GTA Project

Lokesh G

Ranjith Kumar

Problem Statement

- Aimed at a multiple traveling salesman problem (MTSP) with multiple depots and closed paths, this paper proposes a k-means clustering donkey and a smuggler algorithm (KDSA). The algorithm first uses the k-means clustering method to divide all cities into several categories based on the center of various samples; the large-scale MTSP is divided into multiple separate traveling salesman problems (TSPs), and the TSP is solved through the DSA. The proposed algorithm adopts a solution strategy of clustering first and then carrying out, which can not only greatly reduce the search space of the algorithm but also make the search space more fully explored so that the optimal solution of the problem can be more quickly obtained.

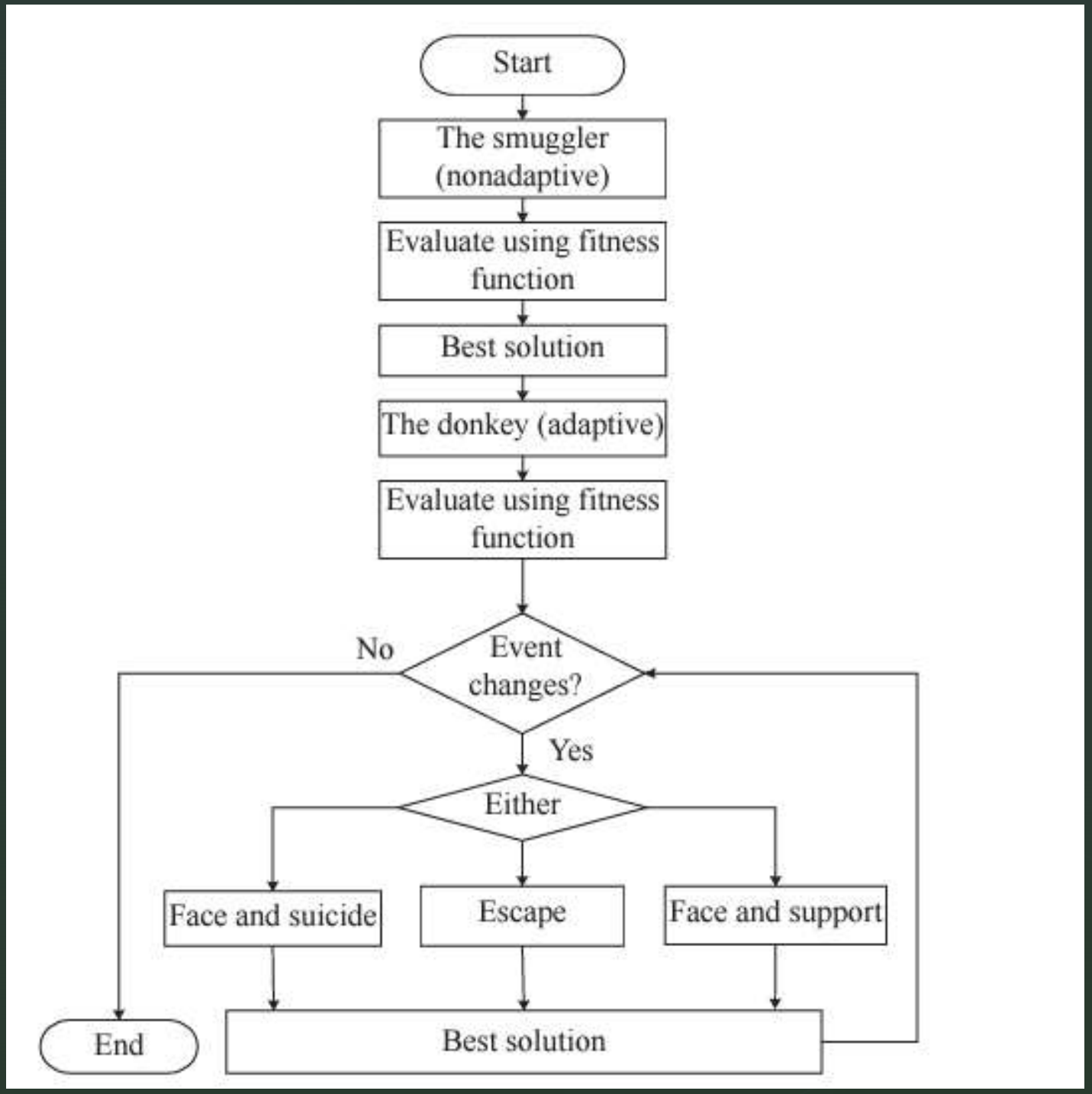
The Proposed Algorithm

- The first part of the algorithm is to divide cities into clusters with the help of K-means clustering. This part of the algorithm is very simple and elementary
- The second part of the algorithm is to evaluate the DSA algorithm on each clustered set. This enables us to find the optimal solution to the DSA algorithm
- We have thus implemented the second part of the algorithm which involves a non adaptive and an adaptive part.

Key Points of the Algorithm

- The algorithm is written and implemented in python. It takes the graph as an input and gives the optimal distance and the path as an output.
- The first step is the non-adaptive part which calculates the optimal path from each node in a greedy way.
- The adaptive part or the donkey part depends on the behaviour of the donkey.
- The next slide shows the algorithm in detail

- The steps to solve the DSA are as follows: Step 1
- Nonadaptive (smugglers)
- (i)Determine the parameters of each solution.
- (ii) Re-evaluate to calculate the fitness value of each possible solution.
- (iii) Pick out the optimal solution then transmit it to the donkey.
- Step 2 Adaptive (donkey)
- (i) Use the best solution selected in Step 1.
- (ii) Re-evaluate the fitness value of the current solution. If its fitness value changes, then continue to use the fitness function to find the optimal solution.
- (iii) Determine whether the current optimal solution is congested. If so, then perform one of the following steps:
- i) Escape: Re-evaluate the fitness value of the possible solutions, then pick out the one with the lowest fitness value as the optimal result.
- ii) Confront and suicide: The second best solution can be selected as the optimal one.
- iii) Confront and support: when achieved the overloaded optimal solution, use (5) to select the second-best possible solution to support the best. Then use (6) to combine the optimal and the suboptimal solution to create the optimal support solution.


Start
The smuggler (nonadaptive)
Evaluate using fitness function
Best solution
The donkey (adaptive)
Evaluate using fitness function
Event changes?
No
Yes
Either
Face and suicide
Escape
Face and support
Best solution
End

Results

```
Enter the distance from the city 2 to 0: 13
Enter the distance from the city 2 to 1: 14
Enter the distance from the city 2 to 3: 10
Enter the distance from the city 2 to 4: 15
Enter the distance from the city 3 to 0: 12
Enter the distance from the city 3 to 1: 16
Enter the distance from the city 3 to 2: 10
Enter the distance from the city 3 to 4: 15
Enter the distance from the city 4 to 0: 16
Enter the distance from the city 4 to 1: 9
Enter the distance from the city 4 to 2: 15
Enter the distance from the city 4 to 3: 17
Here are your choices
1: Update the Matrix
2: Choose the RUN behaviour of the Donkey
3: Choose the SUICIDE behaviour of the Donkey
4: Choose the SUPPORT behaviour of the Donkey
Enter the choice you want to make: 3
New Path Length: 57
Updated path: 142301
The duration for computing the SUICIDE behaviour is 0:00:00.000090
Here are your choices
1: Update the Matrix
2: Choose the RUN behaviour of the Donkey
3: Choose the SUICIDE behaviour of the Donkey
4: Choose the SUPPORT behaviour of the Donkey
Enter the choice you want to make: 2
68
([61, 68, 60, 63, 57], {0: '032140', 1: '142301', 2: '230412', 3: '320413', 4: '410324'})
The duration for computing the RUN behaviour is 0:00:00.000259
Here are your choices
1: Update the Matrix
2: Choose the RUN behaviour of the Donkey
3: Choose the SUICIDE behaviour of the Donkey
4: Choose the SUPPORT behaviour of the Donkey
Enter the choice you want to make: 3
New Path Length: 57
Updated path: 142301
The duration for computing the SUICIDE behaviour is 0:00:00.000092
Here are your choices
1: Update the Matrix
2: Choose the RUN behaviour of the Donkey
3: Choose the SUICIDE behaviour of the Donkey
4: Choose the SUPPORT behaviour of the Donkey
Enter the choice you want to make: 4
The Original Path is: 0,1,4,2,3,0,
The support path is : 0,2,3,4,1,0,
The duration for computing the SUPPORT behaviour is 0:00:00.000085
Here are your choices
1: Update the Matrix
2: Choose the RUN behaviour of the Donkey
3: Choose the SUICIDE behaviour of the Donkey
4: Choose the SUPPORT behaviour of the Donkey
Enter the choice you want to make:
```