Application of multi-agent systems in the case of vehicle routing.

Łukasz Chomątek, Technical University of Łódź

Abstract

Nowadays, the problem of traffic congestion is very serious in many countries. Roads that were built earlier usually don’t have sufficient capacity to handle increasing number of cars. This causes inconveniences such as wasting of time by both private and commercial drivers, greater usage of fuel and many others. What is more, drivers usually choose the same way, every time they drive to the certain place, which causes blocks on the major roads.

On the other hand, we observe the increase of the popularity of GPS devices that drivers can install in their cars. Furthermore many online services are offered, where drivers have an opportunity to plan their trip with certain criteria.

To handle many queries from the Users, efficient algorithm is needed to give them responses as fast as possible. One have to consider two cases in this problem: system can be dedicated either for single User, like in a personal GPS device, or for multiple Users, like in online trip planning services.

In this paper we discuss the possibility of application of multi-agent system to handle Single Source Shortest Path (SSSP) problem. Underlying algorithm is an extension of known Highway Hierarchies algorithm.

1. Introduction

Increasing number of vehicles on the roads caused the increase of the popularity of the GPS devices that can be installed in the car and the Internet sites, when one can plan the trip. As Users usually have some preferences about the route, their needs can be divided into three groups:

- long-distance trip planning – User knows both the source and the destination address. Path is computed automatically by the system, so in this case, User doesn’t have to worry about the efficient way to find the destination place.
- finding some information – system can show the way to the nearest theater, restaurant or gas station.
- information about situation on the road – User can be warned about traffic jams, speed cameras, road works and accidents.

In all of this situations, efficient routing algorithm is needed. The problem of routing is discussed in many articles and one of the most successful attempts is the hierarchical division of the road network. Hierarchical algorithms usually consist of two phases – at first, one has to prepare the hierarchical graph, then the actual queries can be send to the system to find the optimal route.

In this article, we review known algorithms designed for solving SSSP problem, mainly focusing on hierarchical algorithms. Hereafter, we consider a multiagent system dedicated for hierarchical routing algorithm for SSSP problem. After that, we present obtained results. To the end, we show conclusions and proposal of the future work.

2. SSSP algorithms

Algorithms for finding optimal route between two points in the graph can be divided into two groups: one, that operate on the graph directly, and these, that need some preprocessing of the graph. One of the most known algorithms that does not need preprocessing are Dijkstra's algorithm and A*.

In the following subsection, we will focus on algorithms that need some preprocessing. However the preprocessing phase can take a long time, queries are performed faster than in algorithms that operates directly on the road network graph.

2.1 Hierarchical algorithms

One of the earliest hierarchical algorithms was the Hierarchical Path Views proposed in the literature [1, 2], which was based on the following ideas:

- base road network was split into some fragments - places of split were chosen by its geographical coordinates,
- connections which are outside generated fragments belong to higher hierarchy level,
- division and level transfer is an iterative process.
The result of such a division are the matrices containing the shortest path lengths for each segment and each level. After the division phase, to perform a query, A* algorithm was used.

For other branch of hierarchial algorithms for solving SSSP problem was Highway Hierarchies algorithm proposed in [3, 4]. Dijkstra’s algorithm is used in the preprocessing phase to calculate the neighborhood for each vertex. Next, the vertices that fulfill some criteria are moved to the higher hierarchy level. When this phase is done, the higher hierarchy level is preprocessed that allows to generate shortcuts between certain vertices. Number of hierarchy levels and size of the neighborhood are parameters of the algorithm. Proper choose of them influences on the amount of time needed to process a single query.

2.2 Highway Hierarchies

Highway Hierarchies algorithm requires two parameters: \( H \) - identifying the degree to which requests for the shortest way are met without coming to a higher level in the hierarchy and \( L_\alpha \), which represents the maximum permissible hierarchy level. The method used to iteratively generate a higher level with number \( l+1 \) for graph \( G_l \) is as follows:

1. For each vertex \( v \in V \) build the neighborhood \( N^l_H \) for all vertices reached from \( v \) by using Dijkstra’s algorithm in graph \( G_l \), respecting the \( H \) constraint. Set the state of the vertex \( v \) to “active”.

2. For each vertex \( v \in V \):
   a. Build the partial tree \( B(v) \), and assign to each vertex its state. The state of the vertex is inherited from the parent every time the vertex is reached or settled. Vertex becomes “passive” if on the shortest path \( \{v, u, \ldots, w\} \), where \( v \neq u \neq w \):
      \[
      |N^l_H(u) \cap N^l_H(w)| \leq 1
      \]  
      Partial tree is completed, when reached but not settled vertices don’t exist.
   b. For each vertex \( t \), which is a leaf node in the tree \( B(v) \) move each edge \( (u, v) \), where \( u \in N^l_H(t), w \in N^l_H(v) \) to the higher hierarchy level.

During the first stage, a highway hierarchy is constructed, where each hierarchy level \( G_l \), for \( l < L_\alpha \) is a modified subgraph of the previous level graph \( G_{l-1} \) such that no canonical shortest path in \( G_{l-1} \) lies entirely outside the current level for all sufficiently distant path endpoints: this ensures that all queries between far endpoints on level \( l-1 \) are mostly carried out on level \( l \), which is smaller, thus speeding up the search.

2.3 Possible extension

Some parts of construction phase of Highway Hierarchies algorithm can be performed concurrently:

- Weight assignment for each road segment, in general using different rules
- Construction of \( N^l_H \) neighborhoods for each vertex in graph
- Construction of \( B(v) \) trees

We decided to try performing division of road network graph, so that Highway Hierarchies algorithm can be performed on a single part of this graph. After completion of the algorithm on each part, all subgraphs should be merged to obtain a final Highway Hierarchies graph.

To perform a division of a graph, Breadth First Search (BFS) algorithm for certain vertices was applied as follows:

1. Get a list \( BFS_{\text{start}} \) of vertices mentioned to be start points for BFS
2. For each vertex \( v \in BFS_{\text{start}} \) create empty lists \( E_v \) and \( V_v \) to store information about edges and vertices that belong to the subgraph.
3. For each vertex \( v \in BFS_{\text{start}} \):
   a. For the vertices from BFS queue, check if their children are allocated in any subgraph. If not, add them to BFS queue for current vertex and to \( V_v \). Add corresponding edges to \( E_v \).
4. Check if all vertices of the base graph are not allocated in on of the subgraphs. If not, go to step 3.
5. Perform representation dependent postprocessing for each set \( V_v \) (i.e. reorder vertices if needed).

Such an algorithm can be applied to connected graph, if there are more than one connected components in the road network graph, described division can be performed for each connected component treated as a base graph.

Note that in presented algorithm some edges can be not included in any graph. We can denote them as \( E' \). After performing construction phase of Highway Hierarchies algorithm on each subgraph, these edges must be included in the result graph. It is a two step process:

1. For each vertex \( v \in BFS_{\text{start}} \), add all of the vertices from \( V_v \) and all edges from \( E_v \) to the final graph
2. For each edge \( e \in E' \) that connects vertices \( v \), and \( v' \), get the highest hierarchy level from all incoming edges of vertex \( v \).
3. Multi-agent system for vehicle routing

In [5] an application of multi-agent system for building Highway Hierarchies graph was proposed. Two main assumptions were made for this system:

System must be able to take into account user’s preferences (i.e. route should be the shortest, travelling time should be lowest) and environmental conditions (i.e. weather, time of day)

Computations should be done concurrently, where it is possible to be done.

To complete first of these assumptions, weights of the road segments must be assigned using different criteria, such as length, average travelling time, speed limits, etc. It was decided to introduce some number of reactive agents which collects data from different road segments.

This type of agent can work in two different ways, depending on the data structure which is used to store road network topology. The first way is associated with the nodes as it is easy to get information about edges connected to the node. Second way is related to edges. If list of edges in the graph is directly provided, it can be divided into some parts and each part can be analyzed by a single agent. However graph are usually represented in a hierarchical way, where nodes are on the top level and data for edges is usually kept as a list for each node, complete list of edges is helpful for weight assignment criteria based only on some properties of a single edge (i.e. length, speed limit). On the other hand, some important information can be kept in nodes – one can consider criterion of avoiding bigger crossroads so that all of the road segments connected to such node should have its weight properly adjusted.

In our system both graph nodes and edges are kept in the separate lists. However references are duplicated it simplifies the way of access to the needed data and allows both simple and complex weight assignment rules.

Regardless of the chosen solution, this process can be performed in parallel, what means sharing work for several agents. Depending on the selection criterion by which individual weights are calculated, work on each road section may perform one or more agents (each can calculate weight using different method). If the weight of the segment is calculated on the basis of several criteria, use of a coordinating agent for the weights assignment process can be considered. The coordinating agent can calculate weight in accordance with certain rules (e.g. use the weighted average of the values calculated by the agents). Coordinating Agent may have some adaptive abilities, depending on the application of the system [5, 6].

Concurrent computation can be also applied in the other parts of Highway Hierarchies graphs creation process. Obviously, calculation of neighborhood $N^i_H$ for each vertex is independent of each other. The only nuisance is that for each vertex, different queue of vertices intended to be visited must be kept. Any number of agents can be used to calculate such a neighborhood. Depending on the developer's choice, these agents cooperate directly with agents responsible for assigning weights to graph edges or with the coordinating agent.

Another process that can be done in parallel by agents for the individual vertices of the graph is the creation of trees $B(v)$. This process is to be implemented through the cooperation with agents that build neighborhoods.

Responses to user queries for the system should take into account his preferences regarding the itinerary and the current conditions on the road. To this end, it might be necessary to create several Highway Hierarchies graphs, which will be used to obtain a system response depending on certain factors. Different graphs can be prepared for example for the city centre during peak hours and at night. To implement this assumption, the introduction of a special type of agent can be considered. Such an agent will redirect the user query to the appropriate HH graph. Relay agent may assist in work of coordinating agent by suggesting the criteria by which the weight of the edge should be calculated [5, 6].

Fig. 1 Dependencies of agents in Highway Hierarchies algorithm [5].
Proposed architecture of multi-agent system described above is shown on

Fig. 1. Performed tests [5] revealed that for diverse criteria, calculated hierarchies differ very much. For three proposed criteria: speed limits, travelling time and road length, obtained results shown that these hierarchies graph at each level only few common edges with other hierarchies graphs. What is more, expected convergence between dominating user’s preference and number of common edges with the hierarchies graph for this criterion was observed.

3.1. Agent types

Application of multi-agent system described in the previous section, shown such an architecture can be successfully used for both handling user’s preferences and speeding up construction process of Highway Hierarchies graph. In order to apply improvements described in section 2.3, architecture of multiagent system must be significantly changed.

There are two new types of agents required to perform such a process:

Graph splitting agents, which are supposed to prepare proper split of the graph, and pass the information to corresponding neighborhood calculators. Graph splitting agents can be considered as social agents as they have to cooperate with other graph splitting agents while doing their work, because some edges can be allocated in different subgraphs.

Graph merging agent, which task is to merge subgraphs prepared by splitting agents according to certain rules. Although graph merging agent work looks complicated, it is reactive agent – it have to wait for the graph splitting agents to complete their work so that he can perform the merging process.

Introduction of new types of agents to the system implies changes in the data flows in the system. Preferences Handler agent still cooperates with Weight Assignment Coordinator agent in order to pass information about User’s needs. As weights are known, Graph Splitting agents can divide graph into subgraphs and then cooperate with Neighborhood Calculator and Tree Bulider agents to prepare partial Highway Hierarchies graphs. After that, Graph Merging agent can prepare final division of the road network into hierarchy levels. Query processor agent cooperates directly only with Graph Merging agent. Dependencies between other types of agents remain unchanged.

4. Obtained results

Algorithm described in section 2.3 was implemented using C# 4.0 language in Windows environment. Tests were run on different maps both for single and split road network graph. First map was a real road network in the neighborhood of Technical University of Łódź and the second one was an artificial mesh. Exemplary result of full graph division is shown on Fig. 2. The thicker the line is, the higher hierarchy level it belongs to. On Fig. 3 results of algorithm performed on a subgraph is presented. However, in general, road segments on the top hierarchy levels in the full graph are on high hierarchy level in a part of the graph too, overall number on the highest hierarchy level is smaller for the subgraphs. It is caused by the fact that when the number of edges is smaller, promotion to the higher hierarchy level is harder.

Fig. 2 Example of Hierarchical Division for neighborhood of Technical University of Łódź – HH(3,5)
Research made for both maps revealed, that amount of time needed to perform calculations, depends hardly any on number of edges in the subgraph. In case of real road map, after a division both subgraphs contained exact number of the vertices, however first part contained significantly more edges. Time needed to compute hierarchy levels for both parts were almost identical. The second graph was an artificial road mesh, where after a division, in both parts number of edges was the same. Performed tests shown that in this case, time of computing for both subgraphs was the same too.

For the whole artificial and real road graph Highway Hierarchies algorithm was run using such parameters:

a. Maximum hierarchy level: 3
b. Dijkstra’s Neighborhood size: 5

In the next step, Highway Hierarchies was run for the subgraphs with identical parameters. Obtained results for subgraphs show, that when a division is made, number of edges on each level differs from number of these edges when whole graph is taken into account in Highway Hierarchies. For real road network, such a difference was up to 70%. It was caused by the fact, that large number of edges was included in the first subgraph. For the artificial road network, this difference was smaller than in the real network.

As in smaller graphs, it is harder to reach higher hierarchy level (larger graphs usually contain longer paths), one decided to lower the size of Dijkstra’s neighborhood used in the division into the hierarchies. This was supposed to facilitate promotion to higher hierarchy levels (step 2b of Highway Hierarchies algorithm). In case of real road network, number of edges on each hierarchy level was closer to one in the reference division (about 37% better result).

In case of the artificial road graph, the difference in number of edges on each level, between result obtained for Highway Hierarchies computed for whole graph and for two subgraphs, was about 7%.

Generally, lowering size of Dijkstra’s neighborhood resulted in lower differences between reference hierarchical division and a division made for subgraphs.

The second test was performed to check, whether HH algorithm for splitted graph is faster than the algorithm ran for the whole graph. Results are gathered in the table below. Values in the table are given in percents, which are the amount of time needed to perform each phase of the algorithm in addition to time needed to construct Highway Hierarchies for the whole graph. As one can see, when algorithm is supposed to build hierarchies with greater number of levels, the gain is the highest. When number of maximum level is set to 1, the gain is not so high. Split and merge phases have different percent shares in the result time, as the time needed to perform them is constant for a graph.

<table>
<thead>
<tr>
<th>HH parameters</th>
<th>First subgraph</th>
<th>Second subgraph</th>
<th>Merge</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical University of Łódź</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3,5)</td>
<td>10.7 %</td>
<td>11.4%</td>
<td>3.5%</td>
<td>2.1%</td>
</tr>
<tr>
<td>(3,3)</td>
<td>7.3%</td>
<td>7.5%</td>
<td>5.0%</td>
<td>3.0%</td>
</tr>
<tr>
<td>(1,3)</td>
<td>11.8%</td>
<td>11.9%</td>
<td>25.4%</td>
<td>15.2%</td>
</tr>
<tr>
<td>Artificial mesh</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3,3)</td>
<td>18.0%</td>
<td>17.6%</td>
<td>18.5%</td>
<td>18.9%</td>
</tr>
<tr>
<td>(3,5)</td>
<td>19.6%</td>
<td>19.4%</td>
<td>9.5%</td>
<td>9.7%</td>
</tr>
</tbody>
</table>

Table 1. Time amount needed to complete each phase of the algorithm for split road graph in addition to time needed for HH construction for whole graph.
5. Conclusions and future work

Performing hierarchical division of road network on the split of the road graph can improve the construction phase processing time due to the lower computational complexity.

Number of edges on certain levels in subgraphs can differ very much from these from division of whole graph. Adjusting hierarchical algorithm’s parameters can improve results of divisions of subgraphs.

Multi-agent system can be utilized to solve this problem, what allows to compute most parts of the algorithm in parallel. Architecture of presented multi-agent system is extensible. There is a possibility to implement new types of agents for different graph split methods.

Bibliography (Style Chapter)


Authors:

MSc. Łukasz Chomątek
Politechnika Łódzka,
Instytut Informatyki
ul. Wólczańska 225
90-924 Łódź
tel. 509-102-430
email: lukasz.chomatek@ics.p.lodz.pl