TSP Art

Course Instructor: Prof. *Dr. Gabriel Robins*

Author(s): *Tamal Saha, Hamid Bagheri*

(ts4rq, hb2j)@virginia.edu

12/1/2009

Abstract

The traveling salesman problem (TSP) is a well-known NP-complete problem. It has appeared in diverse field of human knowledge. Different heuristics have been developed to generate approximate solutions for TSP. In this project, we are using 3 different heuristics for TSP to implement a half toning algorithm.

Approach

In this project, we have taken the following steps to implement a half toning algorithm.

Given any input image,

- If the image is a color image, convert it into a black & white image.
- Now we generate the "cities"/vertices of a complete graph on which TSP algorithm will be applied. We distribute cities with a density that locally approximates the darkness of a source image.
- Now we run nearest neighbor, greedy and 2-opt heuristics for TSP on the "cities". This produces a single closed path that resembles the original image.

The following diagram presents the steps takes.

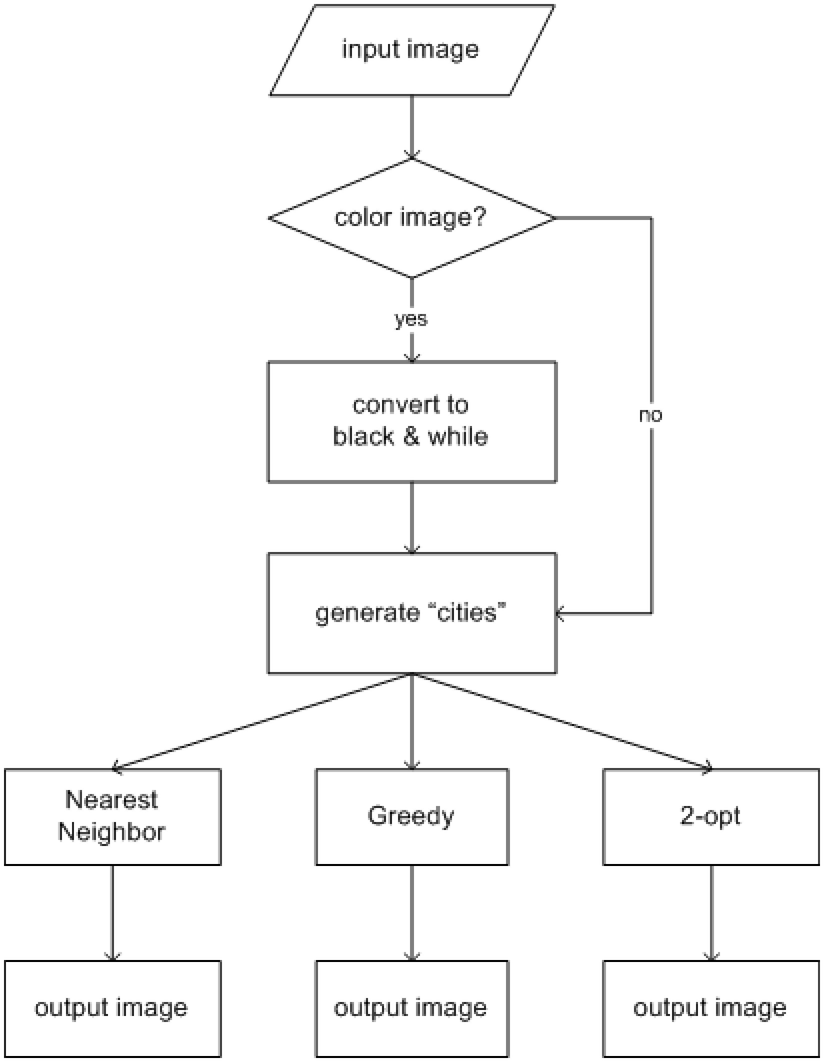


Fig 1: TSP Art steps

Nearest Neighbor algorithm

The nearest neighbor (NN) algorithm chooses the nearest unvisited city as the next destination.

Pseudo code

```
1. Select an arbitrary vertex as current vertex.
2. Find out the shortest edge connecting current vertex and an unvisited vertex V.
3. Set current vertex to V.
4. Mark V as visited.
5. If all the vertices in domain are visited, then terminate; otherwise, go to step 2.
```

The sequence of the visited vertices is the output of the algorithm.

Time complexity:

O(n^2), where n = number of cities

Greedy Algorithm

The Greedy heuristic gradually constructs a tour by repeatedly selecting a new shortest edge and adding it to the tour as long as it doesn't create a cycle with less than N edges.

Pseudo code:

1. Sort all edges.
2. Select the shortest edge and add it to our tour if it doesn't violate any of the above constraints.
3. Do we have N edges in our tour? If no, repeat step 2.

Time complexity:

O($n^2\log_2(n)$), where n = number of cities

2-opt algorithm

The 2-Opt algorithm takes the output tour from either NN or Greedy algorithms as an input and convert it to a shorter one. Each 2-Opt step consists of eliminating two edges and reconnecting the two resulting paths in a different way in order to obtain a new shorter tour. It is important to note that there is just one way to reconnect the two resulting path from eliminating two edges. Checking whether an improving 2-Opt step exists takes *$O(n^2)$* time since we need to consider all pairs of tour edges.

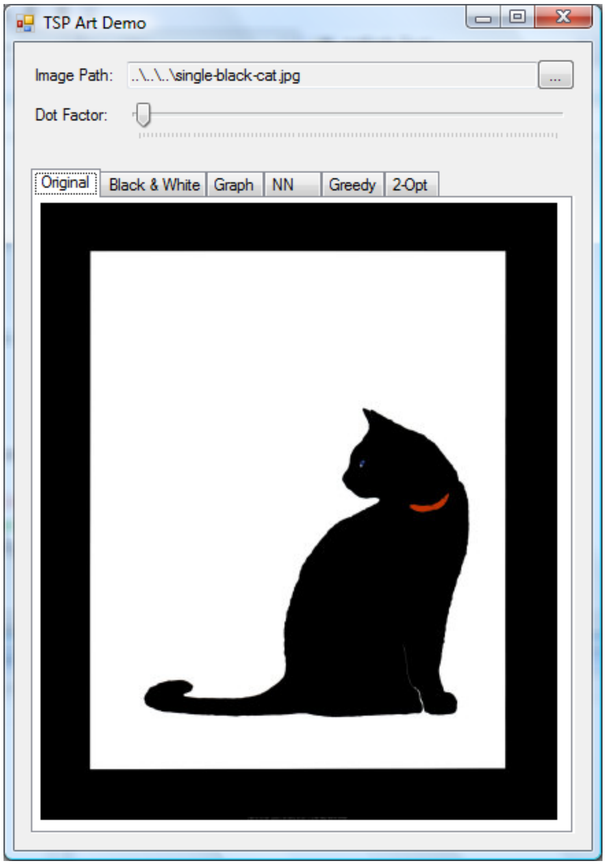


Image Path: ..\..\..\single-black-cat.jpg

Dot Factor:

Fig 2: input color image

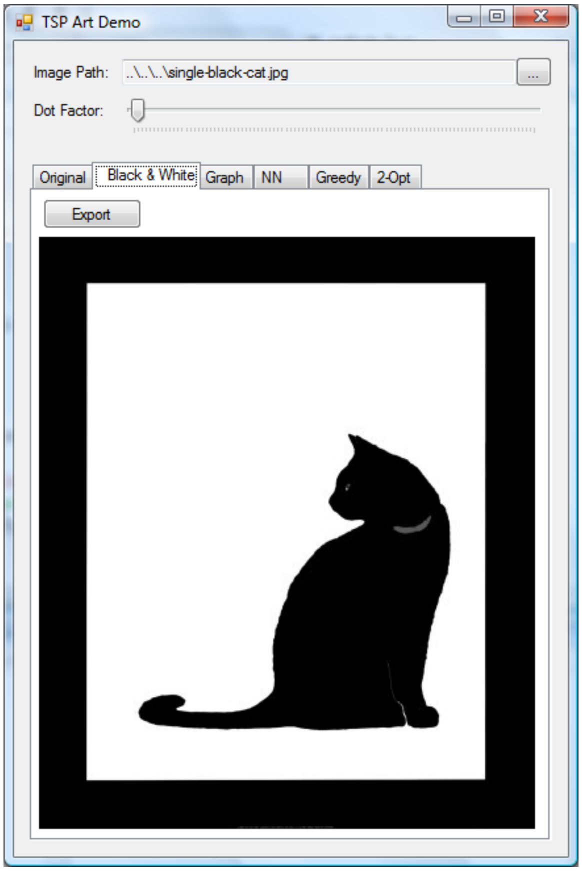
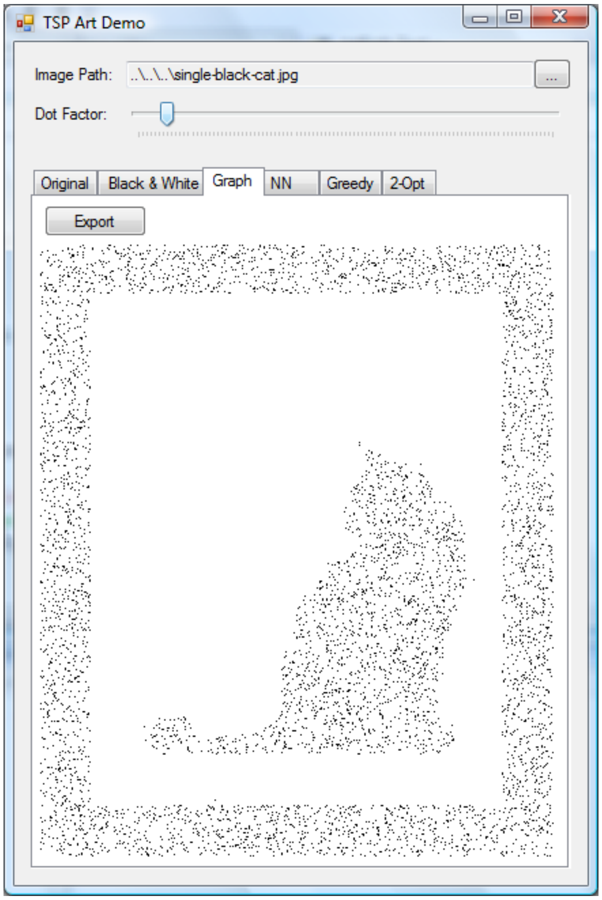


Image Path: ..\..\..\single-black-cat.jpg

Dot Factor:

Fig 3: converted black & white image

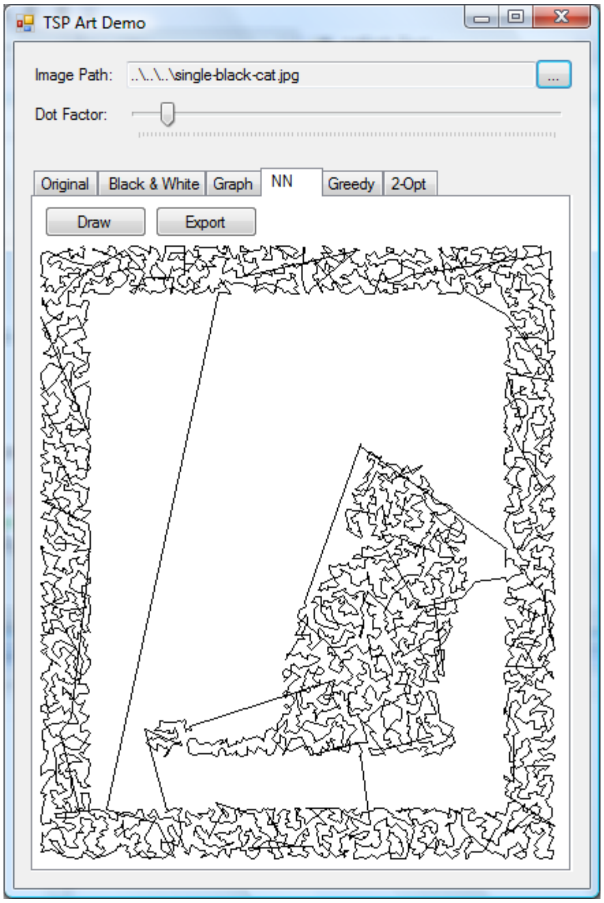


TSP Art Demo

Image Path: ..\..\..\single-black-cat.jpg

Dot Factor:

Fig 4: cities for NN alg



TSP Art Demo

Image Path: ..\..\..\single-black-cat.jpg

Dot Factor:

Fig 5: output of NN alg

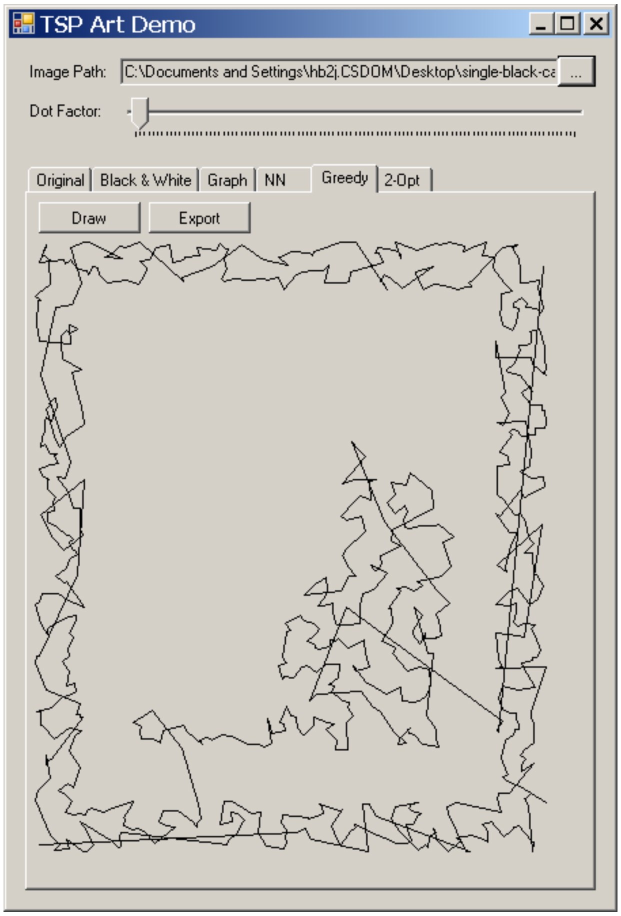


Image Path: C:\Documents and Settings\hb2j.CSDOM\Desktop\single-black-ca

Fig6: output for Greedy alg

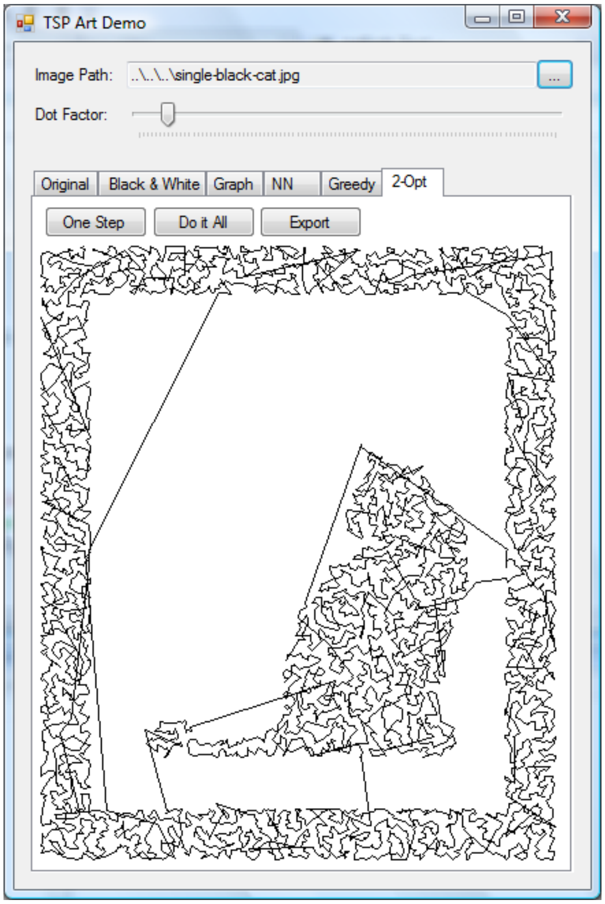


Image Path: ..\..\..\single-black-cat.jpg

Fig 7: output for 2-opt alg

References:

1. http://en.wikipedia.org/wiki/Travelling_salesman_problem
2. http://en.wikipedia.org/wiki/Nearest_neighbour_algorithm
3. "Heuristics for the traveling salesman problem," Nilsson, C. in the journal of Theoretical Computer Science Reports, Linkoping University
4. "Improving Solutions," The Traveling Salesman, Springer Berlin / Heidelberg, vol. Volume 840/1994, 1995, pp. 413-470.