Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

An evaluation of space partitioning methods and meta-heuristics based graph partitioning methods for partitioning road network simulations

Aravind Vasudevan
David Gregg

School of Computer Science and Statistics
Trinity College Dublin

Quentin Bragard
Anthony Ventresque

Computer Science Department
University College Dublin

ABSTRACT

Abstract goes here.

1 MOTIVATION

Introduction goes here.

Our **key contributions** in this paper are:

-
-
-
-

The rest of the paper is arranged as follows.

2 RELATED WORK

3 FORMALIZATION OF THE PROBLEM STATEMENT

In this paper we conduct a statistically significant comparison of different methods of partitioning a road network graph based on some metrics. This section defines what a road network graph is and presents the formal definition of the metrics.

3.1 The Road Network Graph

The road network of a city can be represented by a directed cyclic graph (Holden and Risebro 1995) $G(V,E)$ where V denotes the vertex set and E denotes the edge set. Every edge $e_{ij} \in E$ in the graph represents a unidirectional road in the city that connects intersection v_i to intersection v_j. Every vertex $v_i \in V$ denotes an intersection of two or more roads. A weight w_{ij} is associated with the edge e_{ij} that is representative of the traffic that flows through that road. As discussed in Section 6, we intend to increase the number of weights that can be associated with every edge to be able to represent the number of lanes, length of the road, importance of the road etc. as part of our future work.

3.2 Partitioning a graph

For the sake of completeness of the graph definition, we also assign weights to the vertices denoted by W_i, which is defined as follows :

$$W_i = \sum_{\forall j \in neighbours(i)} w_{ij} \tag{1}$$

where $neighbours(i)$ is the set of all nodes in V that receive an outgoing edge from v_i. We define a partitioning scheme on the given road network G, ζ_G as a function that maps every node v_i from the road network graph to a partition.

$$\zeta_G(v_i) \in \{0, \ldots, p\}, \forall v_i \in V \tag{2}$$

The problem of partitioning a graph has a 2 dimensional space, where one axis represents the node ID(representing the intersection) and the other represents the partition ID. Each point in this 2-D space represents an $\{intersection,partition\}$ pair which implies this intersection is mapped onto this partition. We define a "state" to be a collection of $|V|$ points such that each intersection is mapped to exactly one partition. The total number of possible "states" in this discrete space is $|V|^p$, where p is the number of required partitions.

3.3 Formalizing the objective function

The objective of our framework is to minimize the total application latency as described in Equation (**??**).

In the next section, we present an overview of the methods that are compared under the purview of road network partitioning.

4 META HEURISTICS BASED GRAPH PARTITIONING ALGORITHMS

4.1 Simulated Annealing

Input: Initial Mapping ζ_0 and Starting and Final Temperatures $\mathscr{T}_0$, $\mathscr{T}_f$
Output: Best Mapping ζ_{best}
$\zeta_{current} \leftarrow \zeta_0$;
$C_{current} \leftarrow OBJECTIVE\_FUNCTION(\zeta_0)$; //calculate initial objective function value;
$\zeta_{best} \leftarrow \zeta_{current}$;
$C_{best} \leftarrow C_{current}$;
for $i \leftarrow 0$ *to* ∞ **do**
  $\mathscr{T}_{current} \leftarrow NEXT\_TEMPERATURE(\mathscr{T}_0, i)$;
  $\zeta_{new} \leftarrow NEXT\_STATE(\zeta_{current}, \mathscr{T})$;
  $C_{new} \leftarrow OBJECTIVE\_FUNCTION(\zeta_{new})$;
  $\Delta C \leftarrow C_{new} - C_{current}$;
  $r \leftarrow RAND()$;
  $p \leftarrow ACCEPTANCE\_PROBABILITY(\Delta C, \mathscr{T}_{current})$;
  if $\Delta C < 0$ *or* $r < p$ **then**
    if $C_{new} < C_{best}$ **then**
      $\zeta_{best} \leftarrow \zeta_{new}$; $C_{best} \leftarrow C_{new}$; $\zeta_{current} \leftarrow \zeta_{new}$; $C_{current} \leftarrow C_{new}$;
    end
  end
  else
    if $\mathscr{T}_{current} \leq \mathscr{T}_f \| executionTime \geq maxTime$ **then**
      break
    end
  end
end
return ζ_{best}

Algorithm 1: The Simulated Annealing Algorithm

Simulated Annealing (Kirkpatrick, Vecchi, et al. 1983) is an adaptation of the Metropolis-Hastings algorithm for solving the problem of locating a good approximation of the global optimum of a given function, $\mathscr{F} : \mathbb{R} \to \mathbb{R}$, which has a large search space. This large number of states, as discussed in Section 3.2, makes exhaustive enumeration to find optimal solutions, not feasible. Please note that we will use the term "vertex" and "intersection" interchangeably.

SA is a heuristic algorithm that explores the search space by inspecting one valid state at each iteration. Each of these inspected states are evaluated by an "objective function" which tells us how "good" or "bad" this state is. The "goodness" in an SA algorithm is problem dependent and in our case it is given by the metric defined in Equation **??**, Section 3.3. The algorithm progresses by inspecting a candidate state at each iteration and it either accepts it as its current state or discards the state and "moves" on to another state. We define a move as the generation of the next candidate state and this progress is governed by a global time-varying parameter called the "temperature" which changes based on an "annealing schedule".

The algorithm always accepts a move to a better solution, i.e. a new state which has a "better" objective function value than the current state. When this value is worse however, the SA algorithm accepts this move with a certain "acceptance probability", that changes with the current temperature. When the temperature is high, the algorithm accepts moves to a worse solution with a higher probability; as the temperature reduces over time, this probability decreases as well.

4.1.1 Optimizations

4.2 Genetic Algorithm

Choose an initial random population of individuals
Evaluate the fitness of the individuals
while *Termination criteria not met* **do**
  Select the *best* "n" individuals to be used by the genetic operators
  Generate new offsprings by using the genetic operators
  Evaluate the objective function value for these offsprings
  Replace the *worst* "k" individuals of the current population with the best "k" individuals from the offsprings
end

Algorithm 2: The Genetic Algorithm

Genetic Algorithm is a heuristic search{cite core paper} algorithm that mimics the natural selection process. This heuristic algorithm is often used{cite usage in general context} for solving optimization problems with large search spaces and it belongs to a larger class called Evolutionary Algorithms.

In the context of the problem at hand, we follow the definition for the search space from Section 3.2, as a 2-dimensional space with one axis representing the intersections and the other the partition IDs. Hence each point in the space becomes a $\{intersection,partition\}$ pair.
To characterize a genetic algorithm, we require two things :

- A **genetic representation** of the solution space as shown in Figure **??**
- An **objective function** to evaluate solutions as discussed in Equation **??** from Section 3.3

Once this is fixed, we initialize the algorithm by generating a set of random solution vectors and assigning that as the initial population. The initial population size is problem specific and in our case it is **FILLINVALUEFORPOPSIZE**. We have chosen to generate the initial population randomly as opposed to generating it with some seed, to cover a wider range of candidates from our large search space. We then apply a set of genetic operators as discussed in Section 4.2.1 to generate the next population. This process is repeated and the "n" best solutions are retained at every step. We terminate the algorithm once a fixed time has passed by. Algorithm 2 gives an algorithmic overview of the above mentioned process.

4.2.1 Genetic Operators

Although there are a lot of genetic operators to choose from{cite core paper and some new papers}, we employ the **mutation** and **crossover** operators for the purpose of this comparison

- **mutation** -
- **crossover** -

5 EXPERIMENTS AND RESULTS

6 FUTURE WORK

- One of the limiting factor in this comparison is the limited representation of the road network. We plan to extend our model by allowing edges and vertices in our road network graph to have more than one weight associated with them. In doing so, we can allow for a finer and more accurate representation of the road network which in turn allows us to partition the graph better.
- In the modified move function, we currently move edges(the neighbour correspoding to the edge) if it has a weight which is 2σ away from μ. This is under the assumption that traffic is normally distributed across the edges. One of the improvements that we could do is to fit the distribution of traffic across edges and move edges according to this regressively found curve that fits the traffic distribution.

7 CONCLUSION

In this paper we have outlined the comparison of space and graph partitioning techniques in the context of partitioning road network graphs for simulation. We have discussed

ACKNOWLEDGMENTS

This work is partly funded by the IRC Enterprise Partnership Scheme in collaboration with IBM Research, Dublin, Ireland.

A APPENDICES

Place any appendices after the acknowledgments and label them **A**, **B**, **C**, and so forth.

REFERENCES

Holden, H., and N. H. Risebro. 1995. “A mathematical model of traffic flow on a network of unidirectional roads”. *SIAM Journal on Mathematical Analysis* 26 (4): 999–1017.

Kirkpatrick, S., M. Vecchi et al. 1983. “Optimization by simmulated annealing”. *science* 220 (4598): 671–680.

AUTHOR BIOGRAPHIES

ARAVIND VASUDEVAN
QUENTIN BRAGARD
ANTHONY VENTRESQUE
DAVID GREGG.