Case study for distributional transport agent-based modeling and optimization

Patryk Płoski, Kacper Radzikowski and Paweł D. Domański

***Abstract*— The notions of agent-based modeling and optimization in the area of distributional supply chain management are well established. The literature delivers quite many potential solutions. Agent-based modeling and simulations are quite common, however they mostly incorporate deterministic approaches. Fully stochastic solutions are less frequent, while additional incorporation of optimization introduces another degree of complexity. This research covers two aspects of the distributional transport between distribution center (DC) and final store destinations: agent-based transport modeling and trucks routing. Both, the model and the optimization incorporates knowledge about road network, driving times, traffic jams and accidents. Custom congestion model is designed and used. Optimization solution compares various vehicle routing problem (VRP) and global optimization approaches to select the most appropriate solution.**

***Index Terms*— agent-based model; distributional transport; traffic congestion stochastic model; vehicle routing problem; global optimization**

I. Introduction

Agent-based models (ABM) form important modeling concept, which is now rather underestimated especially in relation to the abused black-box neural-based machine learning (ML) approaches. Both techniques, i.e. ABM and ML, have their inherent advantages and disadvantages and are characterized by their specific ranges of applications in which they are the most effective. ML models are ideal in modeling unknown processes using blind empirical data. ABM models, on the other hand, are ideal for reflecting complex processes consisting of many simple cooperating elements, such as road traffic, evacuation systems, smart cities, and human crowds behavior. They are ideal for modeling human-operated systems, like picker-to-parts warehouses [1], manufacturing plants [2]. Once we consider supply chain management systems, it would be good to have single-concept homogeneous framework, i.e. the entire system model should use the same modeling technique for all sub-processes.

Once the distribution centers (DC) or warehouses are represented in form of ABMs, then the distributional transport, so called the "last mile" network, should be modeled using multi-agent approach [3]. Initial reviews of ABMs utilization for traffic and transportation process started in early 2000s [4]. Computational abilities and new algorithm, introduction of agent modeling frameworks and graph-based technologies now allows to use efficiently ABMs in the distributional transport modeling [5]. Such models allow to incorporate traffic system specific features and its uncertainties. In the considered approach we use stochastic traffic congestion model that uses four parameter kappa (K4P) distribution [6]. Thus, the transport management system may incorporate traffic incidents in route planning.

Unfortunately, alone transportation model does not solve real life problems. However, only by using it in conjunction with a dedicated optimization solution will make it possible to solve real transportation management and planning problems.

Once we have an appropriate model, we may design a decision support solution that minimize according goods distribution travel time. This task is known under the name of the vehicle routing problem (VRP) [7]. Literature shows various approaches and optimization techniques that may address this challenge [8]. We propose to use hierarchical algorithm. This algorithm uses directed road network graph, which includes only endpoint nodes with defined connections and returns a list of trucks along with their sequences of endpoint visits. At the first level, the algorithm is used to distribute endpoints among trucks, while at the second level, it solves the traveling salesman problem (TSP). Additionally, to create a general graph from a detailed graph containing vertices that represent elements such as intersections, shortest-path algorithms are employed.The main paper's contribution is in the application of the homogeneous ABM approach to the problems of vehicle routing and its optimization, while the optimization solution itself is not targeted.

The paper starts with theoretical introductions that present the agent model in Section II and utilized hierarchical optimization concept in Section III. Next, Section IV presents selected the most valuable results, while Section V concludes the paper and shows identified research opportunities.

II. Transport agent-based modeling

The developed agent model is implemented using the Python and MESA framework [9]. Agent-based model structure consists of two elements: the environment in which agents operate and the agents with their functions. Discussing the environment, we must focus on three aspects: the geospatial space in which the agents move, its updating mechanism for accidents simulation, and the representation of time. The directed graph G is defined as

$$G = < V, A > \tag{1}$$

*This work was not supported by any organization

Warsaw University of Technology, Institute of Control and Computation Engineering, Nowowiejska 15/19, 00-665 Warsaw, Poland patryk.ploski.stud@pw.edu.pl; kacper.radzikowski2.stud@pw.edu.pl; pawel.domanski@pw.edu.pl

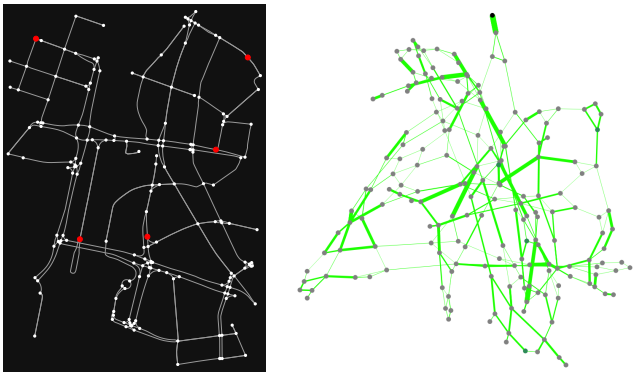
where V is a set of vertices representing the ends of streets and intersections and A represents set of edges, i.e. sections of road between intersections. Graph nodes store information about the longitude, latitude, and the number of road segments connected to a given node. On the other hand, the edges keep information about:

- road ID in the OSM system,
- number of lanes,
- street names,
- the type of road (e.g. highway, residential road),
- maximum speed limit (if the OSMnx library connects two sections of road with different speed limits during built-in graph simplification, a list of these maximum speed limits is included),
- whether the road is one-way,
- whether it is possible to turn around on it,
- crossing speed (if, as part of graph simplification, the OSMnx library connects two road sections with different speed limits, e.g., at the exit of a built-up area, then this parameter is an average of these maximum allowable speeds),
- length of the section (in meters),
- road section travel time in ideal conditions,
- actual travel time,
- modifier of the travel time.

The directed graph is generated from the road data contained in the open source mapping (OSM) system using the OSMnx library, which obtains data from the OSM API and converts the information to the directed graph implemented by the NetworkX library [10]. Directed graph allows to take into account the direction of road movement, and allows to distinguish which direction of travel has a slower travel time due to the occurrence of any incident. Fig. 1a shows sample graph generated using the OSMnx library.

The graph comes to the *Spaces* MESA development platform module. Then, using the geographic coordinate data of the stores, we find the graph node located closest to each of the stores. Finally, the graph space is initialized. Fig. 1b shows such a fully initialized graph space using a model visualization tools from MESA. Unfortunately, such graph does not preserve real geometry. Despite that inconvenience, MESA solution ensures fast system operation and represents road networks very well. Therefore, this solution is further used.

We have to remember the the reality changes and the traffic situation evolves due to various reasons. Some road sections may be closed or due to the high traffic the traveling time may significantly increase. The slowing down of traffic may be caused by factors such as stops at traffic lights, traffic jams or accidents, what makes travel times non-deterministic. Thus, any ABM wishing to accurately simulate varying traffic conditions must account for this randomness. Accidents cause additional challenge. They are important as they slow down traffic much more severely and do so predictably longer than the others. Thus, two update mechanism are designed: one responsible for updating the space in general and the other responsible only for handling accidents.



(a) Real road connection (b) MESA graph representation

Fig. 1: Graphs for small map section. Red dots denote stores

A. General traffic conditions adaptation

Graph adaptations is done according to the procedure shown in Fig. 3a. The terms "update period" and "accident threshold", which appear in it, are model parameters that the user can define. We may specify general percentage of roads that are updated every specified number of simulation steps. The number k is thus defined as

$$k = round(|\text{Road\_List}| \cdot \text{Update\_Per}). \tag{2}$$

The parameterization allows to adjust the number of changes in the traffic volume to the simulation step time. The term "base travel time" reflects the ideal travel time, which is the road length of divided by the maximum allowable speed. The travel time modifier is a positive real number drawn from the K4P random number generator [11]

$$f(x) = \begin{cases} \sigma^{-1}(1-ky)^{(1/k)-1}(F(x))^{1-h} & \text{for } k \neq 0, h \neq 0, \\ \sigma^{-1}(1-ky)^{(1/k)-1}F(x) & \text{for } k \neq 0, h = 0, \\ \sigma^{-1}exp(-y)(F(x))^{1-h} & \text{for } k = 0, h \neq 0, \\ \sigma^{-1}exp(-y)F(x) & \text{for } k = 0, h = 0, \end{cases} \tag{3}$$

where,

$$F(x) = \{1-h[1-k(x-\mu)/\sigma]^{1/k}\}^{1/h}, \tag{4}$$

$$y = \frac{(x-\mu)}{\sigma}. \tag{5}$$

The probability density function is parameterized by four factors: the shift μ, the scale σ and two shape coefficients: h responsible for the left function tail and k, which affects the right tail shape. The K4P distribution is a family that includes many other probability distributions and is useful in modeling phenomena with one-sided distribution, mostly extreme ones [12]. Fig 2 shows adopted K4P shape.

B. Accidents model

The accidents model is programmed according to the procedure sketched in Fig. 3b. The mechanism stabilizes the

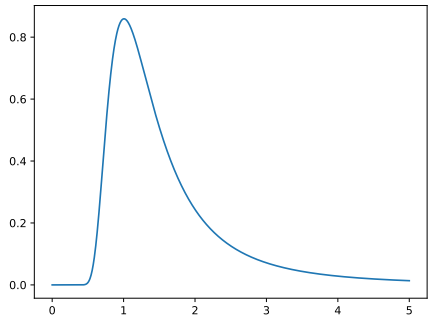
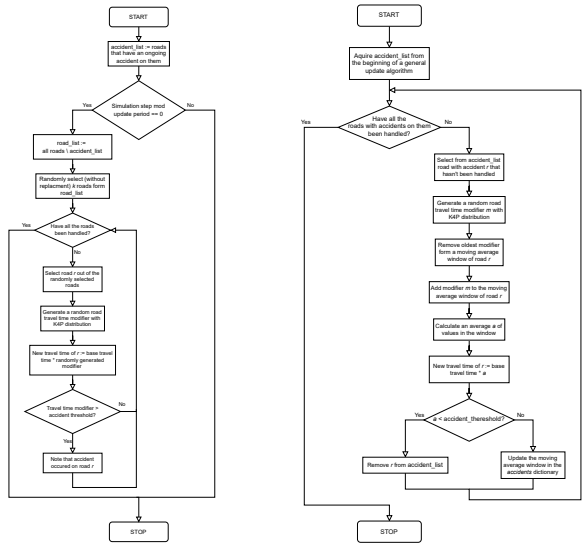


Fig. 2: K4P: $\mu = 1.164$, $\sigma = 0.461$, $h = -0.0235$, $k = -0.437$

number of accidents and addressing around $1.5-2.0\%$ of all edges, which is quite realistic. The implemented mechanisms rationally do their task without being over-complicated.



(a) General traffic conditions (b) Accidents

Fig. 3: Graph adaptation mechanisms

C. Agents

Four types of agents are used in the system: `DistributionPoint`, `DeliveryTruckAgent`, `DriverAgent`, and `TrailerAgent`. Agent `DistributionPoint` represents the warehouse locations, i.e. both the DC distributing goods to stores and the stores themselves. Agent `DeliveryTruckAgent` represents the truck tractor, while `DriverAgent` the driver, who runs the truck. By law, truck drivers are required to take breaks while driving. Due to the complexity of the actual regulations, it is assumed that drivers always have to take a break of a length also specified by the user. The last is the `TrailerAgent` agent, who represents trailers with cargo. It gives us an advantage to swap an emptied trailer for a full one at the regional warehouse.

III. Distributional transport optimization

Having operational distribution transport ABM, we may come to the next step: goods delivery optimization. The problem, for a given number of truck $n_t \in \mathbb{N}, n_t > 0$, defined end points and single starting point (with index $n+1$) $P = \{p_1, p_2, \ldots, p_n, p_{n+1}\}$, which are located in the vertices of a directed graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n, v_{n+1}\}$, for which the edges are weighted according to the travel time

$$E = \{e_{1|2}, \ldots e_{1|(n+1)}, \ldots, e_{(n+1)|1}, \ldots, e_{(n+1)|n}\} \quad (6)$$

$$T = \{t_{1|2}, \ldots t_{1|(n+1)}, \ldots, t_{(n+1)|1}, \ldots, t_{(n+1)|n}\} \quad (7)$$

is to evaluate a list of endpoints for each truck, such as

$$\mathbf{L} = \{\mathscr{L}_1 \subseteq \{l_{1|1}, \ldots, l_{1|n_{l1}}\}, \ldots, \mathscr{L}_{n_t} \subseteq \{l_{n_t|1}, \ldots, l_{n_t|n_{ln_t}}\}\}, \quad (8)$$

for $\forall x, y, x', y' \quad (l_{xy} \neq l_{x'y'}, \ n \geq l_{xy} > 0, \quad l_{xy} \in \mathbb{N}, \ (x,y) \neq (x',y')$ and $\sum_{i=1}^{n_t} n_{li} = n$, where l takes the value of point p index, for function f_g defined as

$$f_g(\mathbf{L}) = \max\{f_l(\mathscr{L}_1), f_l(\mathscr{L}_2), \ldots, f_l(\mathscr{L}_{n_t})\}, \quad (9)$$

where

$$f_l(\mathscr{L}_j) = \sum_{i=1}^{|\mathscr{L}|-1} [t_{l_i l_{j(i+1)}}] + t_{(n+1)l_1} + t_{l_{|\mathscr{L}|}(n+1)}. \quad (10)$$

The problem is visualized by sample graph in Fig. 4. It has vertices with end point, while last vertex with number 10 denotes starting point (distribution center) with

$$V = \{v_1, v_2, \ldots, v_9, v_{10}\}, \quad (11)$$

$$P = \{p_1, p_2, \ldots, p_9, p_{10}\}. \quad (12)$$

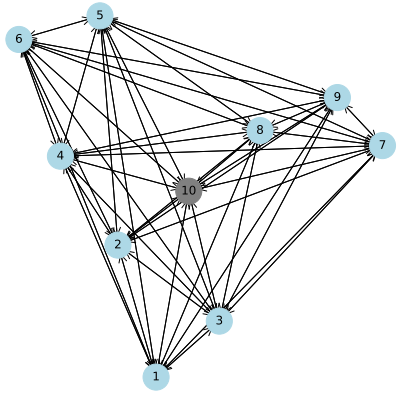


Fig. 4: Sample graph problem visualization

To assignment end points to each truck we need to minimize certain cost function $f_g(\mathbf{L})$. This function directly depends on $\mathbf{L}$, i.e. how the values $\mathscr{L}_1, \ldots, \mathscr{L}_{n_t}$ are selected. All endpoints indexes ore in the lists $\mathscr{L}$, so we understand the search space as

$$S_L = \{\mathbf{L_1}, \mathbf{L_2}, \ldots, \mathbf{L_{n_L}}\}, \quad (13)$$

where $\mathbf{L}$ takes values from

$$\mathbf{L_1} = \{\mathscr{L}_1 \subseteq \{l_{11}, \ldots, l_{1n_{l1}}\}, \ \mathscr{L}_2 = \emptyset, \mathscr{L}_{n_t} = \emptyset\}\}, \quad (14)$$

to

$$\mathbf{L_{n_L}} = \{\mathscr{L}_1 = \emptyset, \ \mathscr{L}_2 \quad = \emptyset, \mathscr{L}_{n_t} = \{l_{n_t 1}, \ldots, l_{n_t n_{ln_t}}\}\}, \quad (15)$$

For known number of vertices n and number of trucks n_t, we may use Stirling numbers [13] regarding also empty sets to write

$$n_L = \sum_{k=1}^{n_t} [S(n,k) \cdot \binom{n_t}{k}], \tag{16}$$

where n_L is a number of possible values of $\mathbf{L}$.

The assumptions allow to define the task to be solved, however the initial graph does not include endpoints only. Thus the second problem must be defined, i.e. the evaluation of the best path between endpoints. Let's consider the base task of riding between two vertices. Thus, for a given set of vertices $T = \{t_1, t_2, \ldots, t_m\}$ and edges $E = \{e_1, e_2, \ldots, e_m\}$ with weights (travel times in seconds) $T = \{t_1, t_2, \ldots, t_m\}$, selected starting vertex v_s and end one v_e we need to find out the optimal linking path. We need to run these calculations with the shortest time. Let denote the path as

$$R = \{r_1, r_2, \ldots, r_{n_r}\}, \tag{17}$$

where e_p is the selected edge index, we obtain the cost function to be minimized

$$f_p(R) = \sum_{i=1}^{n_r} t_{r_i} \tag{18}$$

The solution contains undefined number of edges (one or more), what depends on the weights. The solution is organized in a hierarchical way as shown in Fig. 5.

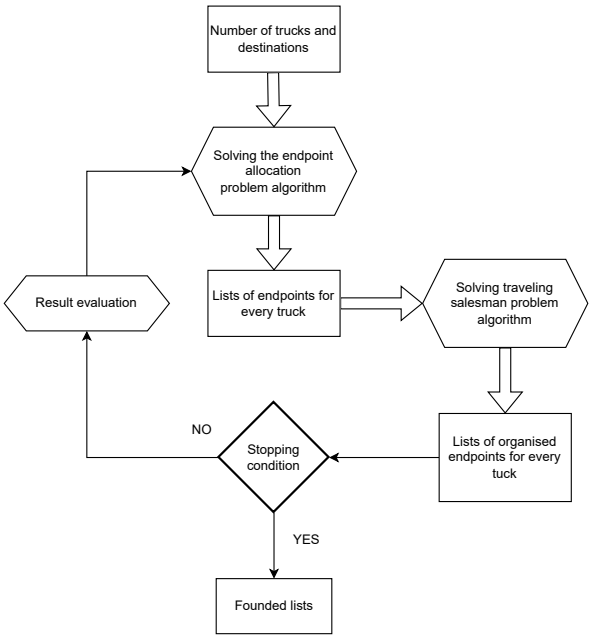


Fig. 5: Hierarchical solution scheme

A. Global optimization problem

The global optimization problem can be solved in many ways. Due to its nature, several approaches can be discarded. First, it's impossible to review all solutions because the time required to do so, even for a relatively small number of endpoints and trucks,expands dramatically and is generally unrealistic for real problems. Next, it is unreasonable to assume that the minimized cost function would have only one local optimum, which would also be the global one. Thus simple fastest descent or simple gradient algorithms are not considered. The best group of algorithms for the global task therefore seems to be heuristic ones, of which simulated annealing [14] and A* [15] are assessed.

The simulated annealing is a well-known stochastic search algorithm. We apply it to look for a predefined solution space, where a single point is represented as a list of endpoints with given indexes. The algorithm requires to define the neighborhood for a point located in the searched space. A neighbor of a given point is defined as any point that differs from the current one by changing the assignment of one store to another truck. Starting point is selected randomly from the search space. It achieves greater confidence that the space is well explored during several experiments. In the implementation, the best found point is saved, so it is not lost when finding potentially worse solutions.

The A* algorithm is usually used to browse a graph to find the cheapest path. Thus the task must be reviewed from a different perspective, i.e. the previously formulated problem has to be re-translated to the tree search problem.

B. Traveling salesmen problem

The traveling salesmen problem requires different approaches. This works compares three algorithms: the nearest neighbor (NNA), greedy algorithm (GrA) and the Christofides algorithm (ChA). The nearest neighbor algorithm uses below procedure:

1) Take general directed graph.
2) Take starting vertex (trucks start point) and set it as actual.
3) Add actual vertex to the list.
4) For an actual vertex take the cheapest edge to unvisited vertices.
5) Define the vertex as visited and set it as actual.
6) If all vertices are visited, then add the initial vertex at the end of the list and return the list.
7) Repeat points from 3 to 6.

In the greedy algorithm, as the name suggests, the approach tries to include the shortest available edges in the path. Unlike the nearest-neighbor algorithm, here there are no parameters such as the starting point that would force the path determination to be run repeatedly. This is an algorithm that is used more widely, and is sometimes used with minor modifications, but for the purposes of this work, the focus is on the classic version of [16].

The Christofides algorithm [17] is compared as a third option. Though the algorithm is relatively complex, it is mathematically proven that the cost to find path is not be greater than the cost of the path in the smallest spanning tree multiplied by 1.5. Just as in the greedy algorithm it is assumed that it's better for the algorithm to take a

non-directed graph. We use Prims algorithm [18] to find a minimum spanning tree.

C. Graph preprocessing

Previously described methods rely on important assumption, that a graph contains endpoints at each vertex. However, the map generated in Section II is far from such an assumption. Thus, an original map detailed graph with multiple intermediate vertices between endpoints must be pre-processed. Moreover, for the purposes of the ABM, we need to evaluate exact routes for each truck, not just the nearest endpoint to reach. The path should also change dynamically, responding to varying model parameters, i.e. changing travel times – edges' weights.

The classic approach is to use the A* algorithm, which finds the cheapest path in a graph. The algorithm does not review all possible paths, which might be utterly time-consuming, but only those indicated as potentially better. The algorithm is similar as in the global optimization task, however the graph structure remains unknown.

Dijkstra's algorithm is a well-known algorithm for path finding. It differs from the A* algorithm in assumptions, as it finds the cheapest path for each vertex in the graph [19].

IV. Experiments and results

All the experiments are performed on a graph representing the city of Poznań and its surroundings. The detailed graph consists of 17712 vertices and 40958 edges. In addition, 109 endpoints and one starting point (DC) are given, which naturally indicates that after reprocessing, the general graph will contain 110 vertices, with all edges defined with positive weights between them. All calculations are carried out on an Intel core i5 4590 processor at 3.7GHz and 16GB of RAM.

A. Traveling salesman problem (TSP) – results

The performance, execution time and associated problems are compared for three selected algorithm. The task is to optimize the path for a single truck with a varying number of endpoints. The algorithm is executed once, as the task is deterministic. The algorithms are compared in Fig. 6, which shows how estimated travel time is affected by the number of endpoints. Execution times are compared in Table I.

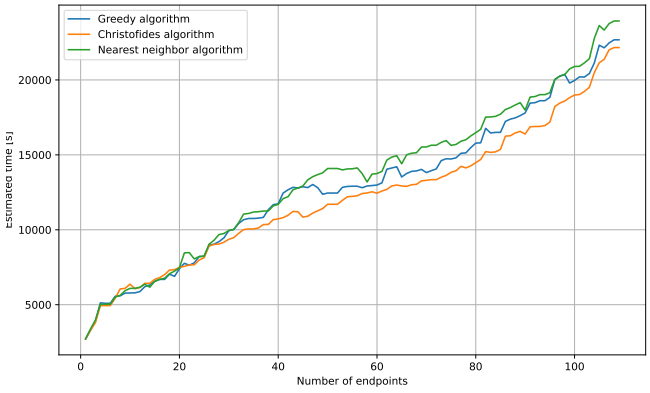


Fig. 6: Comparison of the TSP algorithms

TABLE I: TSP solution execution times

algorithm:	nearest neighbor	greedy	Christofides
mean time [s]:	$1.24 \cdot 10^{-3}$	$3.08 \cdot 10^{-2}$	$4.62 \cdot 10^{-2}$
total time [s]:	$1.36 \cdot 10^{-1}$	3.35	5.04

The comparison shows that the Christofides algorithm solves the problem the most effectively. The nearest neighbor and greedy exhibit better results in case for smaller graphs with less than 25 endpoints. These routes are crucial for the algorithm. We also observe that execution times favors especially the nearest neighbor algorithm.

B. Global optimization – results

Global optimization algorithms cannot be evaluated independently, as they use the cost function optimized by the TSP algorithms. Therefore, their comparison should be done in combination with selected TSP algorithm. At first, we compare the estimated time of accomplishing the simulations depending on number of trucks and the type of the cost function, i.e. the TSP technique. Fig. 7 compares the simulated annealing algorithm. Execution times are compared in Tables II and III. The more complex the algorithm is, the longer the calculations take on average. The best algorithm in terms of optimization quality for the A* algorithm along with acceptable time is the nearest neighbor algorithm. As for the simulated annealing, the results are very similar. Depending on the number of trucks, the greedy, nearest neighbor, or Christofides algorithm give better results. Surprisingly good results are achieved by the nearest neighbor algorithm.

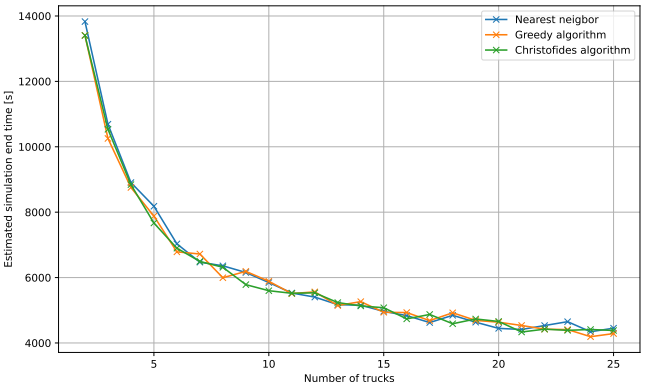


Fig. 7: Simulated annealing with different TSP algorithms

TABLE II: Simulated annealing execution times [s]

n_t	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**	**21**	**25**
NNA	2	2	1	1	1	1	1	1	1	1	1
GrA	41	24	17	14	12	11	10	10	9	9	9
ChA	75	44	34	33	28	28	29	27	27	29	29

C. Graph pre-processing results

The pre-processing approaches are compared in Table IV. We see at that point the superiority of Dijkstra method. However, in task, in which we are looking for a path from a specific vertex A to B, the A* algorithm, which heuristically

TABLE III: A* execution times [s]

n_t	**3**	**5**	**9**	**11**	**15**	**17**	**21**	**25**
NNA	1	4	15	12	24	32	45	65
GrA	381	158	333	616	1331	2022	2559	3519
ChA	75	223	533	745	1120	1478	4352	3748

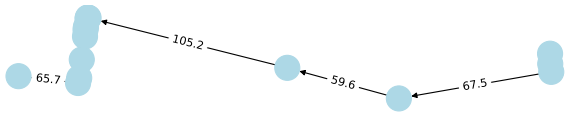
looks specifically for one path, gets an advantage. The difference for one truck might be negligible, but for 30 trucks it matters. Concluding the A* algorithm is indicated as the best fitting to that task.

TABLE IV: Time of general graph generation

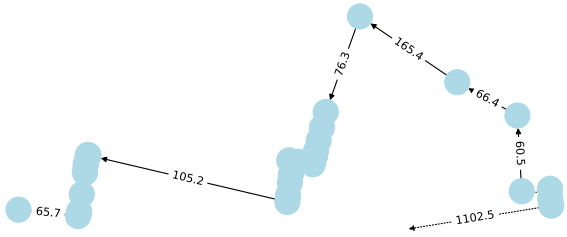
graph size	**A***	**Dijkstra**
40958 edges, 17712 vertices	577,8s	11,2s
7948 edges, 3315 vertices	2,0s	0,2s

D. Dynamic optimization

Finally, we have conducted the experiment with dynamic operation solving the problem in case of an accident. Fig. 8a shows the optimal route before an accident, while Fig. 8b after it. After the accident happened the algorithm bypasses the edge, with weight 1102.5, and chooses a detour. The edge is left in the figure so that one can see the difference.



(a) Before accident



(b) After accident (dotted line – inaccessible route)

Fig. 8: Visualization of dynamic optimization with accident

V. Conclusions and Further Research

The paper presents the results of the project that aims at building dynamic agent-based model environment for distributional transport modeling. Additionally, it includes the dynamic optimization algorithm, which allows to be applied during online vehicle routing task.

The system operates with embedded uncertainties that allow to model slowly varying traffic scenarios and rapidly happening accidents that may close the road. It is especially important for truck drivers, as they cannot make sharp u-turns or turn back. They must remain stuck till the congestion is cleared by road services in case of the road blocked.

The results are confirmed with the real road network for the city of Poznań (Poland) and its surroundings. The potential of the ABM system is very large. The next steps include its combination with the distribution center agent-based modeling to solve coordinated task of warehouse operation and goods transportation to final stores/customers.

References

[1] P. Ribino, M. Cossentino, C. Lodato, and S. Lopes, "Agent-based simulation study for improving logistic warehouse performance," *Journal of Simulation*, vol. 12, no. 1, pp. 23–41, 2017.

[2] T. Pulikottil, L. Estrada-Jimenez, H. Ur Rehman, F. Mo, S. Nikghadam-Hojjati, and J. Barata, "Agent-based manufacturing – review and expert evaluation," *The International Journal of Advanced Manufacturing Technology*, vol. 127, pp. 2151–2180, 2023.

[3] G. O. Kagho, M. Balac, and K. W. Axhausen, "Agent-based models in transport planning: Current state, issues, and expectations," *Procedia Computer Science*, vol. 170, pp. 726–732, 2020, the 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[4] K. Nagel and F. Marchal, "Computational methods for multi-agent simulations of travel behavior," in *10th International Conference on Travel Behaviour Research*, Lucerne, Switzerland, 2007, Resource paper. Workshop on Computational Techniques.

[5] M. Poeting, S. Schaudt, and U. Clausen, "Simulation of an optimized last-mile parcel delivery network involving delivery robots," in *Advances in Production, Logistics and Traffic. ICPLT 2019. Lecture Notes in Logistics.*, U. Clausen, S. Langkau, and F. Kreuz, Eds. Cham, Switzerland: Springer, 2019.

[6] J. R. M. Hosking, "The four-parameter kappa distribution," *IBM Journal of Research and Development*, vol. 38, no. 3, pp. 251–258, 1994.

[7] J. Vokřínek, A. Komenda, and M. Pěchouček, "Agents towards vehicle routing problems," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1*, ser. AAMAS '10, Toronto, Canada, 2010, pp. 773–780.

[8] B. H. Ojeda Rios, E. C. Xavier, F. K. Miyazawa, P. Amorim, E. Curcio, and M. J. Santos, "Recent dynamic vehicle routing problems: A survey," *Computers & Industrial Engineering*, vol. 160, p. 107604, 2021.

[9] J. Kazil, D. Masad, and A. Crooks, "Utilizing python for agent-based modeling: The mesa framework," in *Social, Cultural, and Behavioral Modeling*, R. Thomson, H. Bisgin, C. Dancy, A. Hyder, and M. Hussain, Eds. Cham: Springer International Publishing, 2020, pp. 308–317.

[10] G. Boeing, "Modeling and analyzing urban networks and amenities with osmnx." 2024, working paper. [Online]. Available: https://geoffboeing.com/publications/osmnx-paper/

[11] P. D. Domański, *Back to Statistics. Tail-Aware Control Performance Assessment*. Boca Raton, FL: CRC Press, Taylor & Francis Group, 2025.

[12] M. Giełczewski, M. Piniewski, and P. D. Domański, "Mixed statistical and data mining analysis of river flow and catchment properties at regional scale," *Stochastic Environmental Research and Risk Assessment*, vol. 36, pp. 2861–2882, 2022.

[13] A. A. Jose and B. Beata, "Probabilistic stirling numbers and applications," *Aequationes Mathematicae*, vol. 98, pp. 1627–1646, 2024.

[14] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," in *Handbook of Metaheuristics*. Springer International Publishing, 2019, pp. 1–35.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[16] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "A formal basis for the heuristic determination of minimum cost paths," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427–441, 2011.

[17] A. Haddadan and A. Newman, "Towards improving christofides algorithm on fundamental classes by gluing convex combinations of tours," *Mathematical Programming*, vol. 198, pp. 595–620, 2023.

[18] X. Yang and L. Zhang, "Target-based distributionally robust minimum spanning tree problem," 2023, arXiv:2311.10670v1 [math.OC]. [Online]. Available: https://arxiv.org/abs/2311.10670

[19] B. Haeupler, R. Hladík, V.Rozhoň, R. Tarjan, and J. Tětek, "Universal optimality of Dijkstra via beyond-worst-case heaps," 2024, arXiv:2311.11793v3 [cs.DS]. [Online]. Available: https://arxiv.org/abs/2311.11793