School of Computer Science and Engineering

Optimized node clustering by using genetic algorithm

Haoran Mei

Dec, 2021

iCAN Lab

Overview

- Introduction
- System model
- Simulation result
- Conclusion

Overview

- Introduction
- System model
- Simulation result
- Conclusion

Introduction

❖ Trajectory planning for UAV assisted data collection

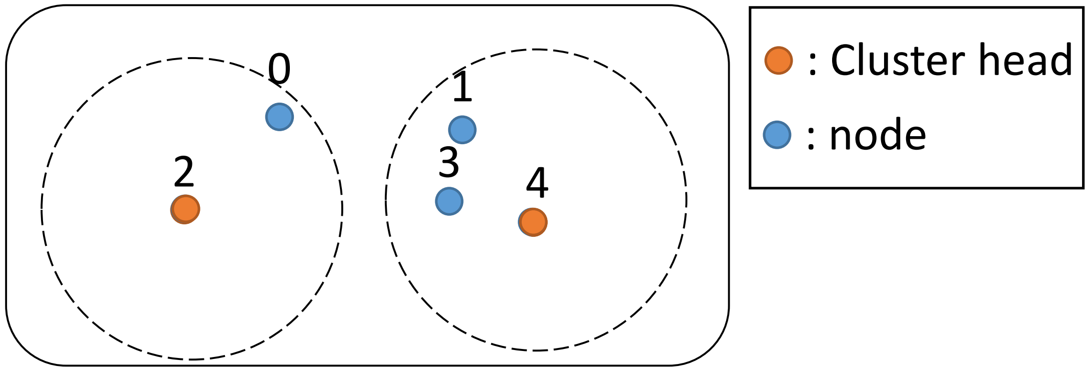
❑ Two general steps

➢ Cluster head election based node clustering

- Cluster head (CH) identification ➔ Group the nodes within the region of a CH as a cluster
- Cluster heads should collect data from nodes with their own region and deliver them to the UAV.

➢ Path planning

- Shortest tour that passes through all cluster heads. (TSP)



Introduction

❖ Trajectory planning for UAV assisted data collection

❑ Two general steps

➢ Cluster head election based node clustering

- Cluster head (CH) identification ➔ Group the nodes within the region of a CH as a cluster
- Cluster heads should collect data from nodes with their own region and deliver them to the UAV.

➢ Path planning

- Shortest tour that passes through all cluster heads. (TSP)

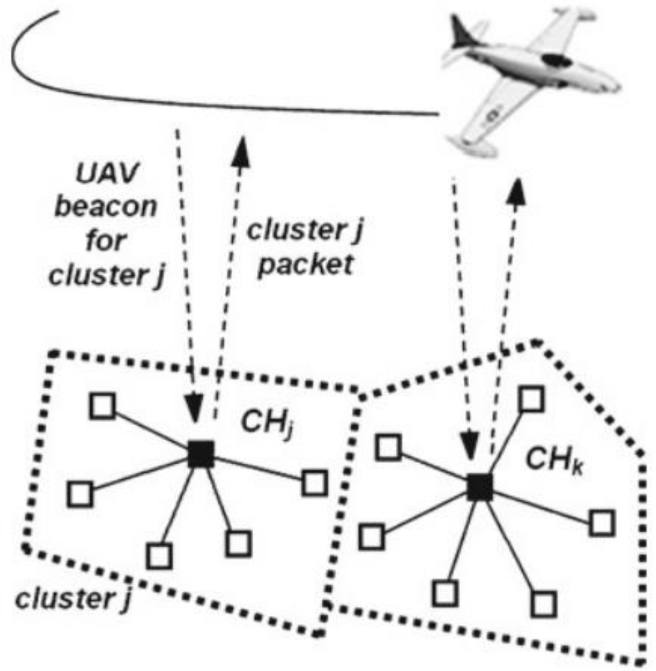
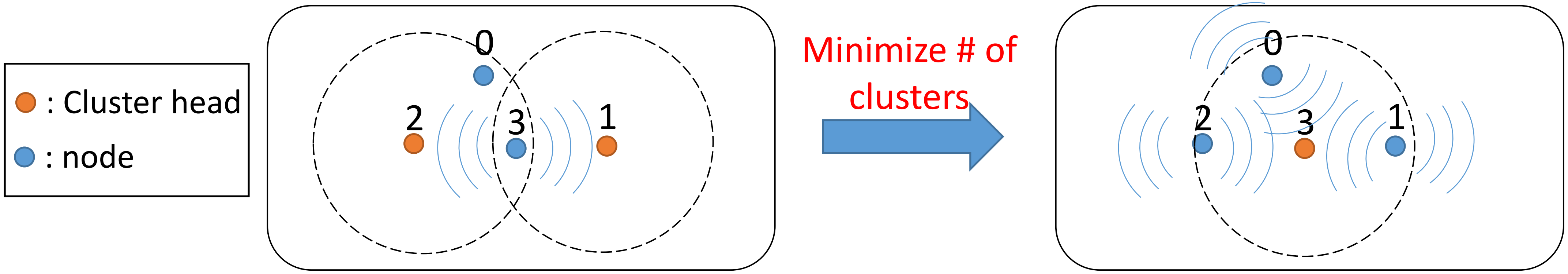


Fig. 2 Basic scheme for cluster-based UAS data collection

Introduction

❖Problem definition

- Given a set $\boldsymbol{D} = \{d_1, \dots, d_K\}$ of K nodes
  - Group them into N clusters with $N < M$, where M is a given integer.
- Basic assumption
  - It is allowed that a node is covered by multiple cluster heads.
    - The UAV would receive redundant data, since the node transmit data to adjacent CHs by broadcasting.
    - The region defined by the CH = The communication range.
  - It is desirable to have as few clusters as possible.
    - This could alleviate the degree of overlap

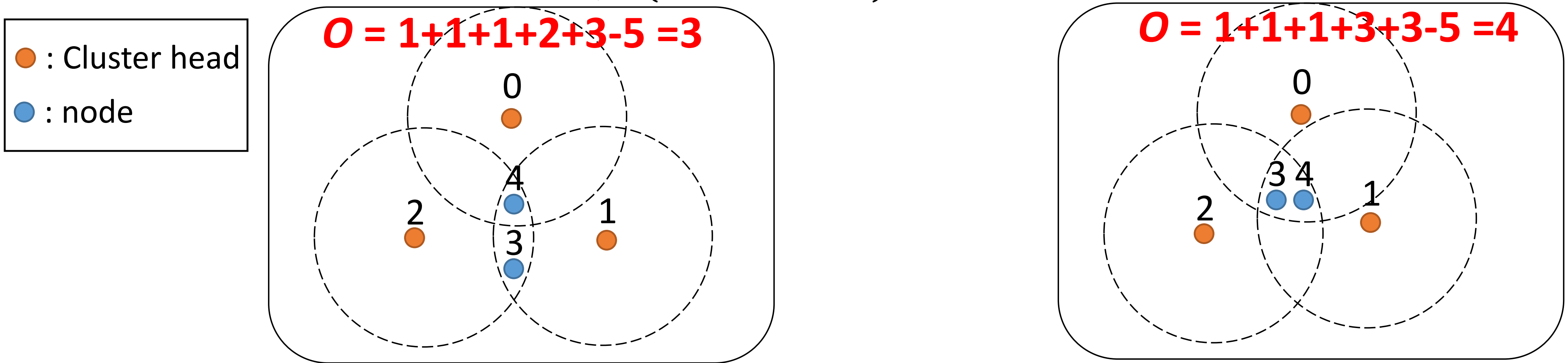


Introduction

❖ Node clustering

❑ Set cover problem with additional goal

- ➢ Given a set ***I*** = $\{1, \ldots, K\}$ of integers and a collection ***S*** = $\{\{1\}, \ldots, \{1,3, \ldots, K\}\}$ of sets whose union is ***I***
  - ***I* = *D*, *S* = potential cluster configurations.**
- ➢ Goal: identify a smallest sub-collection of ***S*** whose union is ***I***
  - Minimizing the # of clusters which can cover all nodes.
- ➢ Extra goal: if multiple feasible solutions ==> chose the one with minimal overlap degree.
  - Overlap degree, $O = \sum_{i=1}^{K}\left(\sum_{s\in S} 1\{i \in s\}\right) - K$: The amount of redundant data.



Overview

- Introduction
- System model
- Simulation result
- Conclusion

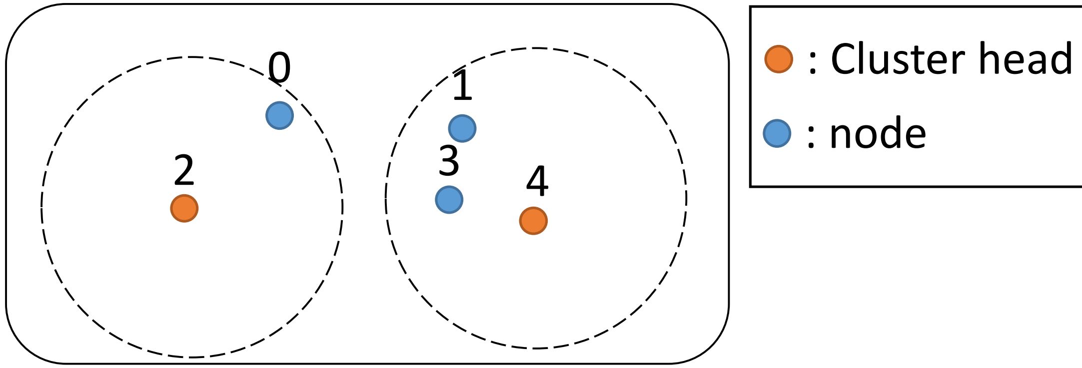
System model

❖Model description

❑Representation (*K* nodes)

- *K* bit binary digit.
  - ✓ *i-th* bit: the binary value implies whether the corresponding node is the cluster header

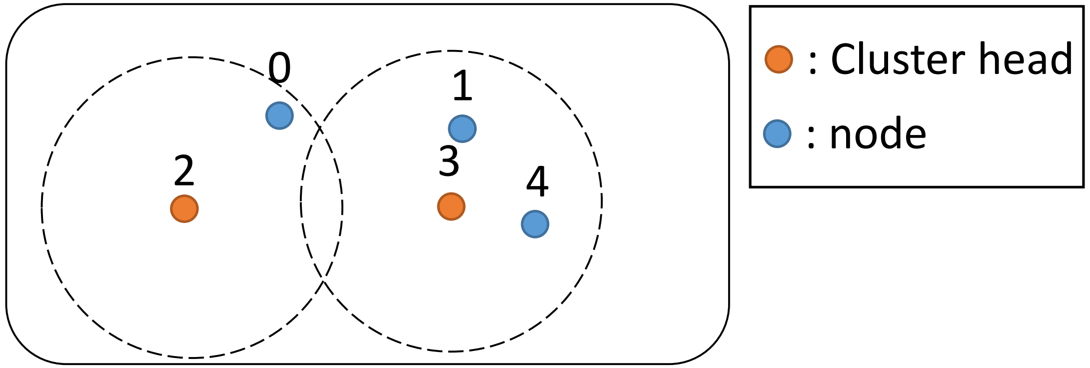
Nodes	0	1	2	3	4
Binary digit	0	0	1	0	1



❑Operators

- Change the binary values in the digit.
- Group the nodes according to the result.

Nodes	0	1	2	3	4
Binary digit	0	0	1	1	0



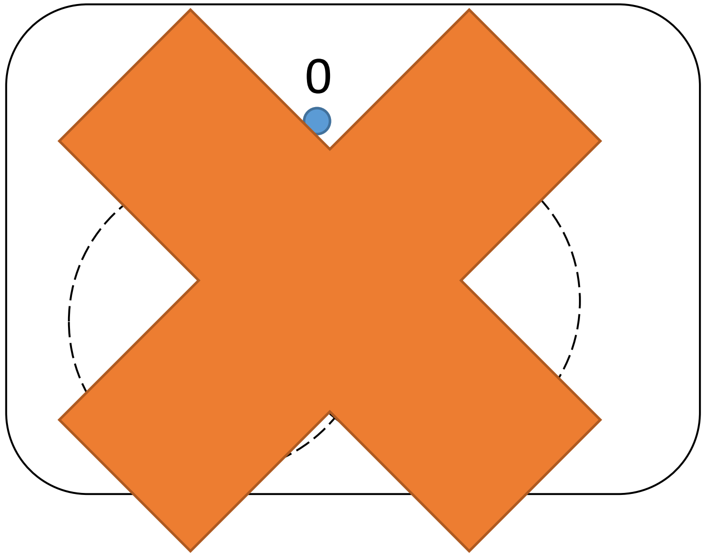
System model

❖Model description

❑Fitness function

➢The priority for this problem.

- P1: Ensure that all *K* nodes are grouped



System model

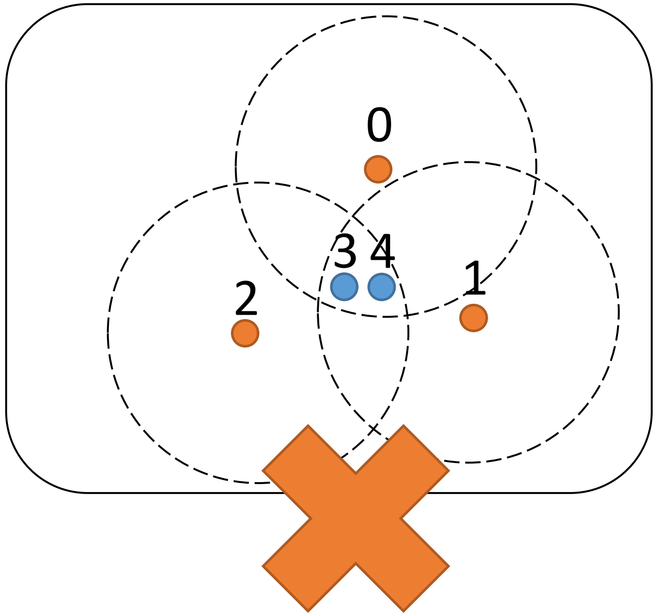
❖Model description

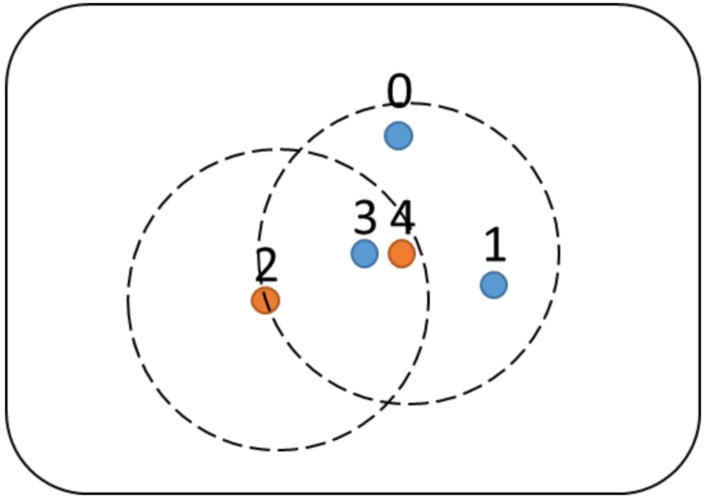
❑Fitness function

➢The priority for this problem.

- P1: Ensure that all *K* nodes are grouped
- P2: Ensure that number of the generated clusters, *N*, is less than or equals to *M*

When M = 2





Okay !

System model

❖Model description

❑Fitness function

➢The priority for this problem.

- P1: Ensure that all *K* nodes are grouped
- P2: Ensure that number of the generated clusters, *N*, is less than *M*
- P3: Minimize the number of clusters
- P4: Minimize the degree of overlap

Constraints

# of ungrouped nodes

The set of clusters

$$fitness = \begin{cases} -\text{ error } * K^3, & if \ \text{ error} > 0 \\ -(|C| - M) * K^2, & if \text{ error} == 0, |C| \geq M \\ -(|C| * K + O), & if \text{ error} == 0, |C| < M \end{cases}$$

P1

P2

P3 & P4

The degree of overlap

```
# calculates the fitness
def calculateFitness(self, M):
    if self.error > 0:
        self.fitness = 0 - self.error * (K ** 3)
    elif self.num_cluster > M:
        self.fitness = (M - self.num_cluster) * (K ** 2)
    else:
        cluster_fit = self.num_cluster * K
        self.fitness = 0 - (cluster_fit + self.overlap)
```

System model

❖ Problem formulation

$$\max_{C^*} \ fitness:$$

$$fitness = \begin{cases} -\text{error} * K^3, & if \ \text{error} > 0 \\ -(|C| - M) * K^2, & if \ \text{error} == 0, |C| \geq M \\ -(|C| * K + O), & if \ \text{error} == 0, |C| < M \end{cases}$$

P1

P2

P3 & P4

$$O = \sum_{n_j \in D} \sum_{c_i \in C} 1\{\|n_j - c_i\| \leq r\} - |\mathbf{D}|$$

The degree of overlaps

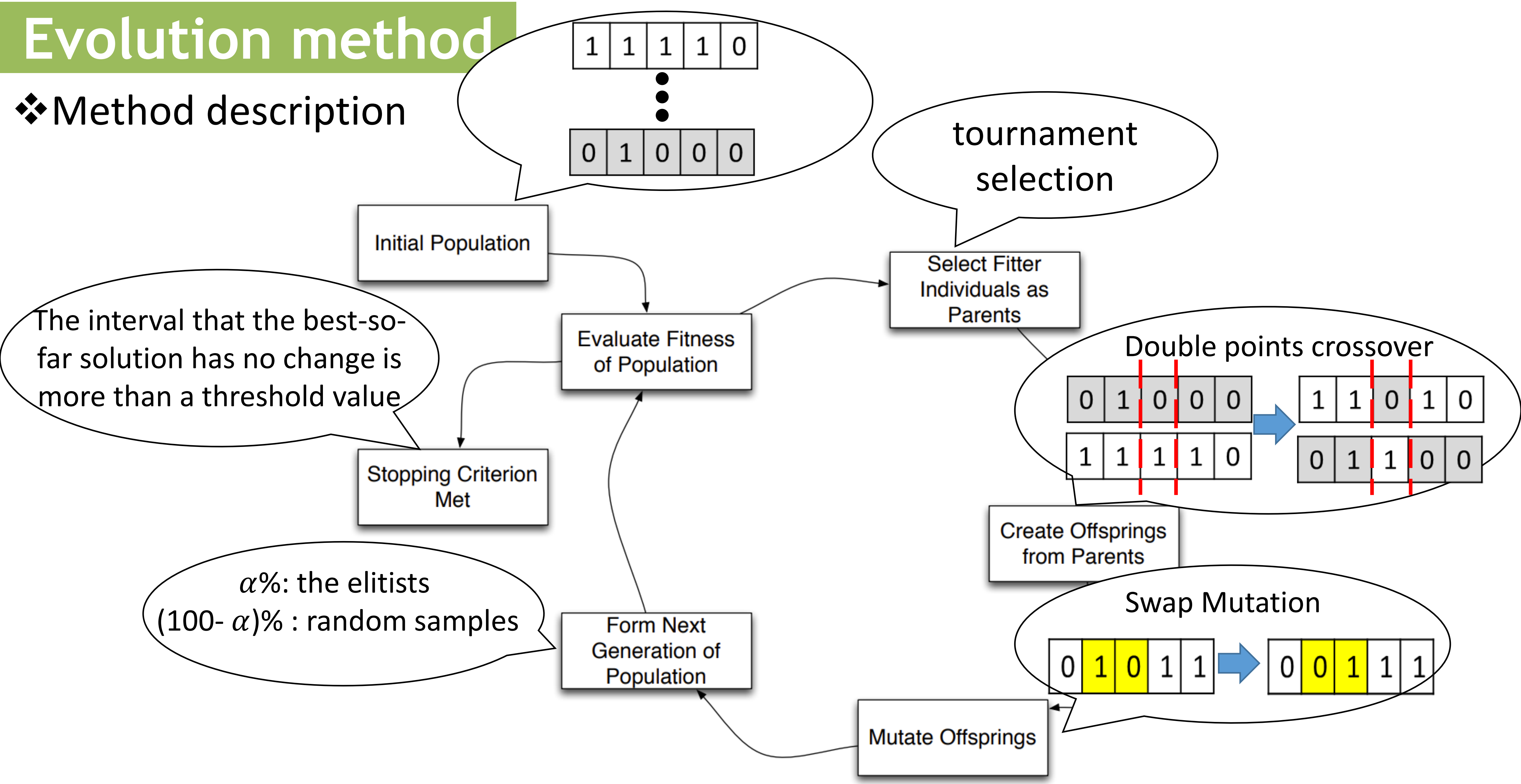
$$\text{error} = \sum_{n_j \in Node} 1\left\{\sum_{c_i \in C} 1\{\|n_j - c_i\| \leq r\} == 0\right\}$$

# of ungrouped nodes

The radius of the region defined by CH

Evolution method

❖Method description



Overview

- Introduction
- System model
- Simulation result
- Conclusion

Simulation result

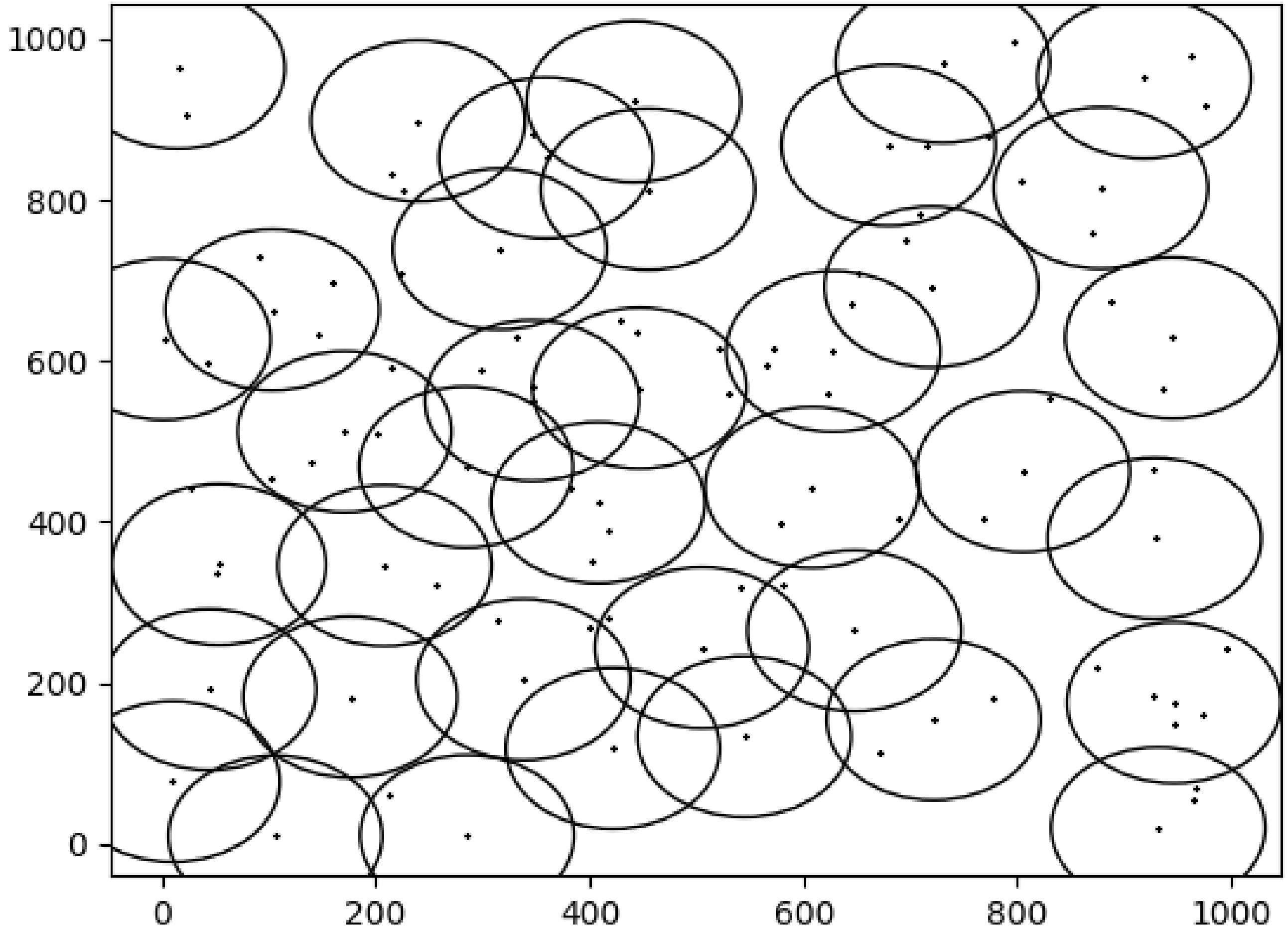
❖Parameter configuration

```
K = 100 # The number of nodes, cannot be change
M = 50 # The required number of cluster
a = 0.5 # alpha: the ratio of the elitist to the next generat
pro_mutation = 0.2 # The probability of executing mutation
size_pop = 1000 # The size of population for each generation
round_crossover = int(size_pop/4) # The rounds of crossover
r = 100 # The radius of cluster region
max_interval = 500 # The maximum interval between old and new
```

Parameters	Values	Notice or Range
The number of nodes	100	Should be fixed
The required number of clusters	50	1 ~ 100
-alpha	0.5	0 ~ 1
The probability of mutation	0.2	0 ~ 0.5
The size of population	1000	1000 ~
The rounds of crossover	250	250 ~ 1000
The radius of the region defined by each CH	100	Fixed
The threshold on the interval that best-so-far solution has no change	500	**200** ~ 1000 (empirical test)

Simulation result

❖ Example of clustering result



Simulation result

- To see the impact of probability of mutation (max_interval = 200)
  - The mutation could more or less speed up the convergence process
  - No significant change in the fitness.

Probability of mutation	Result (running 20 times for the same initial pop)		
	average # of generation	average opt fitness	average time cost (second)
0 (Only crossover)	521.7	-3779.05	174.925
0.1	531.1	**-3776.6**	174.29375
0.2	512.1	-3794.35	157.1421875
0.3	509	-3794.05	152.21953125
0.4	548.7	-3783.8	162.134375
0.5	**503.75**	-3779.15	**134.6125**

Simulation result

- To see the impact of alpha (max_interval = 200)
  - Elitists selection has a positive effect on finding a better fitness.
  - Speed up the convergence process

alpha (The ratio of the elitists to the next generation of population)	Result (running 20 times for the same initial pop)		
	average # of generation	average opt fitness	average time cost (second)
0 (random selection)	779.25	-4197.65	421.65
0.2	579.05	-3794.75	197.26
0.4	554.4	-3789.2	168.8671875
0.6	**534.75**	-3794.15	155.59
0.8	537.35	-3798.8	**152.44**
1 (elitists selection)	540.35	**-3783.95**	152.58

Overview

- ❖Introduction
- ❖System model
- ❖Simulation result
- ❖Conclusion

Conclusion

- Combinatorial problem?
  - To check it
    - Definition: Search an optimal solution from a finite solution space.
    - Characteristic:
      - In most cases, exhaustive search is not tractable.
        - ✓ The size of solution space: 2^K (The # of possible *K*-bit binary digits)
      - The domain of the problems is the discrete set of feasible solutions or can be reduced to discrete.
        - ✓ The size of feasible solution is countable: $\leq 2^K$
      - The goal is to find the best solution
        - ✓ To minimize the number of clusters and the degree of overlap

Conclusion

- Limitations of the design on the genetic algorithm
  - The search space of cluster centers is based on cluster head election
    - Not continuous.
    - There is little possibility that the result is the global optima.
- Future work
  - Develop the grid-based clustering method using genetic algorithm.
    - The search space could be larger.


Thank you

- "E:\2020\IoT Course\NodeClustering\venv\python\Scripts\python.exe" "E:/2020/IoT Course/Advanced-Software-Analysis-202102/SourceCode/mutation_check.py"
- ==mutation_prob=0.0==
- average # of generation: 521.7
- average opt fitness: -3764.05
- average time cost: 174.925
- ==mutation_prob=0.1==
- average # of generation: 531.1
- average opt fitness: -3769.6
- average time cost: 174.29375
- ==mutation_prob=0.2==
- average # of generation: 512.1
- average opt fitness: -3794.35
- average time cost: 157.1421875
- ==mutation_prob=0.30000000000000004==
- average # of generation: 509
- average opt fitness: -3794.05
- average time cost: 152.21953125
- ==mutation_prob=0.4==
- average # of generation: 548.7
- average opt fitness: -3783.8
- average time cost: 162.134375
- ==mutation_prob=0.5==
- average # of generation: 523.75
- average opt fitness: -3779.15
- average time cost: 134.6125

- ==alpha=0.0==
- average # of generation: 779.25
- average opt fitness: -4197.65
- average time cost: 421.65
- ==alpha=0.2==
- average # of generation: 579.05
- average opt fitness: -3794.75
- average time cost: 197.25859375
- ==alpha=0.4==
- average # of generation: 554.4
- average opt fitness: -3789.2
- average time cost: 168.8671875
- ==alpha=0.6000000000000001==
- average # of generation: 534.75
- average opt fitness: -3794.15
- average time cost: 155.5921875
- ==alpha=0.8==
- average # of generation: 537.35
- average opt fitness: -3798.8
- average time cost: 152.4421875
- ==alpha=1.0==
- average # of generation: 540.35
- average opt fitness: -3783.95
- average time cost: 152.58046875

Simulation result

- To see the impact of parent selection operators (max_interval = 500)
  - No significant difference

Operators	Result (running 20 times for the same initial pop)		
	average # of generation	average opt fitness	average time cost (second)
Tournament selection	952.15	-3793.3	421.55
exponential ranking & roulette wheel sampling	947.85	-3808.75	393.19