ROUTE FINDING FOR FACILITATING TRANSPORTATION PLANNING TO MAINTAIN SMOOTH PRODUCT FLOW IN SUPPLY CHAIN

Muhammad Ali Memon
University of Sindh, Pakistan
Asadullah Shaikh
Ilma University, Pakistani
Kamran Taj Pathan
University of Sindh, Pakistan
Majid Hussain Memon
Quaid-e-Awam University, Nawabshah
Kamran Dahri
University of Sindh, Pakistan

ABSTRACT

Purpose: To gain more profit and market share, manufacturers wish to mobilize as many resources as they can to satisfy every manufacturing order. As Manufacturing is nowadays distributed to several sites, where each site manufacture the intermediate product to assemble the final product.
Methodology: Producers, therefore require to transport these products between these sites as well as distribute the final products to faraway customers. Therefore production and transportation planning need to be coordinated. Manufacturing requires the planning of unmovable resources (machines) with fixed sequences of production routing steps, however transportation requires the planning of moveable resources (Vehicles).
Findings: Each transport request is different from another with different origin and destination. Hence, it is needed to find the routing (route from origin to destination) of each transport request dynamically.
Practical Implications: This paper is dedicated to present a route finding tool called Path Finder in order to provide shortest route by distance or time or both for transportation planning.

Keywords: Multi Agent System, Collaborative networks, Transportation planning, Simulation.

Jel Classification: D1, D11, D 12

1	Muhammad Ali Memon:	muhammad.ali@usindh.edu.pk
2.	Asadullah Shaikh :	shaikhasad@hotmail.com
3.	Kamran Taj Pathan :	kamran.taj@usindh.edu.pk
4.	Majid Hussain Memon:	majidhussain@quest.edu.pk
5.	Kamran Dahri :	kamran.dahri@usindh.edu.pk

ILMA-JBS is published by the Ilma University – Formerly IBT
Main Ibrahim Hydri Road, Korangi Creek, Karachi-75190, Pakistan.

1. INTRODUCTION

Nowadays, manufactures have to deal with variety of competitors, suppliers and customers coming from every possible market in the world. With the help of computer-based information interchange, suppliers, manufacturers, distributors, and retailers can share their information in a much faster and more precise pattern (Li, Y. (2007)). To deal with the complexity and heavy weight business routines, people need some automatic mechanism to handle the data and information during the operation in logistics and manufacturing activities. Organizations globally since the 1990s are involved in the implementation and usage of ERP systems in order to have a identical information system in their respective organizations and to re-engineer their business processes (Rajagopal, P. (2002)).

Usually, enterprises equipped with these software or IT systems will benefit more or less from the best practice or improved work process provided. While some critical problems are still expected to be solved are complex production scheduling, supply chain management, finance and accounting, client relationship management, etc.

Logistics and manufacturing enterprises both have the requirement of dealing with transport scheduling problem. Many researches have proposed different active models to solve such distributed resource scheduling and planning problems. People proposed multi-agent system (MAS) which is a computerized system composed of multiple interacting intelligent agents within an environment (Van der Hoek, W., & Wooldridge, M. (2008)). Multi-agent system is the subfield of Artificial Intelligence (AI) that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors Stone, P., & Veloso, M. (2000). In other words, Multi-agent systems are designed to be competent to solve complex problems that are difficult or impossible for an individual agent or a monolithic system to solve by co-operation between several autonomous agents via collaborative networks (Rocha, A. D., Barroca, P., Dal Maso, G., & Oliveira, J. B. (2017).

Driven by the requirement of dealing with distributed resources and their scheduling, different multi-agent models have been proposed and applied trying to solve such scheduling tasks. In the context of transportation, transportation requests arrive with their parameters only pickup location and delivery location. The first and important issue is to find shortest path (routing) path from origin and destination in order. A system's up-to-date awareness of what transportation resources it owns and what a reasonable path (if not optimal) it should follow to fulfill certain transportation tasks is primary concern for the transportation scheduling systems. In this paper, we propose a Path Finder tool as a prototype web service who provide vehicles transport routing in order to achieve the transportation scheduling in a multi-agent system. In this paper, second section is dedicated to state of the art of transportation scheduling techniques followed by Path select algorithm. In third section, we present Path finder and its architecture. In fourth section, we illustrate the implementation of Path Finder and results conducted with benchmarks.

2. STATE OF THE ART

2.1 Transportation scheduling techniques and methodologies

As we know that, any kind of scheduling is actually all about allocating resources to satisfy different requirements, or mapping a set of tasks and jobsto a set of target resources efficiently under some objective constraints. Some scheduling methodologies have been successfully applied into production. We assume that, transportation problems have the similar properties with production planning problems, except the manufacturing has fixed machines and might be fixed routing, on the contrary transportation vehicles are moving and dynamic routing.With such abstraction and mapping, we can easily apply some production scheduling methodologies or techniques to transport scheduling.

Two famous domains for transportation planning solutions are operational research approaches and mathematicalprogramming. In operational research approaches, a complete structure of the shop, activities, jobs, and all other related constraints can be represented in some detail as that, given appropriate input data and simple heuristic dispatching rules at decision points, the computer could extrapolate a given schedule into the future at a relatively low cost. Investigators should generate various types of artificial or historical input data and simulate the effect of using different types of simple heuristics under different conditions. In mathematical programming approach, linear integer programming and dynamic programming are the most widely mathematical techniques used for solving production-scheduling problems. The main advantage of such mathematical models is their capabilities of determining the optimal schedule. For large-scale problems, the mathematical formulation remains difficult and in some cases it cannot be derived (Elwany, H., Shouman, M., & Abou-Ali, 2001).

The traditional and oldy existing scheduling approached face great difficulties when applied to real-world situations due to the usage of simplified theoretical models and carrying out computations in a centralized fashion. The multi-agent technologies, on the contrary, suggest an advanced, and more robust approach to scheduling problems [14].

A definition from Advanced Agent-Robotics Technology Lab of Carnegie Mellon University describes multi-agent system like this: A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver [15]. Agent-based systems technology has been hailed as a new paradigm for conceptualizing, designing, and implementing software systems. This essentially distributed approach is more flexible, efficient, and adaptable to real-world dynamic manufacturing environments.

Recently, researchers have applied multi-agent methods trying to solve scheduling problems. Applications include manufacturing shop scheduling, transportationscheduling, construction scheduling(Horenburg, T., Wimmer, J., & Günthner, W. A. (2012)), power distribution scheduling, Grid-scheduling Solar, M., Rojas, J., Mendoza, M., & Monge, R. (2012), meeting and course timetable scheduling(Oprea, M. (2007)), medicaltest scheduling, and project management.

Agent-based approaches have several potential advantagesfor distributed manufacturing scheduling (Shen, W. (2002)). These approaches use parallelcomputation in a distributed fashion by using many processors, which will enable scheduling systems to provide more results with high efficiency.

2.2 Path Select algorithms

The paper written by Leonhard Euler on the Seven Bridges of Königsberg and published in 1736 is regarded as the first paper in the history of graph theory (Biggs, N., Lloyd, E. K., & Wilson, R. J. (1976)). Since then quite much output of graph theory research had been applied in various fields. With the help of its considerable computation ability, graph theory and its algorithms have been successfully applied to physical, biological, social and information systems. There are always some classic and common scenarios we run into when dealing with real practical problems. For example, graph search algorithms, max flow algorithms, minimum spanning tree algorithms.

In this paper, our concern is to facilitate the scheduling of transportation resources by determining the shortest path. A classic data model for transportation problem is the directed graph, or directed net, which is constructed by combining the arcs, where each arc represent the non-stop travel trajectory of a transport resource. In order to find the route, typically, the single-source shortest-path problem was discovered by Dijkstra (1959), while the all-pair shortest-path algorithm is due to Floyd (1962), who obtained the result based on a theorem by Warshall (1962) (Sokas, A. (2011)). When dealing with a best path or shortest path problem, Dijkstra. and classic A* algorithms are fast reliable options.

A* algorithm is an improved version of Dijkstra method. The algorithm works on two set of nodes OPEN and CLOSED. The OPEN set comprises of candidate nodes that need to be examined, while CLOSED set contains those sets of nodes that have been already examined. In the beginning of execution, OPEN set comprises of only one element, which is the starting position and CLOSED set is completely empty. OPEN set is also called the “frontier” and CLOSED set is called “interior” of the areas visited. Each node keeps a pointer to its parent node so that in can be determined that from where it was found.

3. PATH FINDER AND TRANSPORTATION SCHEDULING SYSTEM OVERVIEW

Figure.1, presents a scenario, where we have several distributed manufacturing sites with their manufacturing planning systems that in collaboration forms as a virtual enterprise. In order to manufacture products, these sites need to interchange the intermediate products to form final products. They need to plan that manufacturing in a distributed fashion, for which we consider a generic multi agent model SCEP for distributed planning (Memon, M. A., & Archimede, B. (2013, April)).

The SCEP multi-agent model which introduces an indirect cooperation between manufacturing order agents representing the set of customers called C and machines agents representing the set of producers called P. This cooperation is performed synchronically via the blackboard environment E and is controlled by the supervisor agent S. The scheduling is achieved after a defined number of cycles. Each cycle corresponds to an activation of customer agents followed by an activation of the producer agents.[23] But the question is how we in case transfer goods among distributed sites with respect to various manufacturing processes and budget optimization. To handle such a more dynamic and collaborative scenario, the we evolved SCEP model and proposed a generic model for transportation planning called I-POVES (Memon, M. A., Letouzey, A., Karray, M. H., & Archimède, B. (2014).) as shown in Figure .2.

I-POVES model is inherited from the multi-agent model SCEP and each instance of I-POVES corresponds to particular transportation service provider such as a logistics company .I-POVES introduces an indirect cooperation between two communities of agents, the order agent community called (O) and the community of vehicle agents called(V). Each or deragent manages one transportation requirement. Each vehicle agent manages a vehicle of the organization. The supporting agent Path Finder (P) develops for each transportation order a route or path between places of loading and delivery. Cooperation between the order agents and the vehicle agents is performed synchronously in the environment E.Functioning of the model is controlled by the agent supervisors.

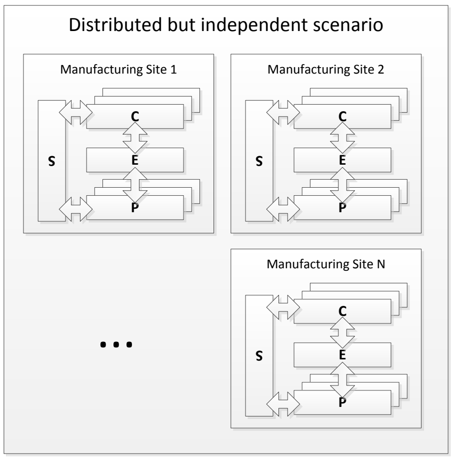


Figure.1: Distributed manufacturing sites with their planning systems.

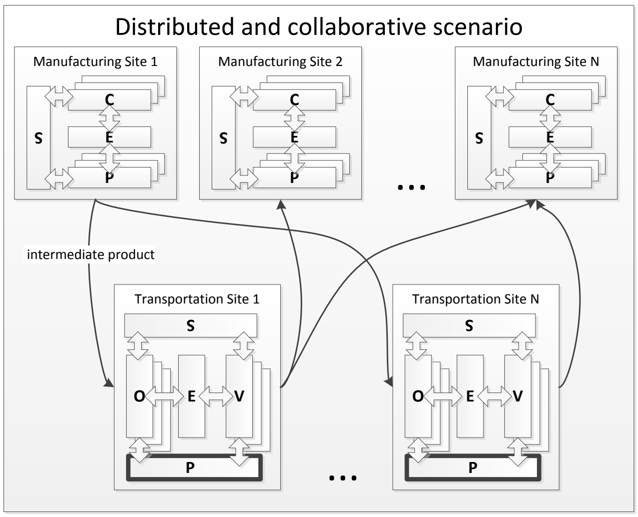


Figure.2**:** Distributed manufacturing sites with transportation incorporated

To fulfill the collaborative scheduling that covers a bunch of sites over a region, we might as well set Path Finder component as a public access for which a web service might be more competent. In our design, Path Finder will use real geographic data from in order to estimate the time and distance consumed by certain vehicles. Path Finder will not query these data in real-time, instead, it will do the updating query in every certain time duration, maybe every week or every month. Then Path Finder caches these data in local database. Path Finder is able to update the data manually by the site manager's command or it had better have the automatic update capability.

Path Finder have different search options, for example, the best path may be considered to be the shortest in distance, or least in time. Path Finder calculates the best path solution based on the average transportation time between two transport activities and the worst time. The calculation strategy can be altered manually by user.

For the data exchange, between different agents and Path Finder, is enables with both human-readable and machine-readable format. The communication message will also be recorded in Path Finder’s log.

Path Finder will have possibility for vague search, if Path Finder can’t find a user input for location’s name in our traffic net, it will query the geographical application (for example google maps) in real time for the place and identify the nearest location existing in our system, and then use this nearest location for the recommended best path search. For example, if the user input a pick up location’s name as “Lourdes”, which matches none of the locations registered in Path Finder. At this time, Path Finder will immediately send a query to geographical application to get the geographic information of Lourdes, and then search the transport net locally for a nearest location. Path Finder finds that the nearest registered location to Lourdes is Tarbes, thus Path Finder will alter the query using Tarbes as the new pick up location. The same mechanism will be applied to delivery location.

Figure .3 illustrate the Client-Server architecture of Path Finder agent. The client side containing either manufacturing or transportation systems are physically distributed, while the multi-agent scheduling computation is managed in a more centralized way. Figure. 4 shows that the same time each transportation scheduling request are sent to its exclusive server (T), Path Finder service returns a reasonable transportation routing which the scheduling must follow after.

We also notice that, there is a component called Shared Resources Register in Figure .4. This component has more concerns with data as it shall be responsible for any notification of transport resources alteration so that the Path Finder would not be blind. For example, when transportation site 1 wants to add its new purchased truck ‘comet’ as a new transportation resources in the system. It should notify Shared Resources Register the capability of the very truck and it competence i.e. how much the capacity of the truck is, from which city to which city it circulates, what kind of timetable it follows each day, and other geographic information like that. There are different dynamics when dealing with different order types. Routing for manufacturing site means the proceeding sequence shifting between different machines or workshops. The corresponding thing for transportation is the sequence of cities and routes. Any transport requirement from any manufacturing site could be equally a transportation order for a relative transportation site.

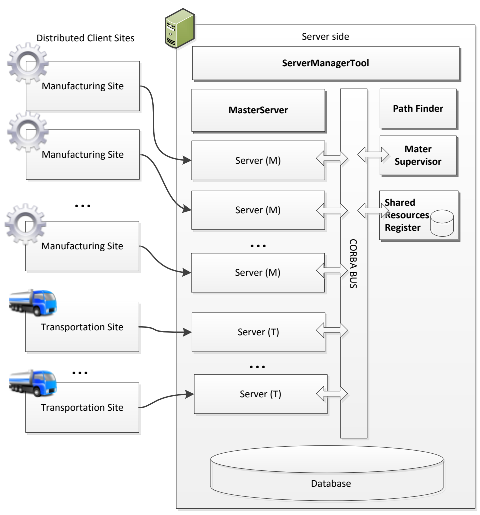


Figure.3: Path Finder architecture

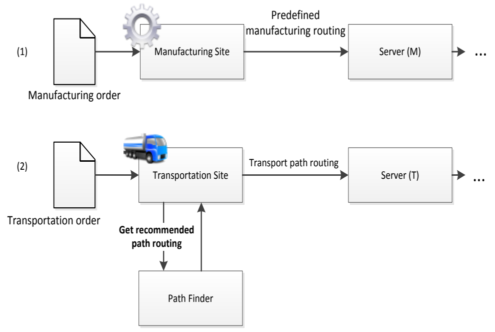


Figure.4

4. IMPLEMENTATION AND RESULT

As a service artifact, Path Finder follows the basic design pattern of request and response between client and server. Figure.4, is a trimmed and simplified structure of Path Finder. Basically, there would be 3 kinds of actions or behavior defined for Path Finder, maintenance operations or status query, path query and update geographic data query, as what the three Path Finder connection threads represent in the Figure. 4. Path Finder connection threads are the temporary sub-server for each request. Path Finder builds a connection via standard socket API with the client application, when the request is finished, then the thread is ceased and the connection will also be shut down. Every request will follow certain XML format protocolthat would be recognized by Path Finder. Path Finder connection thread will parse the message and execute respectively.

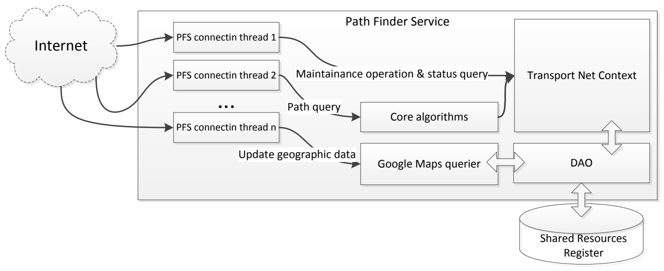


Figure.5: How Path Finder service deal with various request from client sites.

4.1 Transport Net Context

This graph data structure short for TNC is the core data structure of Path Finder service. The presence of TNC will be through the whole lifecycle of Path Finder service. The first thing Path Finder has to do after starting is to initializing TNC. Basically, TNC's design follows a hierarchy pattern. Figure.4 is the internal structure of a class TransportNetContext instance. We use an adjacency table to build the directed graph representing the transport net (g). Then the detailed information of vehicles and transport activities are stored in some sequential vectors. We use pointers in to weave them together.

Notice the member of class Activity (non-stop transport activity between two cities but the detailed geographical information is stored in class GoogleResult) that it contains a pointer that points to the list containing all the Activity's detailed information and another set of pointers that lead to all the related vehicles that are responsible for this Activity. The advantage of using pointer is that good flexibility of TNC is conserved. When we alter the structure during running time,we don't have to re-initializing TNC. Instead, we redirect the pointers. Transport Net and its entity members indicate the nodes and edges for a directed graph which represent the distributed sites and the transport activities. While for the detailed content of these participants, TNC uses pointers pointing to other data containers, the list vector vehicle _info containing all the vehicles' information and their relations with activities. Another list vector named google_result_info contains all the transportation activities' geographical information items that retrieved from Google Maps. Imagine this scenario, when we have to update a vehicle or a transport activity's details, we don't have to jump in the net and apply Deep First Search or Breadth First Search. With the independent data container, no matter if it will be arranged by a sequential or a hashed way; we just identify the target item in such container directly and do the wanted operation for good. In other words, this design will separate TNC's pure structure apart from its content. When the content needs updating, TNC's structure will be easily preserved.

There is a little difference between the relation Activity-Vehicle (shown in dashed line: ·················>) and relation Activity-GoogleResult (shown in dashed line: – – – – ➤). For example, activity from location 1 to location 3 will be executed by two vehicles: vehicle 2 and vehicle 3. One vehicle is ready to serve several activities is also possible, for example, in Figure.3, vehicle 1 is responsible for the two activities back and forth from location 1 to location 2. In other words the Activity-Vehicle relation is actually a multi-to-multi relationship. While for Activity-GoogleResult relation, one abstract transport activity is mapped to the only GoogleResult item. It's one-to-one relationship.

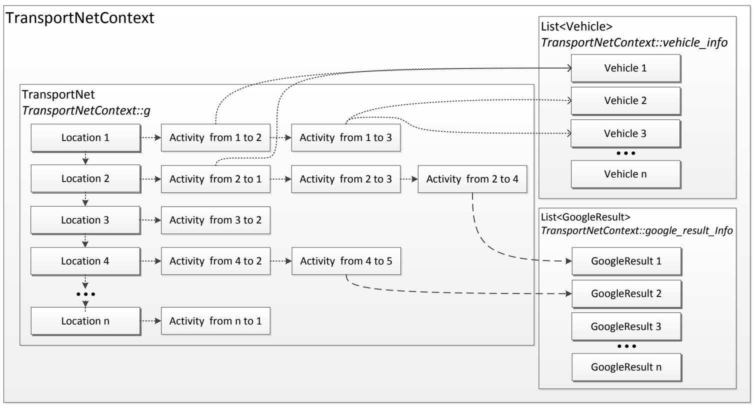
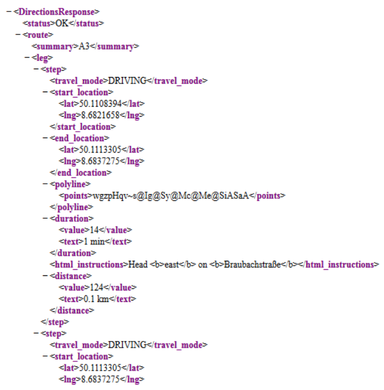


Figure.6: Transport Net Context internal data structure.

a. Google Map Querier

Google Map is a geographic information source and freely accessible over internet. Most of the information about the transport network arcs and real road links are obtained from Google Map service. According to the specification of Google Map, it is quite simple to do the geographic information query. Via HTTPprotocol, we just send a URL ending with necessary parameters to Google Map, and then a result of text format will be retrieved. Figure.5 is a text response in XML format. After receiving the XML or JSON response, Google Map querier will parse it, retrieve the useful geographical information and deposit them into the Shared Resources Register (Khambati, H., Boles, K., & Jetty, P. (2017)).



```
– <DirectionsResponse>
    <status>OK</status>
  – <route>
      <summary>A3</summary>
    – <leg>
      – <step>
          <travel_mode>DRIVING</travel_mode>
        – <start_location>
            <lat>50.1108394</lat>
            <lng>8.6821658</lng>
          </start_location>
        – <end_location>
            <lat>50.1113305</lat>
            <lng>8.6837275</lng>
          </end_location>
        – <polyline>
            <points>wgzpHqv~s@Ig@Sy@Mc@Me@SiASaA</points>
          </polyline>
        – <duration>
            <value>14</value>
            <text>1 min</text>
          </duration>
          <html_instructions>Head <b>east</b> on <b>Braubachstraße</b></html_instructions>
        – <distance>
            <value>124</value>
            <text>0.1 km</text>
          </distance>
        </step>
      – <step>
          <travel_mode>DRIVING</travel_mode>
        – <start_location>
            <lat>50.1113305</lat>
            <lng>8.6837275</lng>
```

Figure.7: XML response fragment (near header) from Google Maps

b. Algorithms performance analysis in Path Finder

By checking the log files, an analysis of performance was carried out to compare the performance of Dijkstra algorithm and A* when dealing with shortest distance path query is shown in Table 4.1 and Figure.5. We develop a case study containing about 60 principle metropolis in France and part of Germany, Spain and Italy. In last columns of the table 4.1, we can see that, when the query strategy is BEST_DISTANCE, A* algorithm is somewhat better than Dijkstra as predicted before. While on the other hand, Dijkstra seems to be more stable when the traffic graph is not too big. The other testing results show that Path Finder owns a quite acceptable performance outcome. No matter the BFS method applied to test if start and finish locations are reachable and the core algorithm, the response time didn't nearly exceed 0.01 seconds. However, further tests and improvement are still needed especially when the TNC grows to a bigger scale.

Table 4.1 Dijkstra and A* running results comparation in strategy BEST_DISTANCE

START	FINISH	TIME BFS(*ms*)	LOOPS IN BFS	BEST DISTANCE (*m*)	TIME Dijkstra (*ms*)	TIME A* (*ms*)
Vanes	Paris	<1	14	509,101	3	4
Troyes	Munchen	4	92	798,345	5	2
Nantes	Toulouse	<1	30	576,603	4	3
Lyon	Bologna	3	78	723,746	4	3
Padova	Bordeaux	1	32	1,299,230	4	2
Antwerpen	Limoges	2	65	900,119	4	4
Rouen	Numberg	4	110	1,035,621	6	4
Firenze	Berlin	12	289	2,410,700	6	7
Montpellier	Lille	2	56	1,008,777	6	2

Koln	Zurich	2	91	1,204,402	5	5
Tours	Frankfurt	3	94	1,018,876	5	3
Toulon	Caen	4	108	1,311,738	5	5

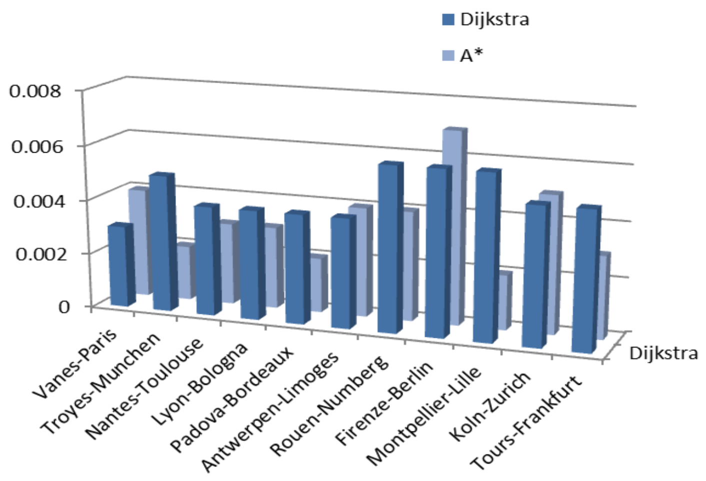


Figure.8: An easy performance comparison of Dijstra and A * under BEST_DISTANCE strategy.

5. CONCLUSION

To obtain a good global scheduling result in a multi-agent system, some consciousness of the situation as a whole is important. For transportation problem, an optimal transport route depends on a centralized calculation power. The more detailed and stricter the objective is, the more powerful the global control and calculating power should be. To support such a composed and complex calculating process to schedule distributed production resources, gathering and exchanging information in time appears extremely important. Path Finder is designed to be a service program that find the routing based on shortest path or time between origin and destination for a transport order in order to facilitate the transportation planning. It builds and maintains a transport net during its lifetime. The information Path Finder uses to build the transport net is retrieved from real time geographical applications. The most important function of Path Finder is dealing with the optimal path query. We can see that graph related algorithms are very useful to handle with transport problems. When we can find a reasonable heuristic function and the transport graph is thin, we might as well apply A* algorithm, which is a guided and more efficient method.

REFERENCES

Li, Y. (2007). Impact of modern logistics on industrial location choice and property markets (Doctoral dissertation, Massachusetts Institute of Technology).

Rajagopal, P. (2002). An innovation—diffusion view of implementation of enterprise resource planning (ERP) systems and development of a research model. Information & Management, 40(2), 87-114.

Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. Autonomous Robots, 8(3), 345-383.

Elwany, H., Shouman, M., & Abou-Ali, 2001 M. Production Scheduling Techniques–A Review. Deparment of Production Engineering, Alexandra University, Alexandra, Egypt.

Horenburg, T., Wimmer, J., & Günthner, W. A. (2012). Resource allocation in construction scheduling based on multi-agent negotiation. In Proceedings 14th International Conference on Computing in Civil and Building Engineering.

Solar, M., Rojas, J., Mendoza, M., & Monge, R. (2012). A multiagent-based approach to the grid-scheduling problem. CLEI Electronic Journal, 15(2), 2-2.

Oprea, M. (2007). MAS_UP-UCT: A multi-agent system for university course timetable scheduling. International Journal of Computers Communications & Control, 2(1), 94-102.

Shen, W. (2002). Distributed manufacturing scheduling using intelligent agents. IEEE intelligent systems, 17(1), 88-94.

Biggs, N., Lloyd, E. K., & Wilson, R. J. (1976). Graph Theory, 1736-1936. Oxford University Press.

Sokas, A. (2011). Algorithms and procedures of determining the shortest route in the graph. In The 7th International Conference TRANSBALTICA, Vilnius, Lithuania.

Memon, M. A., Letouzey, A., Karray, M. H., & Archimède, B. (2014). Collaborating Multiple 3PL Enterprises for Ontology-Based Interoperable Transportation Planning. In Enterprise Interoperability VI (pp. 319-329). Springer International Publishing.

Memon, M. A., & Archimede, B. (2013, April). Towards a distributed framework for transportation planning: A food supply chain case study. In Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on (pp. 603-608). IEEE.

Van der Hoek, W., & Wooldridge, M. (2008). Multi-agent systems. Foundations of Artificial Intelligence, 3, 887-928.

Rocha, A. D., Barroca, P., Dal Maso, G., & Oliveira, J. B. (2017). Environment to Simulate Distributed Agent Based Manufacturing Systems. In Service Orientation in Holonic and Multi-Agent Manufacturing (pp. 405-416). Springer, Cham.

Khambati, H., Boles, K., & Jetty, P. (2017). Google Maps offers a new way to evaluate claudication. Journal of vascular surgery, 65(5), 1467-1472.