Optimizing Urban Traffic Network Design: A Multi-Objective Evolutionary Algorithm Approach.

Josh Bohde, BS in Computer Science

May 3, 2009

Abstract

A method is presented for solving both the discrete and continuous network design problem in a multiobjective manner with respect to urban traffic. A framework is presented using a multiobjective evolutionary algorithm and the traffic simulator Simulation for Urban Mobility (SUMO), using custom genetic operators. A sample network is optimized for both its discrete and continuous aspects, with comparisons drawn between the respective performance of the two. This framework is then applied to real world data drawn from a dataset of Cologne, Germany. These results are then used to show the feasibility of usage of the presented framework in a real world scenario.

Keywords

Network Design Problem, Evolutionary Algorithm, Multiobjective

Part I
Introduction

As urban areas grow, the aging road network is no longer able to handle the amount of traffic generated by the population, resulting in congestion and delays. These delays translate into a direct economic cost which is rising. A study of urban congestion in the United States shows that the congestion costs rose from $45.5 billion in 1995 to $78.2 billion in 2005, adjusted for inflation [6]. This congestion also resulted in 2.9 billion gallons of wasted fuel, and 4.2 billion hours in delays [6]. The proposed research aims to limit this congestion, while balancing a variety of limiting factors to the network. The economic burden placed on the tax payers will be taken into account, as well as car emissions and the likelihood of accidents, with the goal of minimizing these. A multi-objective approach will present a set of solutions that do not dominate each other, allowing for city planners to choose the most appropriate one, given their constraints.

The investigator will develop a method for optimizing the design of urban traffic networks. Important to urban traffic networks are the secondary characteristics of the network, such as the cost of the improvements to the network. Previous methods for solving this problem have involved applying weights to the values of these characteristics in ranking possible solutions. The proposed method will instead use a multi-objective evolutionary algorithm (MOEA) to provide a Pareto Optimal set of urban traffic networks. The objectives of this algorithm will be to maximize performance of the network, minimize cost of changes to the network, minimize emissions, and minimize accidents.

The investigator aims to improve upon previous related research in three ways. First, by using a multi-objective approach to optimize with respect to overall network performance and cost. Second, by using a traffic simulator in the fitness function in the evolutionary algorithm in order to provide fitnesses that are more applicable to real world situations. Third, scalability testing in order to demonstrate the feasibility of the proposed method in real world applications.

The inclusion of a traffic simulator presents a challenge in the form of high computational costs. This necessitates investigation into representations of road networks and respective genetic operators that generate valid solutions, while minimizing total evaluations.

Part II
Background

Urban traffic network design is a version of the network design problem (NDP). This problem is maximizing the amount of traffic on a network while minimizing the cost of the network, between a set of nodes representing places traffic can start or end, and links representing connections between nodes. There are two aspects to optimizing the performance of a network, discrete network design problem (DNDP) and continuous network design problem (CNDP). DNDP optimization is done by altering the discrete variables of the network, which is adding and subtracting links between nodes, which corresponds to adding roads and highways. CNDP optimization is achieved by altering the continuous variables, such as traffic light timings, speed limits, and number of lanes.

Because the number of links is exponential with regards to the number of nodes, the search space for DNDP is very large. With the inclusion of CNDP, the search space is even larger. The search space is not simple, as evidenced by Braess's paradox, which shows that adding capacity to a network can decrease the overall performance of the network. It was shown that for a random graph, the probability of Braess's paradox occurring is high[7]. This implies that when using stochastic algorithms to solve NDP, the search space is complex.

In a paper on urban road quality, the authors identify several other factors that affect the quality of the road network for the city inhabitants, and several restrictions for traffic network design [2]. Two objectives presented are the minimization of emissions and accidents. The effects of emissions are lowered as a result of lower speeds and traffic volume, with space between traffic and distance from residential areas also playing a part. Traffic accidents are less numerous with less traffic volume, and are less severe with lower speeds.

Part III
Related Work

Various search algorithms have been used in traffic network design. Search heuristers were used to improve the Sioux Falls traffic network, a CNDP, using a bi-level programming model and simulated annealing [5]. Fitness

was determined by the travel cost between each node in the graph. Genetic algorithms were shown to be optimize for CNDP [9], using a case study of network with 4 nodes. This algorithm employed a deterministic user traffic assignment to establish fitness. Genetic algorithms were shown to be comparable to simulated annealing for optimization of a CNDP using a real traffic network [1] These were also tested against hill climbing and tabu search heuristics.

Another paper presented a new formulation of the network design emphasizing robustness of the produced system in order to account for varying demands between nodes in the graph [8]. This formulation of the problem and the resulting genetic algorithm was shown to produce high quality solutions on example problems.

A MOEA is a type of evolutionary algorithm to optimize for one or more fitness values without reducing them to a single value. One solution is said to dominate another if its respective fitness values are no less than the other's, and at least one value greater than the other's respective value [10]. Instead of a single solution, a set of non-domination solutions are returned, called the Pareto-optimal front.

While there are many different formulations of MOEAs, a common problem of them is to have a large Pareto-optimal front, especially as the number of fitness values increases. -dominance limits the Pareto-optimal front by not allowing solutions that dominate each other less than to be included within the Pareto-optimal front [4]. Conceptually this divides the area of possibly accepted fitness into a grid, only allowing one solution in each grid.

Part IV
Methodology

1 Evolutionary Algorithm

The proposed MOEA is the -domination based multi-objective algorithm (-MOEA), as described in [3]. It was chosen based upon its property of ensuring diversity amongst the Pareto-optimal front, which is an important quality for this problem. From a parameter perspective, -MOEA is much like a standard MOEA, with the exception of its offspring size and values. Because -MOEA is a steady state algorithm, there is no need for an offspring size parameter. The algorithm also differs in the need for values for each objective.

2 Network Representation

The network is represented as a graph: a set of vertices and edges, with each edge connecting two vertices. The addition and deletion of edges from the network satisfies discrete aspect of the NDP.

To satisfy the continuous aspect, each edge has several properties affecting the performance of the individual edge. These include the shape of the edge, max speed, number of lanes, edge priority, and the direction of the edge.The priority of an edge determines how likely cars traveling on it are going to yield to cars on some intersecting edge. The max speed is the maximum speed that cars are allowed to travel. The number of lanes determines how many cars can be at the same progression on the edge. The direction of the edge determines whether or not the edge road on it is one way. Finally the shape of the edge determines the placement of the edge on the simulated area. This is represented as a set of points, consisting of floating points. The actual path of the road must visit each of these points in order before reaching the end point.

3 Genetic Operators

3.1 Recombination

Recombination of individuals is achieved by through a process analogous to uniform crossover for bit string representations. For each edge in both parents, the edge from the first parent is assigned to a randomly chosen child, with the remaining edge going to the remaining child. For each edge not in both parents, it is randomly assigned to a randomly chosen child. The exact procedure is given in Algorithm 1.

Algorithm 1 Network Recombination

for all $edge \in parent_1\_edges \cup parent_2\_edges$ **do**
  Pick a random integer r uniformly from $[0, 1]$
  if $edge \notin parent_2\_edges$ **then**
    $child_r\_edges \Leftarrow child_r\_edges \cap edge$
  else
    $child_{(1-r)}\_edges \Leftarrow child_{(1-r)}\_edges \cap$ ($edge$ in $parent_1\_edges$)
    $child_r\_edges \Leftarrow child_r\_edges \cap$ ($edge$ in $parent_2\_edges$)
  end if
end for

3.2 Mutation

3.2.1 Discrete

Mutation of networks is achieved by randomly adding or deleting edges. For every child generated, it is mutated one time. The exact mutation procedure is described in Algorithm 2.

Algorithm 2 Discrete Mutation

$n \Leftarrow$ number of elements in *vertices*
Pick two random integers j, k uniformly from $[0...n]$ where $j \neq k$
Generate an edge e connecting $vertices_j$ and $vertices_k$
if $e \notin edges$ **then**
  $edges \Leftarrow edges \cap e$
else
  $edges \Leftarrow edges - e$
end if

3.2.2 Continuous

Continuous mutation is done by altering individual edges. The aspects modified are edge shape, number of lanes, lane spread, and edge priority. Edge shape is modified by taking the current edge shape, choosing a point on it, and randomly moving it. For edges in the original graph, edge shape cannot be modified. For other edges, the chance to mutate this is user defined. This algorithm is described in Algorithm 3.

Algorithm 3 Edge Shape Mutation

shape is the ordered set of points making up the shape of the edge.
Pick a random real number r uniformly from $[0, 1]$.
if $r < chance\_to\_mutate$ **then**
  Pick a random integer n uniformly from $[1, n-1]$ where n is the number of elements in *shape*
  $H \Leftarrow$ the hyperbolic function created by $shape_n$ and $shape_{(n+1)}$
  Pick a random real number t uniformly from $[0, 1]$.
  $p \Leftarrow H(t)$
  Pick two random real numbers x_r, y_r uniformly from $[0, 1]$.
  $p \Leftarrow p + (x_r * max\_delta\_x, y_r * max\_delta\_y)$
  Insert p into *shape* between $shape_n$ and $shape_{(n+1)}$
end if

With a user-defined chance, the number of lanes is mutated according by adding or subtracting a random number of lanes, up to a user-defined maximum (max delta). For edges in the original graph, this cannot be altered. The algorithm is described in Algorithm 4.

Algorithm 4 Lane Number Mutation

Pick a random real number r uniformly from $[0, 1]$.
if $r < chance\_to\_mutate$ **then**
  Pick a random integer l uniformly from $[-max\_delta, max\_delta]$
  $lanes \Leftarrow lanes + l$
  $lanes \Leftarrow max(1, lanes)$
end if

With a user-defined chance, the lane spread is mutated by randomly choosing one of the three possibilities, right, center, or left. The algorithm is described in Algorithm 5.

Algorithm 5 Lane Spread Mutation

Pick a random real number r uniformly from $[0, 1]$.
if $r < chance\_to\_mutate$ **then**
  Pick a random integer s uniformly from $[-1, 1]$
  $spread \Leftarrow s$
end if

With a user-defined chance, the number of lanes is mutated according by adding or subtracting an a randomly generated number to the edge's priority, up to a user-defined maximum (max delta).The algorithm is described in Algorithm 6.

Algorithm 6 Edge Priority Mutation

Pick a random real number r uniformly from $[0, 1]$.
if $r < chance\_to\_mutate$ **then**
  Pick a random integer p uniformly from $[-max\_delta, max\_delta]$
  $priority \Leftarrow priority + p$
  $priority \Leftarrow max(0, priority)$
end if

4 Fitness Measure

The fitness is a tuple, consisting of the network performance and cost.

4.1 Network Performance

Common to all of the previous research is the usage of mathematical models to emulate the traffic of the network. While this is useful for determining the theoretical maximum performance of a network, it may not accurately reflect the performance of a real urban network. To better model the performance of a real network, a traffic simulator is proposed to generate the characteristics of the network from which the fitness will be derived. The planned traffic simulator is Simulation of Urban Mobility (SUMO) [8]. SUMO is a microscopic, space-continuous traffic simulator, developed by the Institute of Transport Research at the German Aerospace Centre. Its microscopic model emulates traffic on a per-car basis, while keeping computation time low. This simulator will provide details of network performance, vehicle emissions, and network safety. In order to test the performance in the simulation, the user must supply a flow file, describing a set of flows, composed of a start edge, and ending edge, a beginning and end time, and the number of times to repeat the route. From this, the program can generate specific routes for evaluating the network. Network performance is measured by compiling the edges generated by the algorithm into a traffic network, and running the simulation. From the simulation output, the average time for a car to reach its destination journey will be measured. This is used as the measure for network performance.

A flowchart is shown in Figure 4.1, detailing the exact procedure on network evaluation.

4.2 Network Cost

Network cost is the sum of the land cost, building costs, and intersection costs of the network.

4.2.1 Land Cost

Land cost is determined by a supplied to the algorithm, giving the prices for land at a given interval. Using a ray tracing algorithm, each edge produces a corresponding inhabitance matrix. This algorithm is described in Algorithm 7.

Algorithm 7 Edge Intersect Matrix

$I_{m \times n}$ is the intersect matrix, initialized as the zero matrix.
$t \Leftarrow 0$
j is the initial x value of the first point in I.
k is the initial y value of the first point in I.
$I_{j,k} \Leftarrow 1$
while $t < 1$ **do**
  t_y is the distance along the y-axis to the next element in the matrix, along the edge.
  t_x is the distance along the x-axis to the next element in the matrix, along the edge.
  if $t_y < t_x$ **then**
    $t \Leftarrow t + t_y$
    $k \Leftarrow k + 1$
  else
    $t \Leftarrow t + t_x$
    $j \Leftarrow j + 1$
  end if
  $I_{j,k} \Leftarrow 1$
end while

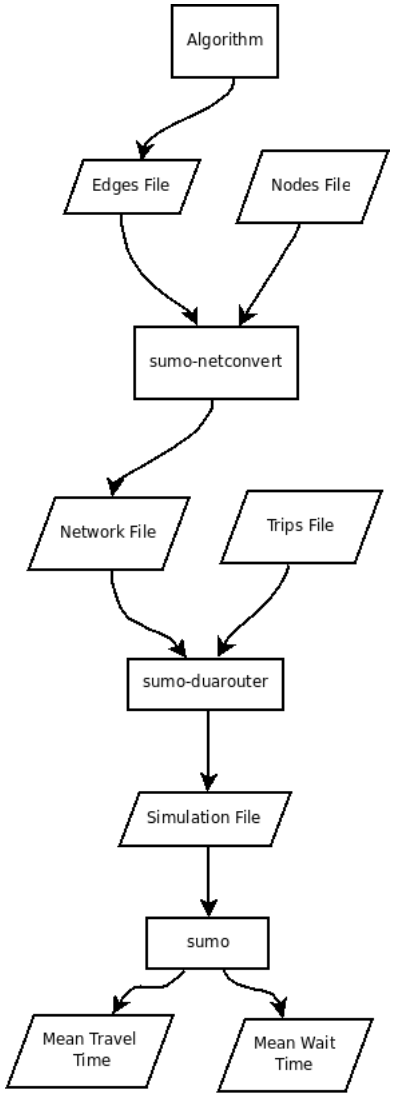

Figure 1: Flowchart for Network Performance Evaluation

For each edge in the network, a network inhabitance matrix is produced, where each element is the bitwise or of the corresponding elements in each edge inhabitance matrix. This algorithm is described in Algorithm 8.

The overall land cost is determined by summing the multiplication of corresponding elements of the cost matrix and network inhabitance matrix. This algorithm is described in Algorithm 9

Algorithm 8 Network Intersect Matrix

$I_{m \times n}$ is the intersect matrix, initialized as the zero matrix.
for all $i \in edge\_intercept\_matrices$ **do**
  for $j = 1$ to m **do**
    for $k = 1$ to n **do**
      $I_{j,k} \Leftarrow max(0, i_{j,k})$
    end for
  end for
end for

Algorithm 9 Network Land Cost

$I_{m \times n}$ is the intersect matrix, containing the network intercepts
$C_{m \times n}$ is land cost matrix
$r = 0$
for $j = 1$ to m **do**
  for $k = 1$ to n **do**
    $r \Leftarrow r + I_{j,k} \times C_{j,k}$
  end for
end for
return r

4.2.2 Building Costs

Building costs require two user set parameters, the cost of road per unit per lane and cost of intersection. The building cost is the sum of all costs of intersections and the sum of the road costs for each edge. The number of intersections are determined when establishing the edge intersection matrices. The road costs for each edge is determined by finding the lengths edge, and multiplying that by the number of lanes in the edge, and the cost of road per unit per lane.

Part V
Experimental Setup

1 Comparison of Discrete and Continuous Network Design Problem

The goal of this experiment was to determine whether optimizing for the CNDP as opposed to the DNDP provided a benefit in the quality of solutions contained in the Pareto-optimal front that offset the added complexity to the search space. To do so, the algorithm was run on a randomly generate dataset, with the continuous mutations turned off for one, and on for another. In Table 1, the values common to both tested algorithms are given. Values for the specific algorithms are in Table 1.

Number of Runs	Evaluations	Initial Population	Fitness Epsilons
30	1000	50	(.1, .1, .1)

Table 1: Common Experiment Values

Attribute	Continuous	Discrete
Shape Mutate Chance	5%	0%
Shape Mutate Delta	1	0
Priority Mutate Chance	5%	0%
Priority Mutate Delta	1	0
Lane Mutate Chance	5%	0%
Lane Mutate Delta	2	0
Spread Mutate Chance	1%	0%

Table 2: Individual Experiment Values

2 Testing of Continuous Network Design Problem on Real World Data

In order to demonstrate the feasibility of using the algorithm for real world applications, it was tested on a subset of the freely available data of Cologne, Germany. The algorithm specific values used for testing are shown in Table 2.

Runs	5
Initial Population	50
Evaluations	1000
Fitness Epsilons	(.1, .1, .1)
Shape Mutate Chance	5%
Shape Mutate Delta	1
Priority Mutate Chance	5%
Priority Mutate Delta	1
Lane Mutate Chance	5%
Lane Mutate Delta	2
Spread Mutate Chance	1%

Table 3: Individual Experiment Values

Part VI
Results

1 Comparison of Discrete and Continuous Network Design Problem

The final nondominating set from all of the runs for the continous experiment is shown in Table 1, with the set from the discrete experiment in Table 1.

17.0	0.0	107998.690502
16.0	0.0	243551.39066
18.0	0.0	23350.3471944

Table 4: Continuous Nondominated Set

19.0	1.5	21944.0
24.67	1.0	23749.0

Table 5: Discrete Nondominated Set

A graph showing all the nondominated individuals over all the runs from the continuous dataset are shown in Figure 1. A similar graph is shown in Figure 1.

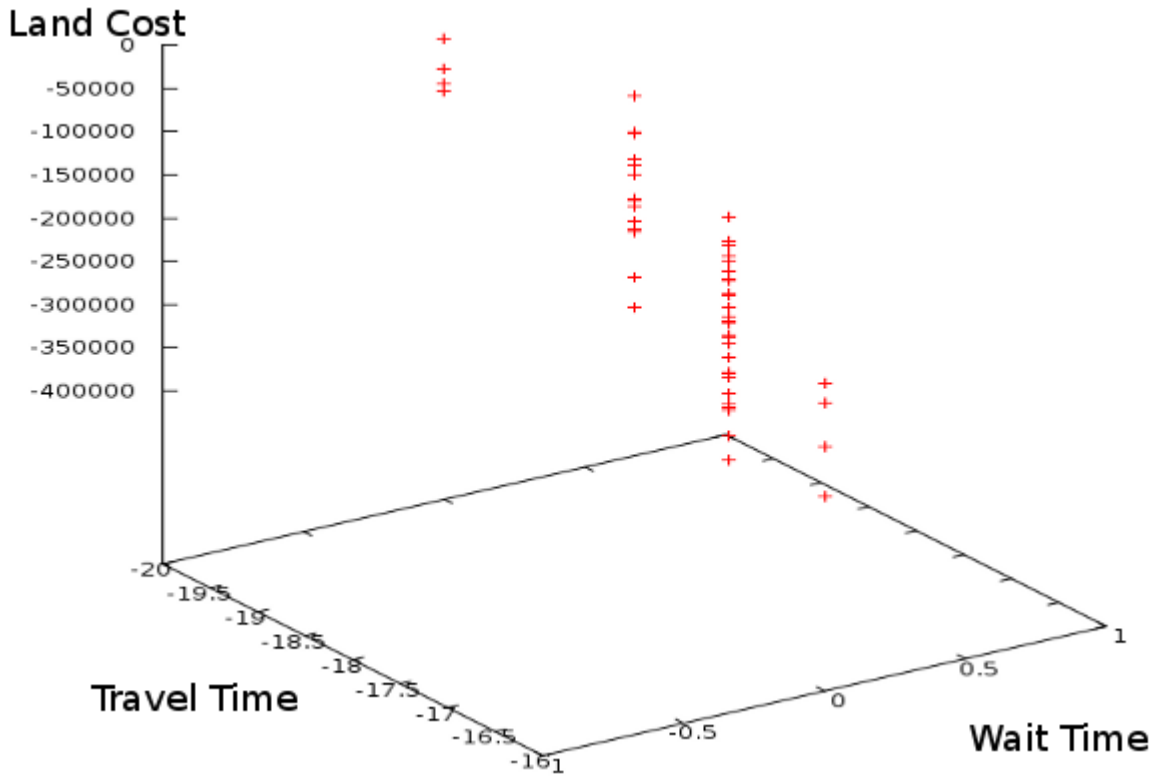

Figure 2: Continuous Nondominated Individuals

2 Testing of Continuous Network Design Problem on Real World Data

Forthcoming.

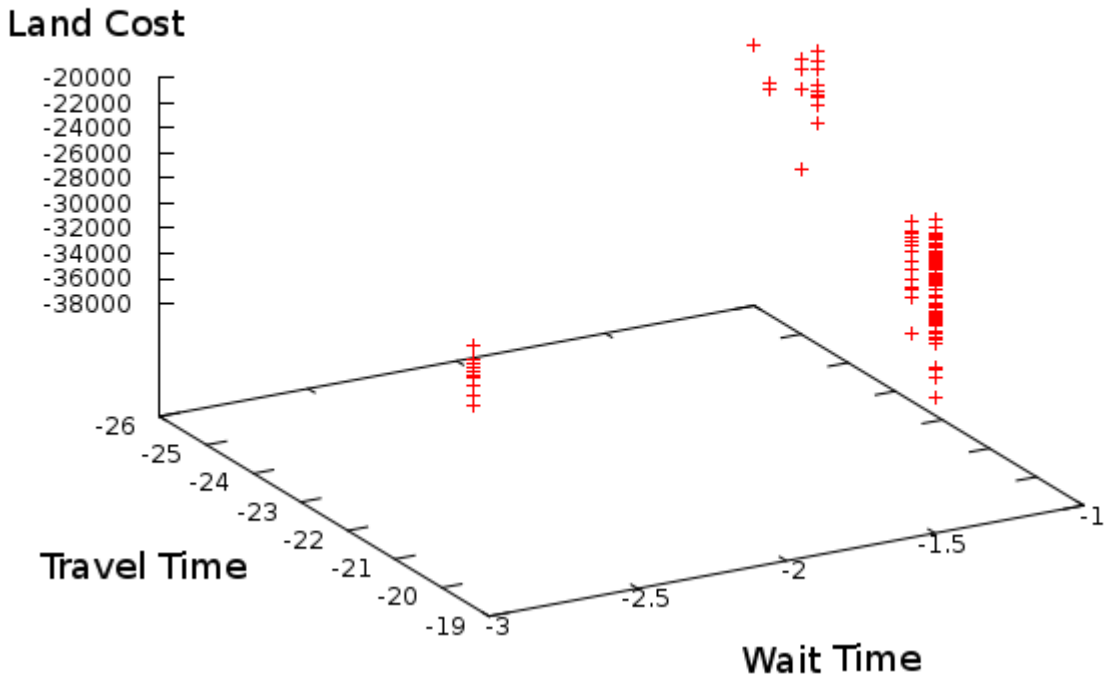

Figure 3: Discrete Nondominated Individuals

Part VII
Discussion

1 Comparison of Discrete and Continuous Network Design Problem

The results presented show that it the given method does provide a set of nondominating solutions representing an improvement over the original urban network, for both the DNDP and CNDP.

With the results presented demonstrate that solving for the continuous aspects of the network can provide better solutions than solving for the discrete aspect alone. This possibility is gained at the expense of an expanded search space. The most notable of the improvements that the CNDP al-

gorithm offered over the DNDP algorithm is that the CNDP was able to eliminate wait time by shifting avenues of traffic away from each other, preventing intersections, which results in stops.

2 Testing of Continuous Network Design Problem on Real World Data

Forthcoming.

Part VIII
Conclusion

The framework presented here is an effective method for solving both the DNDP and CNDP for urban traffic networks, while presenting the user with more options than producing a weighting function and a single objective approach would afford. Ultimately, this algorithm can assist city planners with designing urban areas. While the computation time for the simulation can be quite high, the time frame for such projects should allow for adequate computation time.

Part IX
Future Work

The framework should be expaned to handle more fitness measures, adding to its usefulness for city planners. The amount of pollution caused by traffic and how prone to accidents the roadway is should be a priority. To expand upon this framework's application in real world situations, a method should be devised to allow for two way dialog between the user and the program, allowing the program to suggest upgrades that can be made now, while the user can update the progress of the corresponding real network.

References

[1] G.E. Cantarella, G. Pavone, and A. Vitetta. Heuristics for urban road network design: Lane layout and signal settings. *European Journal of Operational Research*, 175(3):1682 – 1695, 2006.

[2] Licinio da Silva Portugal and Leonardo Amorim de Araújo. Procedure to analyze the performance of urban networks in brazilian cities. *Journal of Urban Planning and Development*, 134(3):119–128, 2008.

[3] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Evaluating the -domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation*, 13(4):501–525, 2005. PMID: 16297281.

[4] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002. PMID: 12227996.

[5] Qiang Meng and Hai Yang. Benefit distribution and equity in road network design. *Transportation Research Part B: Methodological*, 36(1):19 – 35, 2002.

[6] David L. Schrank and Timothy J. Lomax. The 2007 urban mobility report. 2007.

[7] Greg Valiant and Tim Roughgarden. Braess's paradox in large random graphs. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 296–305, New York, NY, USA, 2006. ACM.

[8] Yafeng Yin, Samer M. Madanat, and Xiao-Yun Lu. Robust improvement schemes for road networks under demand uncertainty. *European Journal of Operational Research*, 198(2):470 – 479, 2009.

[9] Guoqiang ZHANG and Jian LU. Genetic algorithm for continuous network design problem. *Journal of Transportation Systems Engineering and Information Technology*, 7(1):101 – 105, 2007.

[10] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000. PMID: 10843520.

Appendices

A Code Repository

http : *//github.com/joshbohde/mondp/*