Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2012) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 1st International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications

Exploiting Graph-theoretic Tools for Matching and Partitioning of Agent Population in an Agent-based Model for Traffic and Transportation Applications

Daniel Keren[a]*, Ansar-Ul-Haque Yasar[b], Luk Knapen[b], Sungjin Cho[b],Tom Bellemans[b], Davy Janssens[b], Geert Wets[b], Assaf Schuster[c], Izchak Sharfman[c]

[a]*Department of Computer Science, Haifa University, Haifa 31905 (Israel)*
[b]*Transportation Research Institute (IMOB), Hasselt University, Wetenschapspark 5 bus 6, 3590 Diepenbeek (Belgium)*
[c]*Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa 32000 (Israel)*

Abstract

In this position paper, we exploit the tools from the realm of graph theory to matching and portioning problems of agent population in an agent-based model for traffic and transportation applications. We take the agent-based carpooling application as an example scenario. The first problem is *matching*, which concerns finding the optimal pairing among agents. The second problem is *partitioning*, which is crucial for achieving scalability and for other problems that can be parallelized by separating the passenger population to sub-populations such that the interaction between different sub-populations is minimal. Since in real-life applications the agent population, as well as their preferences, very often change, we also discuss incremental solutions to these problems.

Keywords: Carpooling, Scalability, Agent-based, Modeling, Matching, Partitioning, Graph theory.

1. Introduction

The well-known ***carpooling application*** concerns the assignment or matching of passengers(agents) so they may share a ride and as a result reduce travel costs, fuel, toll / parking costs, emissions and the

* Corresponding author. Tel.: (+972) 4-824-9730; Fax: (+972) 4-824-9331.
E-mail address: dkeren@cs.haifa.ac.il.

overall traffic load. In this paper we shall deal only with the restricted problem of pairing two agents. Typically, every two agents are assigned (either by a coordinator or according to their own decision) some compatibility measure, which is a positive number that determines their preference to ride together. This measure can depend on many factors such as the proximity of residence and workplace of the agents, their personal considerations / preferences, schedule compatibility, etc. One way to determine this measure is for each potential carpooler to upload his/her data to a website or to an online service, allow a coordinator to automatically determine it based on parameters as discussed earlier and then the passengers(agents) can update the results. Note that the compatibility measure is not necessarily commutative (e.g. agent A may want to ride with B, but not vice-versa), and it should be made commutative (by taking the average, or minimum, of both corresponding measures).

In this position paper we advocate the tools from the realm of graph theory to matching and portioning problems of agent population in an ***agent-based model*** for traffic and transportation applications. An *agent-based model* is a class of computational models for simulating the actions and interactions of autonomous agents with a view to assessing their effects on the systems as a whole [9]. We take the agent-based carpooling application as an example scenario. The first problem is ***matching***: the input is the list of agents and their compatibilities, and the output is a (not necessarily complete) partitioning of the agents to disjoint pairs. Typically, this partitioning attempts to maximize the sum of compatibility measures of the pairs; following this, the agents may be allowed to reject their partners, or possibly modify their preferences and have the coordinator run another matching.

The second problem, ***partitioning***, concerns scalability. Since all solutions to the matching problem run in super-linear time, solving it for a large number of agents may be prohibitive – even more so since the agents' preference may change rapidly, and it may be required to run the matching very often. A provably advantageous paradigm for solving such large-scale problems is to "break them up" into smaller sub-problems, and solve the sub-problems in parallel. To achieve this, the sub-problems should be as *independent* as possible. Here, independence means that the agent population can be split into a disjoint union of sets, such that for each two different sets, the overall sum of the compatibility measures taken over all agents pairs such that one agent is in the first set and the other in the second set is as small as possible. After this stage is completed, the matching is run in each subset independently.

Another important goal is to find an ***incremental*** solution to these problems. As noted before, the agent population – and their preferences – are highly dynamic; i.e. people either retire or join the work market, and they change their workplace and address, as well as their preferences for the carpool partner. It is, of course, inconceivable to re-run the matching and partitioning every time such a change occurs; it is necessary to apply an incremental algorithm, which will both identify when it is required to rerun the two processes, and perform the task quickly, using the previous solution and relying on the fact that typically the changes are relatively small.

2. Graphs and their Relation to the Agent-based Carpooling Application

Reminder: a *graph G* consists of a set of nodes, *V*, and a set of edges, *E*, such that each edge is associated with a pair of nodes. Denote $G = (V,E)$. A directed graph is the same as above, but where each edge is associated with an ordered pair of nodes. In a weighted graph, each edge has an associated real number with it, defined as its weight.

For the agents *matching* problem in an agent-based carpooling application, define a weighted graph whose nodes are the potential agents, and there is an edge between any two agents who are potential

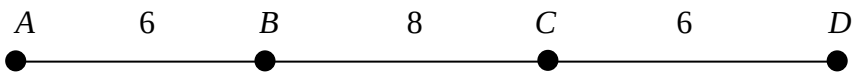
partners for carpooling. The edge weights are all non-negative, and the larger the weight, the higher the preference of the two corresponding agents to carpool. There are various ways in which to define the weight; at first the parameters which influence the weight should be chosen. One important parameter is the amount of fuel (or distance, etc.) saved when two agents carpool; the other is their personal preference, which may depend on various factors, e.g. how is it easy for the driver to pick up his/her partner, how well do they get along, etc. The second parameter may be allowed to obtain negative values. The overall weight is a combination of these parameters.After the graph has been constructed, we seek an optimal *matching* in it, which will correspond to the assignment of the desired carpooling. Some definitions relating to matching follow.

2.1 Matching and the carpooling problem

- Given a graph *G = (V,E)*, a *matching M* in *G* is a set of pairwise non-adjacent (disjoint) edges – that is, no two edges share a common node.
- A node is *matched* (or *saturated*) if it is an endpoint of one of the edges in the matching. Otherwise the node is *unmatched*.
- The *weight* of a matching *M* is the sum of the weights of the edges in *M*.
- A *maximum (optimal) matching* is a matching such that it obtains the maximum weight of all matchings. It does not have to be unique.
- If all the edge weights are equal, the problem reduces to finding a matching with a maximal number of edges (which here is equivalent to finding a carpooling scheme which maximizes the number of carpoolers). This problem is referred to as *maximum cardinality matching*, which is easier to solve than the general problem. It corresponds to the case in which the only preference for carpooling partners is *binary* (i.e. does agent *A* agree to ride with agent *B* or not).

Ideally, after the graph corresponding to the carpooling is constructed, we recover a maximum matching and then assign the carpooling according to it – that is, two agents will carpool *iff* the edge connecting them is in the matching. Note that no inconsistency may result from this assignment, as the edges are disjoint. Note, also, that some agents may be left without partners.

Finding the optimal matching is non-trivial; as the following simple example demonstrates, a greedy algorithm (which commences by choosing an edge with a maximal weight and then continues by choosing an allowable maximal weight edge among those which remain), may not reach an optimal solution:



The greedy solution here is to start by choosing the edge between *B* and *C*; however, that will constitute the entire matching, as *A,D* will be left without partners. It is obvious, however, that a maximum matching is given by edges $\overline{AB}, \overline{CD}$.

A special case of the matching problem is when the graph is *bipartite* – that is, its nodes can be partitioned into two disjoint sets, $V = X \cup Y, X \cap Y = \phi$, such that all edges are between a node in *X* and a node in *Y*. For the agent-based carpooling application, this may correspond to the case in which the agent population is composed of drivers and non-drivers [1]. For a schematic example, see Fig. 1 (right).

2.2 Complexity of the maximum matching problems

Following the early famous work by Edmonds [6], various algorithm have been developed to solve the maximum matching problem. There are solutions which run in $O(|E|\sqrt{|V|})$ time. More recently, methods based on *relaxation* have been introduced, which allow to apply a rich selection of optimization techniques to the matching problem. While space does not permit a comprehensive survey of these methods, we briefly demonstrate how they are applied for the case of a bipartite graph. Assume we have four drivers(agents) and four non-drivers, and there is a compatibility measure c_{ij} for every driver i and non-driver j. In order to find an optimal matching, we can look at the following optimization problem: define binary decision variables x_{ij}, which can attain values of 0 or 1, and maximize the expression $\sum_{i,j} c_{ij}x_{ij}$, subject to the constraints $\forall i,j : \sum_i x_{ij}, \sum_j x_{ij} \le 1$. Note that these constraints guarantee that each driver/non-driver will be matched with only one non-driver/driver. Generally speaking, the main difficulty in solving such optimization problems is the binary constraint; this can be *relaxed* to solving exactly the same problem but with x_{ij} allowed to attain every value between 0 and 1. After solving this problem, driver i can be matched with non-driver j for which x_{ij} is maximal.

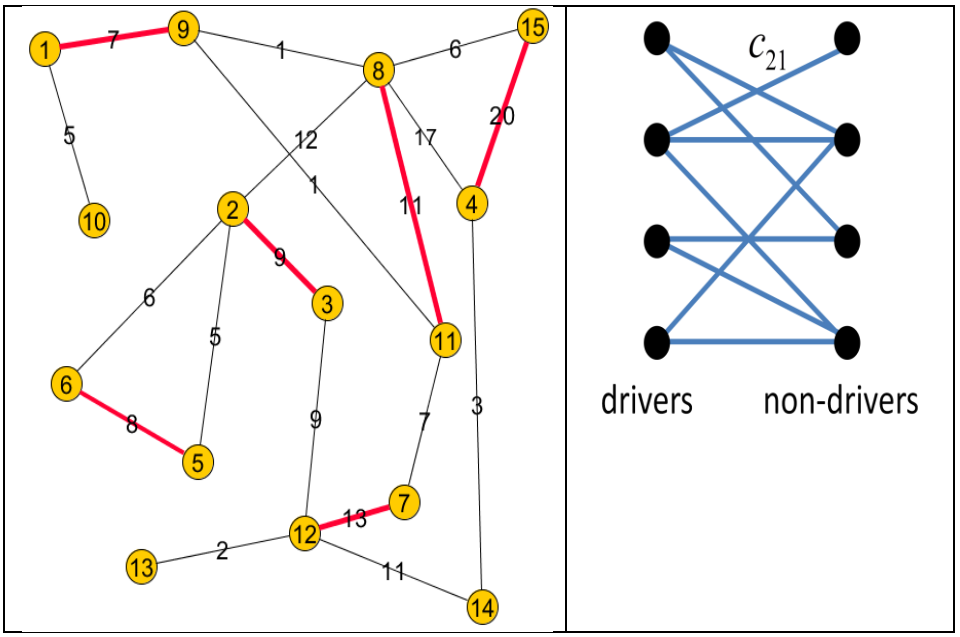


Fig. 1. Left: a maximal matching result for a graph with 15 nodes (or agents). Edge weights are depicted. The edges corresponding to a maximum matching are colored in red. Right: schematic example for the bipartite graph corresponding to drivers/non-drivers matching.

2.3 Incremental and online solutions

As discussed in the introduction, the input to the agent-based carpooling application is highly dynamic. This necessitates developing, in addition to the well-studied batch solutions, algorithms which

are *incremental* or *online.* An incremental algorithm assumes that the optimal solution of the carpooling for some input was computed already, and then it attempts to solve (either accurately or approximately) the problem for the same input but under a small perturbation. Online algorithms assume that the entire data arrives in an online fashion, e.g. each agent uploads his/her preferences, and the algorithm has to immediately assign the agent a carpooling partner. Typically, online algorithm achieve results which are inferior to incremental, but it enjoys a very fast "response time". Existing pairs show a reluctance against being broken that grows with the inverse of the time remaining before the start of the trip (since reorganizing human activities comes not for free).

3. Scalability and Graph Partitioning

A major difficulty with running both simulations and real life applications on *large agent-based* tasks is the size of the problem. For the agent-based carpooling application, for example, the computational complexity increases in a super-linear fashion, making it difficult to compute an optimal solution in the presence of millions of agents. The same holds for other problems in large-scale transportation scenarios; hence, the necessity of parallelizing these problems.

The crucial issue in the *parallelization* of a problem is the ability to decompose it into disjoint sub-problems, whose solution can be run in parallel. For transportation problem, "disjoint" means that the set of localities (and/or agents) can be partitioned into a disjoint union of sets which cannot interact with each other; for example, if no travel is allowed between different nations, the problem can be solved for each nation separately. Alas, this is not the case. In the terminology of graph theory, the graph representing e.g. the agent-based carpooling application is *connected*, that is, for every two nodes (or agents) there is a path connecting them. The most direct solution to this problem is to try and partition the graph to sets which are: (i) *nearly disconnected*, and (ii) *of roughly equal size*. Property (i) guarantees that solving each sub-problem separately is a good approximation to the global solution, and property (ii) guarantees that the sizes of the sub-problems are roughly identical, which is necessary for an efficient parallel solution.

3.1 Graph partitioning – Problem definition and solution

We now formalize the partitioning problem, following the considerations discussed above. A *cut* in a graph (*V,E*) is defined as a partition of the node set *V* into two disjoint sets A & B. The *cut value* is defined as $Cut(A,B) = \sum_{i\in A, j\in B} e(i,j)$, where $e(i,j)$ is the weight of the edge between nodes *i,j* (assumed to be 0 if no edge exists). Clearly, the smaller the cut value, the better is the partitioning for our purpose, so the goal is to find a cut with minimal cut value (*min cut*). However, mincut often yields sets very unbalanced in size, one of which may consist of a single node. To overcome this, [7] introduced a measure for the cut which also attempts to balance the sizes of *A* and *B*, referred to as *normalized cut* (Ncut), defined by

$$Ncut(A,B) = -\left[\frac{\sum_{i\in A, j\in B} e(i,j)x_i x_j}{\sum_{i\in A} d_i} + \frac{\sum_{i\in A, j\in B} e(i,j)x_i x_j}{\sum_{i\in B} d_i} \right] \qquad x_i = 1(i \in A),\ x_i = -1(i \in B) \ ,$$

where d_i is the *degree* of node *i* (defined as the sum of edges which are incident on it). The main difference is that here, the size of the cut is normalized by the sum of the degrees of the nodes in the two sets *A* and *B*, thus biasing the partition towards larger sets. Minimizing *Ncut* is known to be NP-Hard [7], however a good approximation can be found by reducing it to a generalized eigenvalue problem [7] with

a matrix whose elements are determined by the weights (in our case, compatibility measures) $e(i, j)$; space does not allow us to cover the details. Here, too, the problem is relaxed in order to obtain an efficient solution, hence the resulting x_i will be real numbers (instead of 1 or -1); in order to define the partition *A,B* a threshold *T* should be chosen, and , *A(B)* defined to be the set of indices *i* such that x_i is larger(smaller) than *T*. A schematic example of the resulting partition is provided in Fig. 2.

3.2 Incremental algorithm

The partitioning problem in the agent-based carpooling application is particularly amenable to incremental analysis, since typically the overall change in the compatibility measures over time will be slow. Since the solution is based on computing an eigenvalue problem, efficient algorithms from the realm of *perturbation theory* can be applied [8].

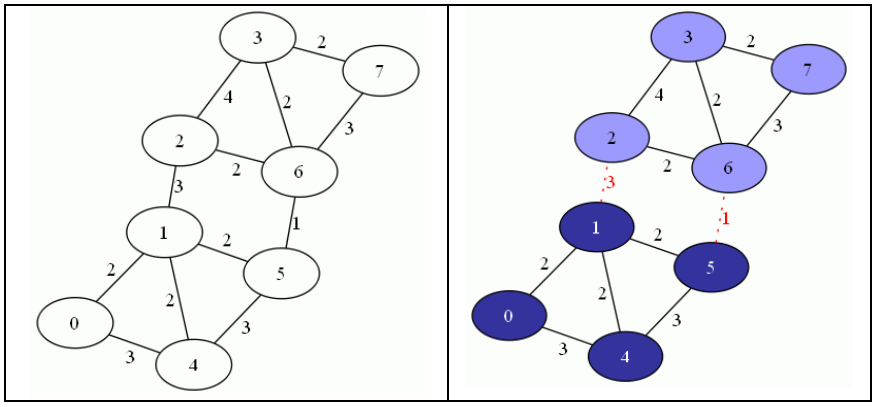


Fig. 2. a graph (left) and a minimal balanced cut (right).

4. Previous Work

In [1], bipartite matching was applied in a simulation to traffic data collected in the Atlanta area, yielding a substantial improvement over a greedy approach. A similar idea was introduced in [2]. Some papers discuss the application of the related *assignment problem* to carpooling [3]. For a comprehensive survey, see [4]. A theoretical study of a general assignment problem is offered in [5].

5. Conclusion and Future Work

In a nutshell in this paper, we explore the use of the graph-theoretic tools for matching and portioning problems of agents in an agent-based model for the carpooling application that falls under the domain of transportation and traffic modeling.

In this paper, as a proof of concept, we have first performed an initial experiment with a maximal matching result for a graph with 15 agents (as illustrated in Fig. 1). We have as a second step performed an experiment related to the partitioning problem with a limited number of agents based on computing an eigenvalue (as illustrated in Fig. 2).

The outcome of both of our experiments show promising results to solve the matching and partitioning of agents in an agent-based carpooling application. Moreover, the application of our proposed solutions is

not limited to the agent-based carpooling application only but can also be equally applicable to other similar agent-based traffic and transportation models, methodologies and applications.

As a part of the future work, we intend to develop a prototype for the agent-based carpooling application based on the concepts and solutions presented in this position paper with a large number of agent population data. Furthermore, we intend to apply our solutions to other similar agent-based traffic and transportation models, methodologies and applications.

Acknowledgements

The research leading tot these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 270833.

References

[1] Niels Agatz, Alan Erera, Martin Savelsbergh and Xing Wang (2010). The Value of Optimization in Dynamic Ride-Sharing: a Simulation Study in Metro Atlanta. ERIM REPORT SERIES RESEARCH IN MANAGEMENT ERS-2010-034-LIS.

[2] Gyozo Gidofalvi, Gergely Herenyi, and Torben Bach Pedersen (2008).Instant Social Ride-Sharing. Proc. 15th World Congress on Intelligent Transport Systems, p 8, Intelligent Transportation Society of America.

[3] Roberto Baldacci, Vittorio Maniezzo and Aristide Mingozzi (2004). An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation. Operations Research, Volume 52 Issue 3, June 2004.

[4] Sophie N. Parragh, Karl F. Doerner and Richard F. Hartl. A survey on pickup and delivery problems: Part I: Transportation between customers and depot, Part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58 (1, April), 21-51 and 58 (2, June), 81–117.

[5] James Zou, Sujit Gujar and David Parkes (2010). Tolerable Manipulability in Dynamic Assignment without Money . 24th AAAI Conference on Artificial Intelligence (AAAI '10), 2010.

[6] Edmonds and Jack (1965). Paths, trees, and flowers. Canad. J. Math. 17: 449–467.

[7] Jianbo Shi and Jitendra Malik (2000). IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888 – 905.

[8] Avraham Levy and Michael Lindenbaum (2000). Sequential Karhunen–Loeve Basis Extraction and its Application to Images. IEEE Transactions on Image Processing, vol. 9(8), 2000, 1371 – 1374.

[9] Muaz Niazi and Amir Hussain. (2011). Agent-based Computing from Multi-agent Systems to Agent-Based Models: A Visual Survey, Springer Scientometrics: 89(2), pp. 479-499.