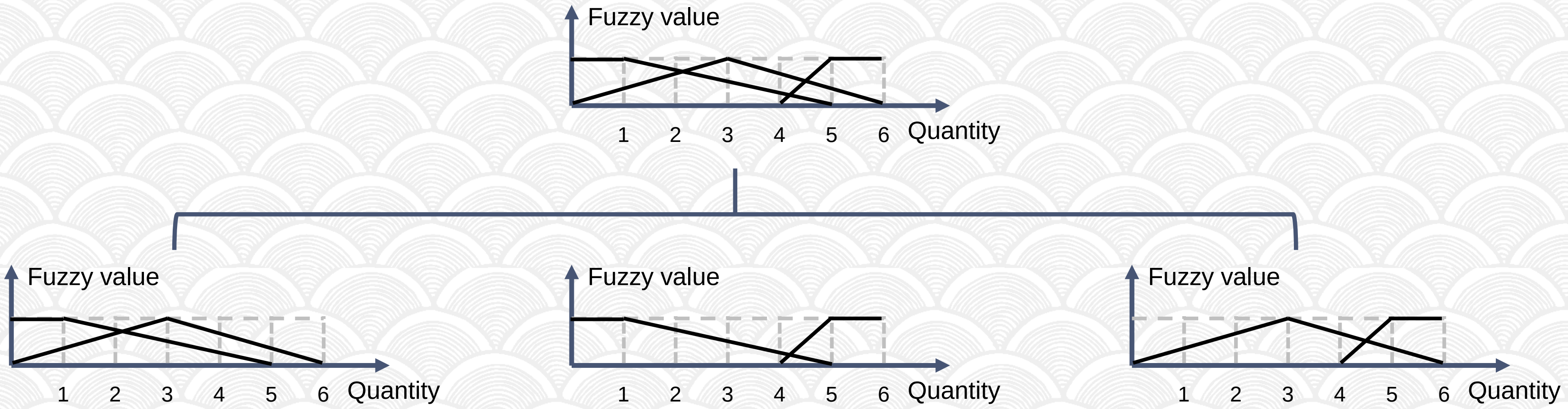
$$\sum_{j=1}^{m}[\max\left(\left(\frac{overlap(R_{jk,} R_{ji})}{\min(w_{jk}, wji)}\right), 1\right) - 1]$$

  - **Coverage**

$$\frac{1}{\frac{range(R_{j1,\ldots,} Rji)}{\max(I_j)}}$$

Overlap ↓ better

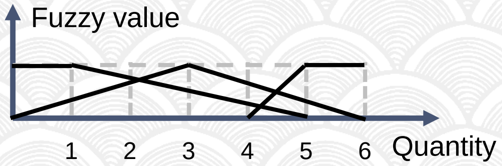
- $\sum_{j=1}^{m}[\max\left(\left(\frac{overlap(R_{jk},R_{ji})}{\min(w_{jk},wj_i)}\right),1\right)-1]$

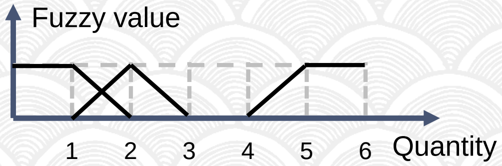


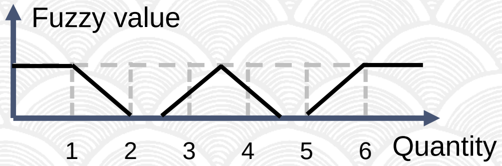
$\left(\frac{overlap(R_{jk},R_{ji})}{\min(w_{jk},wj_i)}\right)$	$\frac{5}{\min(5,3)}=\frac{5}{3}$	$\frac{1}{\min(4,1)}=1$	$\frac{2}{\min(3,1)}=2$
$\max\left(\left(\frac{overlap(R_{jk},R_{ji})}{\min(w_{jk},wj_i)}\right),1\right)$	$\max\left(\frac{5}{3},1\right)=\frac{5}{3}$	$\max(1,1)=1$	$\max(2,1)=2$
$\sum_{j=1}^{m}[\max\left(\left(\frac{overlap(R_{jk},R_{ji})}{\min(w_{jk},wji)}\right),1\right)-1]$	$\frac{5}{3}-1=\frac{2}{3}$	$1-1=0$	$2-1=1$

Coverage ↓ better

$$\frac{1}{\frac{range(R_{j1},\ldots,R_{ji})}{\max(I_j)}}$$







$$\frac{range(R_{j1},\ldots,Rji)}{\max(I_j)}$$

$$\frac{6}{6} = 1$$

$$\frac{3+1}{5} = \frac{4}{5}$$

$$\frac{2+2+1}{6} = \frac{5}{6}$$

$$\frac{1}{\frac{range(R_{j1},\ldots,R_{ji})}{\max(I_j)}}$$

$$1$$

$$\frac{5}{4}$$

$$\frac{5}{6}$$

Large One

Output			
Milk	Low	Mid	High
1	1	0.75	0
3	0	0.75	0
1	1	0.75	0
2	0.5	1	0
Support	2.5 / 5 = 0.5	3.25 / 5 = 0.65	0 / 5 = 0
Jam	Low	Mid	High
1	1	0.67	0
Support	1 / 5 = 0.2	0.67 / 5 = 0.134	0 / 5 = 0
Candy	Low	Mid	High
3	0.5	1	0
1	1	0.5	0
Support	1.5 / 5 = 0.3	1.5 / 5 = 0.3	0 / 5 = 0
Bread	Low	Mid	High
1	1	0.51	0
3	0	0.3	1
1	0	0.9	0
2	0.74	0.81	0
Support	1.75 / 5 = 0.348	2.51 / 5 = 0.501	1 / 5 = 0.2

- If minimum support is 0.3, Then |L1| = 6
- Large itemset:
  - ***milk.low, milk.mid, candy.low, candy.mid, bread.low, bread.mid***

Overlap

Input chromosome

index	Low middle	Low range	Mid middle	Mid range	High middle	High range
0	1	2	2	4	4	1
1	2	1	3	3	5	2
2	2	2	3	4	5	2
3	1.65	1.35	2.65	3.35	4.65	1.65

Example

Overlap

$$1.5 + 2.5 + 2 + 2.65 = 8.65$$

Index	Low, Mid	Low, High	Mid, High	
0	$\max\left(\frac{3}{\min(4,2)}, 1\right) - 1 = \frac{1}{2}$	$\max\left(\frac{3}{\min(2,4)}, 1\right) - 1 = \frac{1}{2}$	$\max\left(\frac{6}{\min(4,4)}, 1\right) - 1 = \frac{1}{2}$	$\frac{1}{2} + \frac{1}{2} + \frac{1}{2} = 1.5$
1	$\max\left(\frac{3}{\min(1,3)}, 1\right) - 1 = 2$	$\max\left(\frac{0}{\min(1,2)}, 1\right) - 1 = 0$	$\max\left(\frac{3}{\min(3,2)}, 1\right) - 1 = 1/2$	$2 + 0 + \frac{1}{2} = 2.5$
2	$\max\left(\frac{4}{\min(2,4)}, 1\right) - 1 = 1$	$\max\left(\frac{1}{\min(2,2)}, 1\right) - 1 = 0$	$\max\left(\frac{4}{\min(2,4)}, 1\right) - 1 = 1$	$1 + 0 + 1 = 2$
3	$\max\left(\frac{3.7}{\min(1.35,\ 3.35)}, 1\right) - 1 = 1.74$	$\max\left(\frac{0}{\min(1.35, 1.65)}, 1\right) - 1 = 0$	$\max\left(\frac{3}{\min(3.35, 1.65)}, 1\right) - 1 = 0.82$	$1.74 + 0 + 0.82 = 2.56$

Coverage

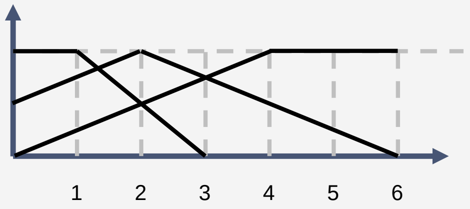
Input chromosome

index	Low middle	Low range	Mid middle	Mid range	High middle	High range
0	1	2	2	4	4	1
1	2	1	3	3	5	2
2	2	2	3	4	5	2
3	1.65	1.35	2.65	3.35	4.65	1.65

Example Coverage

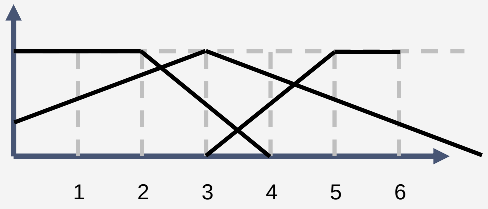
$$1 + 1 + 1 + 1 = 4$$

0



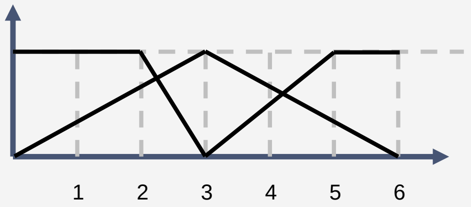
$$\frac{\frac{1}{6}}{6} = 1$$

2



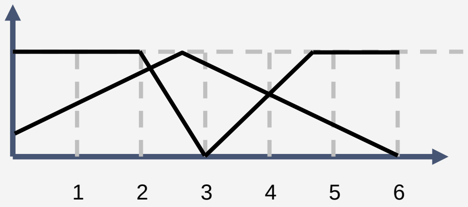
$$\frac{\frac{1}{7}}{7} = 1$$

1



$$\frac{\frac{1}{6}}{6} = 1$$

3



$$\frac{\frac{1}{6}}{6} = 1$$

Fitness Value

$$\frac{|L1|}{suitability(Cq)}$$

$$= \frac{|L1|}{\sum_{j=1}^{m} overlap(Cqj) + coverage(Cqj)}$$

$$= \frac{6}{8.65 + 4} = 0.4743$$

Output			
Milk	Low	Mid	High
1	1	0.75	0
3	0	0.75	0
1	1	0.75	0
2	0.5	1	0
Support	2.5 / 5 = 0.5	3.25 / 5 = 0.65	0 / 5 = 0
Jam	Low	Mid	High
1	1	0.67	0
Support	1 / 5 = 0.2	0.67 / 5 = 0.134	0 / 5 = 0
Candy	Low	Mid	High
3	0.5	1	0
1	1	0.5	0
Support	1.5 / 5 = 0.3	1.5 / 5 = 0.3	0 / 5 = 0
Bread	Low	Mid	High
1	1	0.51	0
3	0	0.3	1
1	0	0.9	0
2	0.74	0.81	0
Support	1.75 / 5 = 0.348	2.51 / 5 = 0.501	1 / 5 = 0.2

- If minimum support is 0.3, Then |L1| = 6
- Large itemset:
  - ***milk.low, milk.mid, candy.low, candy.mid, bread.low, bread.mid***

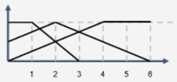
$1.5 + 2.5 + 2 + 2.65 = 8.65$

Index	Low, Mid	Low, High	Mid, High	Overlap
0	$\max\left(\frac{3}{\min(4,2)},1\right)-1=\frac{1}{2}$	$\max\left(\frac{3}{\min(2,4)},1\right)-1=\frac{1}{2}$	$\max\left(\frac{6}{\min(4,4)},1\right)-1=\frac{1}{2}$	$\frac{1}{2}+\frac{1}{2}+\frac{1}{2}=1.5$
1	$\max\left(\frac{3}{\min(1,3)},1\right)-1=2$	$\max\left(\frac{0}{\min(1,2)},1\right)-1=0$	$\max\left(\frac{3}{\min(3,2)},1\right)-1=1/2$	$2+0+\frac{1}{2}=2.5$
2	$\max\left(\frac{4}{\min(2,4)},1\right)-1=1$	$\max\left(\frac{1}{\min(2,2)},1\right)-1=0$	$\max\left(\frac{4}{\min(2,4)},1\right)-1=1$	$1+0+1=2$
3	$\max\left(\frac{3.7}{\min(1.35,\ 3.35)},1\right)-1=1.74$	$\max\left(\frac{0}{\min(1.35,1.65)},1\right)-1=0$	$\max\left(\frac{3}{\min(3.35,1.65)},1\right)-1=0.82$	$1.74+0+0.82=2.56$

Index	Coverage

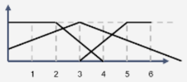
$1 + 1 + 1 + 1 = 4$

0



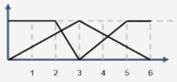
$\frac{\frac{1}{6}}{6} = 1$

2



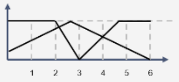
$\frac{\frac{1}{7}}{7} = 1$

1



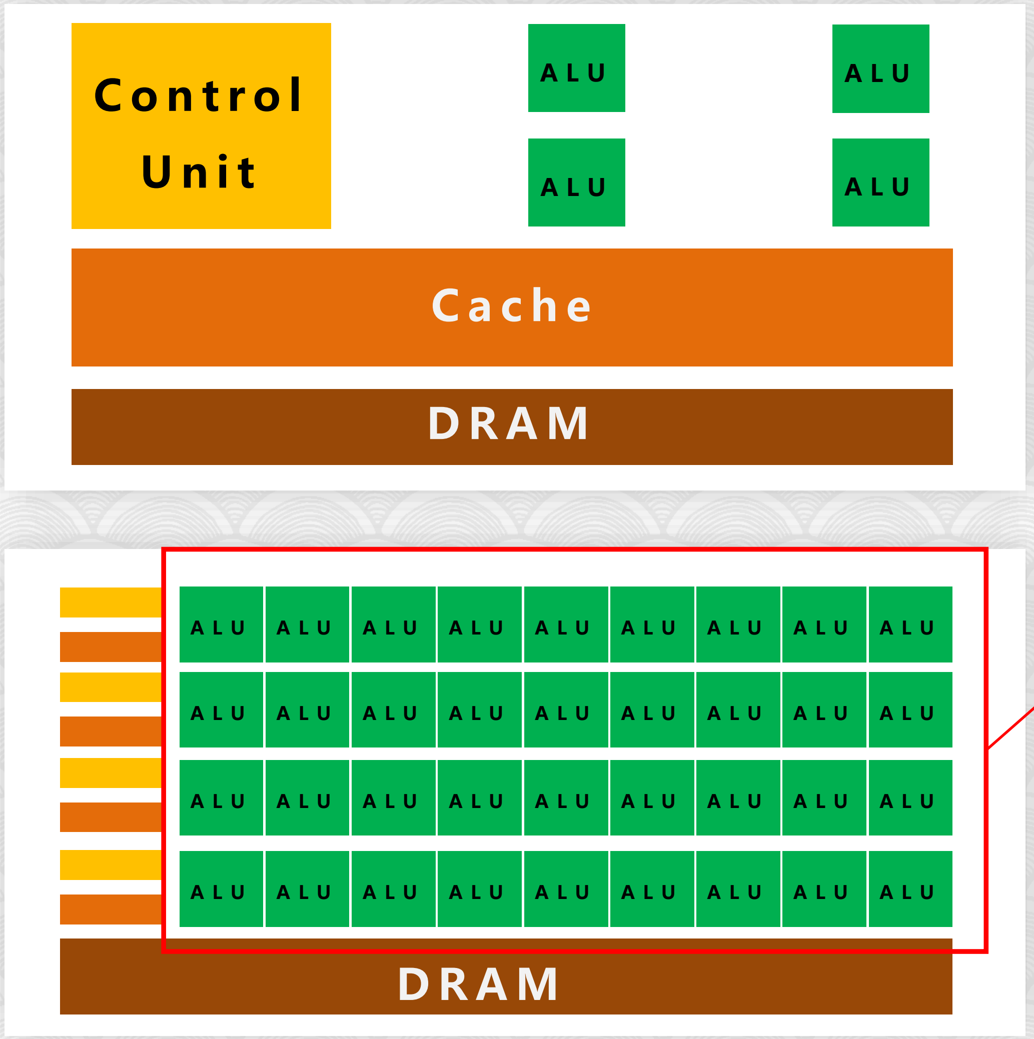
$\frac{\frac{1}{6}}{6} = 1$

3



$\frac{\frac{1}{6}}{6} = 1$

Fitness on GPU

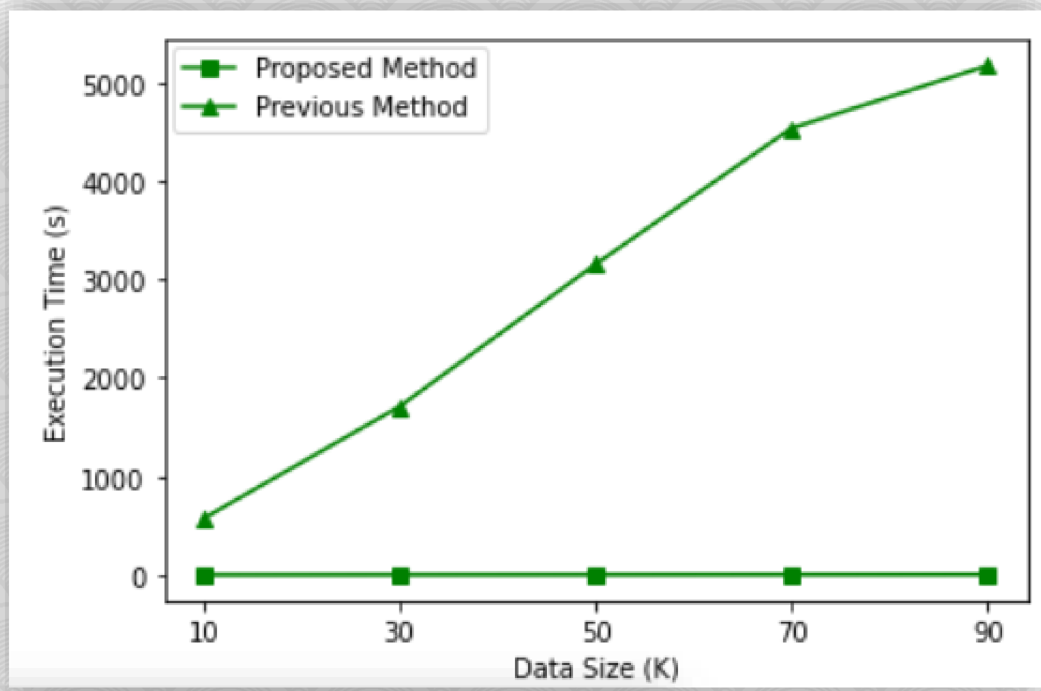


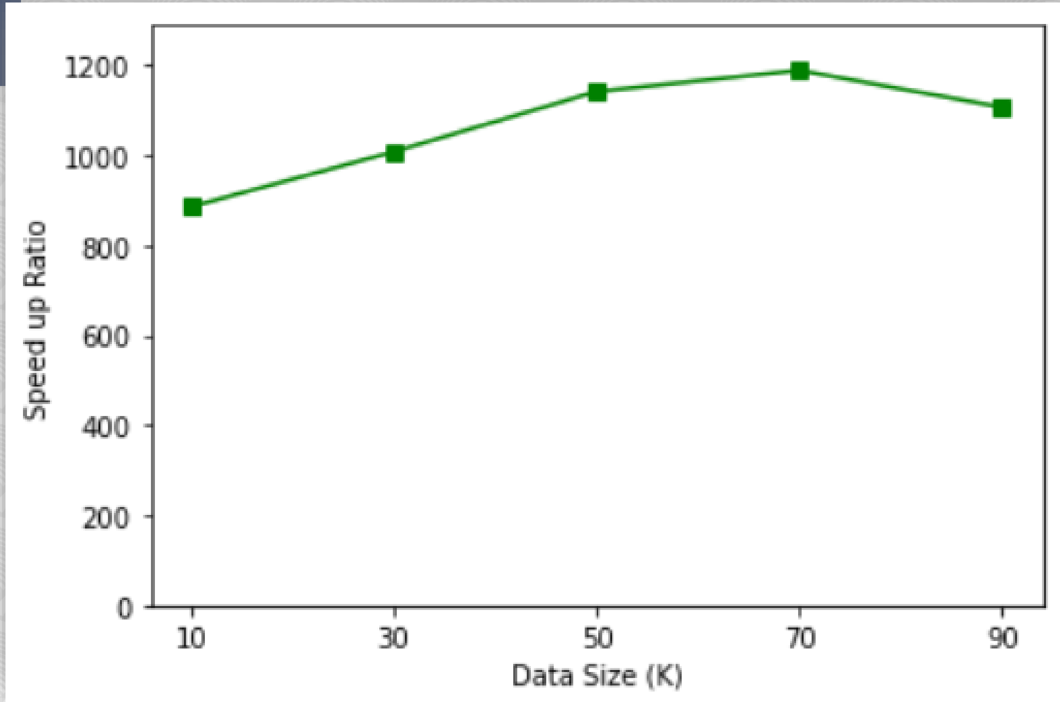
CONCEPT

- Each chromosome has to calculate its fitness value
- Each of the calculation, which includes overlap, coverage and fitness function, is independent
- With the massive parallelism of the GPU, we can execute the same function with different parameters, which are called by the pointer
- The fitness value calculation in our method is parallel executed based on chromosome

Experiment

- Time cost comparison between our method and the previous method
- Our method is almost 900~1200 times faster
- With the growth of the data, our time cost does not change much





Conclusion

- Significantly peed up
- Time has not grown exponentially with the amount of data
- There are some limitation in our method because of the GPU architecture

An Effective Approach for Genetic-Fuzzy Mining Using the Graphics Processing Unit

Thanks you!

Questions?

Chun-Hao Chen[1], **Yu-Qi Huang**[2] and Tzung-Pei Hong[2, 3]
Email: chchen@ntut.edu.tw, **cream08111230@gmail.com**, tphong@nuk.edu.tw