Finding Near-Optimal Routes for Multiple Security Patrols in a University Campus Road Network

James Carlo Plaras and Jaime Samaniego
Institute of Computer Science
University of the Philippines Los Baños
College, Laguna, Philippines, 4031
jceplaras@uplb.edu.ph, jmsamaniego@uplb.edu.ph

ABSTRACT

Security patrol is an important preventive measure against threat and danger. Since this involves routing, valid and near-optimal routes can be determined using techniques in Graph Theory. This study finds routes for a given number of security patrols in a certain university campus using the Min-Max k-Chinese Postman Problem (MM k-CPP) as the problem model. This problem was solved using simulated annealing (SA) and the results were competitive with other heuristics. The method was implemented with a user friendly interface that would accept Open Street Maps (OSM) data in XML format.

Keywords

Open Street Maps, Routing, Security Patrols, Simulated Annealing

1. INTRODUCTION

1.1 Significance and Objectives of the study

Route planning is an important area because of the need to solve well-designed and optimal routes. As Ahr [1] mentioned, a lot of organizations perform services that are usually related to routing. Examples of these are picking up and distributing materials and waste collection and management for which these organizations spend a lot of time and money for repeated transport activities.

Security patrols also need to plan their patrol routes in order to maximize police visibility and minimize operational cost.

This study finds near-optimal solution routes for a given number of security patrols in a university. Specifically, it intends to plan a tour for each patrol where all streets in the university campus are covered by at least one patrol, and the longest tour is minimized. This effectively balances the load among the different patrols. The program implemented accepts an input map from Open Street Map (OSM) and outputs the map together with the solution routes. The problem was modelled as a Min-Max k-Chinese Postman problem.

1.2 Review of Related Literature

In the dissertation by Ahr [1], he suggested that when studying routing problems in road networks, it is advantageous to represent this network as a graph, a mathematical concept which consists of nodes and arcs. Nodes may represent cities, towns, intersection or any point that may be needed to service. Arcs represent connections between these nodes which could be roads.

Routing problems could either be a node routing problem, where services need to be performed on the nodes, or be an arc routing problem, where services are need to be performed on arcs. Arc routing problems are also known as postmen problems. An example of an arc routing problem is the Chinese postman problem.

The Chinese postman problem (CPP) was formulated by Mei-ko Kwan. The objective of the CPP is to find a route for a single postman where he passes and delivers mail on each street at least once, and gets back to the starting point [14].

The k-chinese postman problem (k-CPP) is a generalization of the Chinese postman problem according to Guan, as cited by Osterhues and Mariak [10]. It allows $k > 1$ postmen to visit the roads with the same objective to be fulfilled.

Other arc routing problems include the following. The Rural Postman Problem (RPP), which is a generalization of the CPP in which a subset of the edges in the graph, referred as required edges, are only needed to be traversed [6]. The Capacitated Arc Routing Problem (CARP) is defined by Golden and Wong as finding a number of tours such that each arc with positive demand is serviced by exactly one vehicle, the sum of demand of those arcs serviced by each vehicle does not exceed the capacity, and the total cost of the tours is minimized [15].

New problems can be derived from these problems by adding more constraints. For example, by adding the constraint that an arc could be passed in only one direction, it will produce the directed variations of CPP, RPP, and CARP. More constraints can be added like time dependencies, priority arcs, etc.

Usually, real world arc routing problems are related to Min-

Max k-CPP (MM k-CPP) because these problems are often divided into sub-problems which are handled by multiple *postmen* which need to provide services given a shortest amount of time.

Let $G = (V, E)$ be a connected graph where $V = \{v_0, ..., v_m\}$ is the set of nodes in the graph and $E = \{e_0, e_1, ..., e_n\}$ is the set of edges required to be passed. The structure of each edge is $e_n = (v^s_{e_n} - v^d_{e_n}, w_{e_n})$ composed of a source node, v^s and destination node, v^d and a corresponding weight w. The objective of MM k-CPP is to find a k-postmen tour from the set of all tours $T = (T_0, T_1, ..., T_k)$ that minimizes the maximum cost tour in T [1].

$$min\{max^k_{i=0}(cost(T_i)) \mid T_i \in T\}$$

MM k-CPP is NP-hard but approximable within $2 - 1/k$ of the optimum according to Frederickson et. al. [5]. Since this problem is NP-hard, finding exact solutions in polynomial time is unlikely.

In a study by Reis et. al. [12], they proposed a tool called GAPatrol, a tool used for assisting police managers to find the routes. It uses a genetic algorithm to find routes for police patrol units.

A similar study in routing using a genetic algorithm was done by Huang et al. [7]. Route planning for hazardous materials (HAZMAT) was considered in the study. Compared to the first study which also uses a genetic algorithm, a Geographic Information System (GIS) was used in the study for providing data to be input to the modified genetic algorithm used by the authors in the study.

Using GIS for security planning is not new. As stated by Kuo et al. [9] in his study that uses GIS to organize police patrol routes, GIS software can solve the above problems by providing the graphical output, and it became the most popular tool for visualization of crash data and hotspots analysis.

Ruan et al. [13] proposed a solution on stochastic patrolling by dividing the nodes of interest, which may be cities and hotspots. These divided parts are called sectors. Each sector of the road network will be assigned to a patrol. Then, each patrol will follow the predefined route set on each sector or may respond to a trigger event which may alter the predefined route. In this setup, there are three subproblems needed to be solved before arriving to the final solution. These are partitioning the road network, finding an optimal route in a sector, and generating multiple patrolling routes.

Chawathe [3] created a tool for determining important patrol routes based on importance scores of locations and the topology of the road network. This is just a subproblem of solving routes in road networks. Often, route planning does not consider the actual events that could happen in the patrol environment because of the large number of possibilities that could affect the predefined route but knowing where the crime will focus may also matter. An example of this is the study by Paruchuri et al. [11] which focuses on security patrolling that considers uncertainty of the movement of the adversaries.

These are not the only available methods for finding near optimal solution to k-CPP. A well known algorithm for solving MM k-CPP is FHK-Algorithm by Frederickson, Hecht and Kim which uses the idea of route-first-cluster-second strategy. Its idea starts from solving the optimal CPP then partitioning the solution [5]. Ahr also created new constructive heuristics namely Augment-Merge algorithm and Cluster algorithm for solving MM k-CPP. They have also proposed exact algorithms for MM k-CPP by using Integer Programming formulations and a branch and cut algorithm [1].

Ahr and Reinelt [2] also published a tabu search algorithm for solving MM k-CPP. Its idea is to start with a solution that computes neighbour solutions to improve the objective function, which in the considered problem is minimizing the maximum tour. Most of their heuristics require shortest paths between nodes to be computed earlier, thus apply algorithms like Floyd-Warshall algorithm [4]. Having precomputed shortest paths will lead to easy path reconstruction from solutions.

Simulated annealing is very similar to the tabu search algorithm. Kirkpatrick et al [8] developed the simulated annealing algorithm by finding the connection between statistical mechanics and combinatorial optimization. By adapting the idea from condensed matter physics that annealing metals will improve certain properties, they simulated many scientific processes and properties involved in annealing like energy, temperature and applied it to solving combinatorial optimizations.

2. METHODOLOGY

The program is composed of 3 major parts which are:

1. Parsing of OSM XML data and creation of preliminary requirements.
2. Solving of routes using simulated annealing.
3. Rendering of maps with solution routes and preparing patrol tour reports.

Figure 1: OSM Map Snapshot

The OSM data of the university was downloaded and then cleaned to exclude roads that are not part of the university campus and to ensure that the resulting road network is connected.

3 important tags are considered in parsing the OSM XML data. These are:

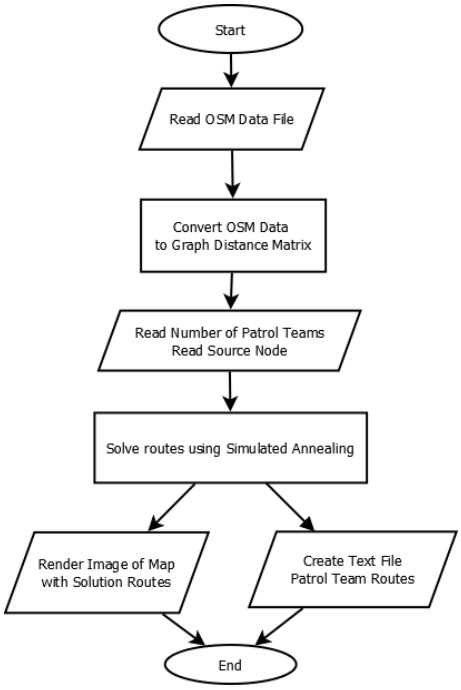

Figure 2: Program Flow

1. The *bounds* tag - This contains the minimum and maximum longitude and latitude of the rectangular region defined in the OSM XML data. This will define what tags to fetch from the OSM data file.

2. The *way* tag - This defines the regions in the OSM data and contains attributes that define whether a way is a highway, building, etc. Each Way tag is defined by the nodes that build its structure. The highway tag is used for all hardened and recognised routes between two places used by motorised vehicles, pedestrians, cyclists and others.

3. The *node* tag - This represents a point in the OSM data and the building block of OSM data. Each node tag contains an ID attribute which are referenced by way tags. Node tags contain the latitude and longitude coordinate of the node considered.

Once all the nodes considered are stored, a distance matrix of appropriate size to fit the count of nodes is created. Initially, the distance matrix is filled with values equal to infinity. Afterwards, each passable way in the OSM data is appended to an edgelist and distances between included nodes in the OSM way are used to update the distance matrix. Each passable OSM way is considered a single edge in annealing the solution. However, when computing the shortest path from a node to another, an OSM way is dissected into parts.

After building the required distance matrix, the shortest path is computed using *Floyd-Warshall All Pairs Shortest Path Algorithm* [4]. This is required for computing the cost of the tours and reconstructing the solution.

```
<osm version='0.6' generator='JOSM'>
  <bounds minlat='14.150599199999998'
    minlon='121.23288459999999'
    maxlat='14.170431599999999'
    maxlon='121.25587'
    origin='CGImap 0.0.2' />
  <node id='253795285'
    timestamp='2011-05-01T04:26:35Z'
    uid='44938' user='ianlopez1115'
    visible='true' version='10'
    changeset='8017532'
    lat='14.1641767' lon='121.2411449' />
  ...
  <way id='38349874' timestamp='2011-01-13T14:12:40Z'
    uid='44938' user='ianlopez1115'
    visible='true' version='2'
    changeset='6957500'>
        <nd ref='432196208' />
        <nd ref='1102056955' />
        ...
        <tag k='highway' v='service' />
        <tag k='service' v='driveway' />
  </way>
  <relation id='965401' timestamp='2010-06-17T12:24:55Z'
    uid='44938' user='ianlopez1115'
    visible='true' version='1' changeset='5008365'>
        <member type='way' ref='33541359' role='outer' />
        <member type='way' ref='62115417' role='inner' />
        <member type='way' ref='62115418' role='inner' />
        <tag k='type' v='multipolygon' />
  </relation>
</osm>
```

Figure 3: OSM XML Sample Format

$$E = \{e_0, e_1, e_2, e_3, ..., e_{n-2}, e_{n-1}, e_n\}$$

$$T = \left\{ \begin{array}{l} T_0 = \{e_0, e_{k+1}, ...\} \\ T_1 = \{e_1, e_{k+2}, ...\} \\ \vdots \\ T_k = \{e_c, e_{2k-1}, ..., e_n\} \end{array} \right\}$$

Figure 4: State Initialization and Edge Assignment

Once the preliminaries are done, the program proceeds to the solver. The solver uses simulated annealing to evolve a solution to become a better solution. A solution in simulated annealing is called a *state* in this problem. Let $G = (V, E)$ be the graph created from parsing the OSM data where $V = \{v_0, v_1, ..., v_m\}$ is the set of nodes in the graph and $E = \{e_0, e_1, ..., e_n\}$ is the set of edges required to be passed. Each edge is composed of a source node, v^s and destination node, v^d. Also, each edge has a corresponding weight w. The structure of each edge is $e_n = (v^s_{e_n} - v^d_{e_n}, w_{e_n})$, where an edge has its own source edge and destination edge, with its own weight. The required edges are sorted according to increasing distances.

The initial state is then created. First,the number of patrols, k, is required. Also, the universal source node $v^S \in V$ is required. Once all these values are already known, the set of tours $T = (T_0, T_1, ..., T_k)$ is initialized. Each edge from E is equally distributed to each tour in T.

```
begin
  InitialTemperature := 10^10
  CoolingRate := 0.999
  FreezingPoint := 2.23 × 10^-308
  BestSolution := InitialSolution
  BestEnergy := COST(BestSolution)
  t := InitialTemperature
  while t > FreezingPoint do
      NewSolution := NEIGHBOUR(BestSolution)
      NewEnergy := COST(snew)
      Δ := NewEnergy − BestEnergy
      if NewEnergy < BestEnergy ∨ random() < e^(Δ/t)
        BestSolution := InitialSolution
        BestEnergy := NewEnergy
      fi
      t := t × CoolingRate
  od
end
```

Figure 5: Simulated Annealing Pseudocode

Once the initial state is already built, the annealing of the state starts. The pseudocode of the annealing used is shown on Figure 5.

Simulated annealing is not just about minimizing the solution cost. It possesses a hill climbing technique that allows it to anneal solutions with a cost greater than the current best solution at a probability equal to $e^{\frac{\Delta}{t}}$ for it may produce a better neighbour.

The *neighbour* function used in annealing the states are done by altering some of the elements in the parameter state. It returns a new arrangement of edges which defines a new assignment of edges for each tour. There are 4 ways to create a neighbour of a state:

1. *Exchange edges between two tours* - Two tours in T are randomly chosen, and random edges in these tours are exchanged.

Initial State:

$$T = \left\{ \begin{array}{l} T_0 = \{e_0, e_2, ...\} \\ T_1 = \{e_1, e_3, ...\} \end{array} \right\}$$

Neighbour State:

$$T = \left\{ \begin{array}{l} T_0 = \{\mathbf{e_1}, e_2, ...\} \\ T_1 = \{\mathbf{e_0}, e_3, ...\} \end{array} \right\}$$

Figure 6: Sample exchange edge between two tours

2. *Invert traversal of edge in a single tour* - A single edge is randomly selected from a randomly selected tour in the set of all tours T. The traversal direction of that edge is reversed if the chosen edge is undirected.

3. *Transfer a single edge from a tour to another tour* - Two tours are randomly selected from the set of tours T. One of those tours will transfer a randomly selected edge to another selected tour.

Initial State:

$$T = \left\{ \begin{array}{l} T_0 = \{e_0, e_2, ...\} \\ T_1 = \{e_1, e_3, ...\} \end{array} \right\}$$

Neighbour State:

$$T = \left\{ \begin{array}{l} T_0 = \{e_2, ...\} \\ T_1 = \{\mathbf{e_0}, \mathbf{e_1}, e_3, ...\} \end{array} \right\}$$

Figure 8: Sample transfer a single edge from a tour to another tour

4. *Exchange position of two edges in a single tour* - A single tour is randomly selected from the set of tours T. Two edges from that tour are randomly selected and their positions are exchanged.

Initial State:

$$T = \left\{ \begin{array}{l} T_0 = \{e_0, e_2, ..., e_5, e_6, ...\} \end{array} \right\}$$

Neighbour State:

$$T = \left\{ \begin{array}{l} T_0 = \{\mathbf{e_6}, e_2, ..., e_5, \mathbf{e_0}, ...\} \end{array} \right\}$$

Figure 9: Sample exchange position of two edges in a single tour

Initial State:

$$T = \left\{ \begin{array}{l} T_0 = \{e_0, e_2, ...\} \end{array} \right\}$$

$$e_0 = v^s_{e_0} - v^d_{e_0}$$

Neighbour State:

$$T = \left\{ \begin{array}{l} T_0 = \{\mathbf{e^*_0}, e_2, ...\} \end{array} \right\}$$

$$e^*_0 = v^d_{e_0} - v^s_{e_0}$$

Figure 7: Sample invert single edge in a single tour

The *cost* function, $COST(T)$, determines the total energy of the current state which represents maximum cost of a tour in the set of all tours.

From the defined source node, each edge is traversed according to its ordering in the tours. Each edge is converted to its node form. This leads the edge list to become a series of nodes. Route from an edge to another is determined by the precomputed shortest path, $SP(v_s, v_d)$, earlier. Since k-CPP requires the source node and the destination node (depot) to be the same, it is appended to the start and end of the tour. Shortest paths to the depot are then computed.

After the tour is expanded, the cost is easily computed by getting the distance between two nodes from the distance

State:

$$T = \{\ T_0 = \{e_0, e_1, e_2\}\ \}$$

Edge expansion (edge list to node list):

$$T = \{\ T_0 = \{v^s_{e_0}, v^d_{e_0}, v^s_{e_1}, v^d_{e_1}, v^s_{e_2}, v^d_{e_2}\}\ \}$$

Shortest path computation for a single tour:

$$T_0 = \{v^S, SP(v^S, v^s_{e_0}), v^s_{e_0}, v^d_{e_0}, SP(v^d_{e_0}, v^s_{e_1}), v^s_{e_1}, ..., v^S\}$$

Figure 10: Expanding a single tour

matrix. After computing the cost of all the tours in T, the maximum cost will be the energy of the state.

As the temperature t decreases, the energy of the state decreases because of the continuous search for a more optimal neighbour from a considered state. This leads to achieving a near-optimal solution to MM k-CPP.

Map images are downloaded from *tile.openstreetmap.org*. The download URL is in the form of *tile.openstreetmap.org/z/x/y*, where z is the zoom level, x is the tile image index of a given longitude and y is the tile image index of a given latitude. Tile image index of a Mercator coordinate is computed as specified in Figure 11

$$x = \left\lfloor \frac{lon + 180}{360} \times 2^z \right\rfloor$$

$$y = \left\lfloor \left(1 - \frac{\ln\left(\tan\left(lat \times \frac{\pi}{180}\right) + \frac{1}{\cos\left(lat \times \frac{\pi}{180}\right)}\right)}{\pi} \right) \times 2^{z-1} \right\rfloor$$

Figure 11: Formula for computing tile image index of a given coordinate

Required tiles are downloaded and stitched into a single image. The solution will be rendered as a series of line segments in the considered map.

The program is designed to easily start the annealing process and customize input parameters depending on the choice of the user. Colors of the solution routes can be changed depending on the choice of the user. A panel which displays the map image can be seen which shows the preliminary solution image and the display to set the source location. Buttons to save the solution image and tour reports are available for convenience.

3. RESULTS AND DISCUSSION

The algorithm was first tested in the test graph defined by Ahr [1] to test his heuristics for solving routing problems. For ease of comparison of the implemented method to well known methods, the same graph was also used. The test graph is shown in Figure 13.

The simulated annealing approach implemented nearly matches the performance of other algorithms for MM k-CPP. Table

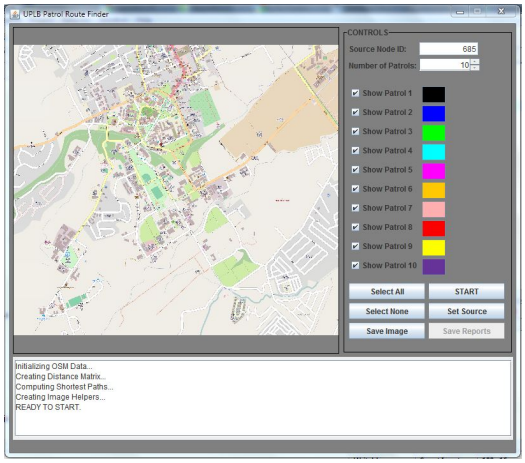

Figure 12: Program Snapshot

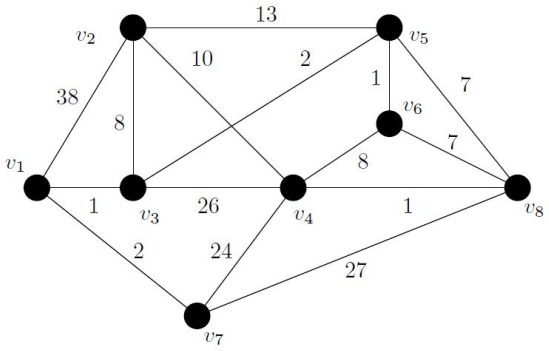

Figure 13: A connected weighted undirected test graph

1 displays the solutions for the input graph. The approach is compared to the FHK algorithm of Frederickson et. al. and the 3 other algorithms presented by Ahr.

Table 1: Comparison of Simulated Annealing (SA) to other known MM k-CPP methods on the small test graph (Figure 13) for $k = 2$

	$C(T_0)$	$C(T_1)$	$C_{max}(T)$
SA (10 Run Average)	94.6	95.0	95.3
FHK	89.0	98.0	98.0
Augment-Merge	101.0	107.0	107.0
Cluster	37.0	155.0	155.0
ClusterWeighted	117.0	93.0	117.0

Table 1 shows that the SA implementation results is not far from the other known methods as tested on the input graph Ahr created. It has actually outdone the methods laid out by Ahr in his dissertation with the test input in consideration. This cannot be tested further on other inputs since these methods are not implemented and those results are only shown on his publication.

As for the results on applying the SA approach on the actual patrol routes solving, Table 2 shows the values for $k =$

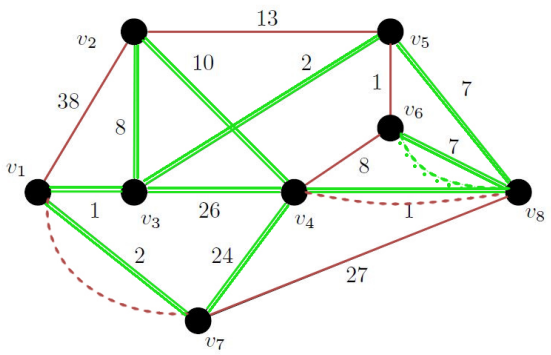

Figure 14: Sample result of simulated annealing on the test graph for $k = 2$ and v_1 as source node

$2, 3, 4, ..., 9, 10$ and source coordinates $14.1646, 121.2419$. The input used is the OSM data of the university road network. For each k, 10 runs of SA were performed and the results were averaged.

Table 2: Results of Simulated Annealing in OSM Data (10 Run Average)

k	C_{max}	σ
2	49848.80 m	7.0
3	34271.10 m	4.4
4	26416.80 m	10.0
5	21556.00 m	22.0
6	18047.50 m	33.0
7	15708.50 m	16.3
8	13835.30 m	19.4
9	12572.10 m	26.6
10	11474.10 m	28.0

Table 3: Absolute and Percentage Difference of Results of Simulated Annealing in OSM Data (10 Run Average)

k	C_{lb}	$C_{max} - C_{lb}$	$\lvert\frac{C_{max}-C_{lb}}{C_{lb}}\rvert$
2	18970.00 m	30878.80 m	1.63
3	12646.67 m	21624.43 m	1.71
4	9485.00 m	16931.80 m	1.79
5	7588.00 m	13968.00 m	1.84
6	6323.33 m	11724.17 m	1.85
7	5420.00 m	10288.50 m	1.90
8	4742.50 m	9092.80 m	1.92
9	4215.56 m	8356.54 m	1.98
10	3794.00 m	7680.10 m	2.02

In the Table 2, $C_{lb} = \frac{TotalCost(E)}{k}$ is defined as the division of the total cost of all edges $TotalCost(E) = \Sigma C(e_*)$ by the number of patrol teams k, where e_* denotes all the edges in E of $G = (V, E)$. Finding a solution with $C_{max} = C_{lb}$ is difficult since C_{lb} is not a tight lower bound. This value is just the lower bound of solution to MM k-CPP with $k = c$. Finding the real optimal solution is difficult because it means computing the exact solution for MM k-CPP through integer programming.

As Table 3 has shown, the task of dividing the tour cost is achieved because of the trend of C_{max} to decrease as k increases. These results may be affected by the fact that the results of the SA may be different from every run. However, this trend may be a close depiction of the trend of solutions for each k.

Deviation between individual tours in tour set for each k varies and has no trend. Increasing the number of tours, k, may decrease the absolute difference, which may mean lesser cost of tours but increases the percentage error, which may mean greater relative distance to optimal solution.

In the main program, 2 results can be produced, either a solution image or a tour report. Figure 15 shows a sample of a solution image for $k = 10$ and source coordinates equal to $14.1646, 121.2419$ with the solution trend.

Solutions are painted in colors chosen by the user. Default color settings are available. This image is just a simple depiction of the solution. Another solution form is the tour report which contain the tours by wayname and/or by coordinate form.

4. CONCLUSION AND FUTURE WORK

Solving near-optimal routes for multiple security patrols in the considered university road network is dependent on the approach for solving the problem model MM k-CPP. Different approaches are possible for solving MM k-CPP mostly involve heuristics. Nevertheless, the method proposed was successful in finding near-optimal routes for multiple security patrols in the university as an aide for security planning.

The simulated annealing method proposed in solving the MM k-CPP performs quite well for the near-optimal definition in the problem although further improvement of the method used is still possible. These include better neighbour solution generation, adaptive annealing structure and elite seeding.

Another possible future work is assessing the parallelizability of the input road network. Knowing whether the input is highly parallelizable will help the program to figure out the optimal number of patrols to use for that network.

Even though any OSM Map is accepted by the program, it is recommended to develop tools for the automated cleaning of the input OSM data and preparing it for the routing process.

The program implemented can be used as a start on planning security patrols in different universities or larger road networks.

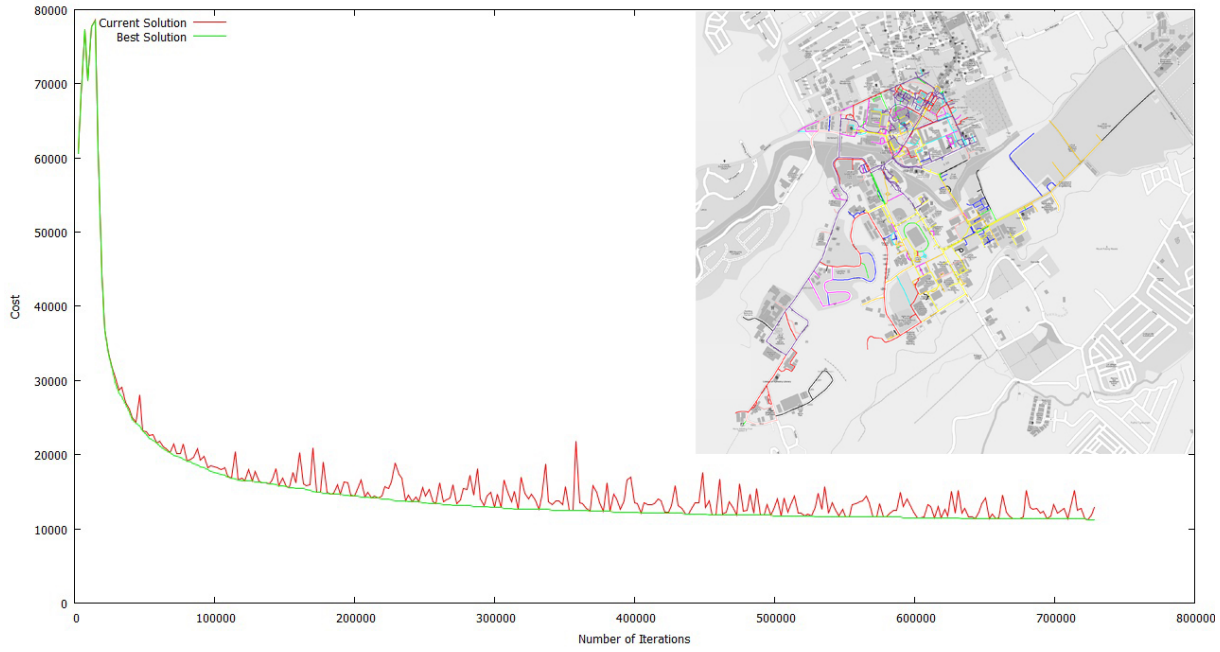

Figure 15: Solution Trend on Actual Data, inset is a sample solution for $k = 10$ patrols

5. REFERENCES

[1] D. Ahr. *Contributions to Multiple Postmen Problems*. PhD thesis, Rupercht-Karls-Universitat, Heidelberg, 2004.

[2] D. Ahr and G. Reinelt. A tabu search algorithm for the min-max k-chinese postman problem. *Comput. Oper. Res.*, 33:3403–3422, 2006.

[3] S. S. Chawathe. Organizing hot-spot police patrol routes. In *Proceedings of the 5th IEEE Intelligence and Security Informatics Conference*, pages 79–86. IEEE, 2007.

[4] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5:345, June 1962.

[5] G. Frederickson, M. Hecht, and C. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[6] G. Groves and J. Vuuren. Efficient heuristics for the rural postman problem. *ORiON*, 21(1):33–51, 2005.

[7] B. Huang, R. Cheu, and Y. Liew. Gis and genetic algorithms for hazmat route planning with security considerations. *INT. J. GEOGRAPHICAL INFORMATION SCIENCE*, 18(8):769–787, 2004.

[8] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[9] P. Kuo, D. Lord, and T. Walden. Using geographical information systems to organize police patrol routes effectively by grouping hot spots of crash and crime data. In *Third International Conference on Road Safety and Simulation*, 2011.

[10] A. Osterhues and F. Mariak. On variants of the k-chinese postman. *DiskussionsbeitrÂĺage des Fachgebietes*, 2005.

[11] P. Paruchuri, J. Pearce, M. Tambe, O. F., and S. Kraus. An efficient heuristic approach for security against multiple adversaries. In *AAMAS'07 Proceedings of the 6th international joint conference on Autonomous agents and multiagent system*, 2007.

[12] D. Reis, A. Melo, A. Coelho, and V. Furtado. Towards optimal police patrol routes with genetic algorithms. *LNCS*, 3975:485–491, 2006.

[13] S. Ruan, C. Meirina, F. Yu, K. R. Pattipati, and R. L. Popp. Patrolling in a stochastic environment. In *10th International Command and Control Research Symposium*, 2005.

[14] Y. Saruwatari and T. Matsui. A note on k best solutions to the chinese postman problem, 1993.

[15] S. Wohlk. *A decade of capacitated arc routing*. Operations Research/Computer Science Interfaces Series. 2008.