Technical Sciences, 2026, 29, 77–94

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE

Yearbook peer-reviewed scientific journal
ISSN 1505-4675
e-ISSN 2083-4527
TECHNICAL SCIENCES
Homepage: https://czasopisma.uwm.edu.pl/index.php/ts/



DOI: https://doi.org/10.31648/ts.11919

ANALYSIS OF SELECTED PATHFINDING ALGORITHMS IN 2D POROUS MEDIA WITH BINARY REPRESENTATION

Filip Klimek, Wojciech Sobieski

[1]ORCID: 0000-0003-1434-5520
Faculty of Technical Sciences, University of Warmia and Mazury in Olsztyn, Poland

Received 19 October 2025, accepted 29 April 2026, available online 19 October 2026.

K e y w o r d s: porous media, tortuosity, porosity, path searching algorithms.

A b s t r a c t

The article presents an analysis of the influence of geometric parameters of a porous medium represented in a binary form – grid size, obstacle width, and porosity – on the tortuosity of transport paths and the efficiency of pathfinding algorithms. Numerical simulations were carried out for grids of 100–200 nodes, obstacle widths of 1–13 nodes, and porosity values ranging from 0.9 to 0.5, using the Dijkstra, A*, BFS, and Greedy BFS algorithms. The results confirmed the existence of a percolation threshold at $\phi \approx 0.6$ and showed that decreasing porosity increases path tortuosity. For high porosity ($\phi = 0.9$), paths were nearly straight ($\tau \approx 1.03$), while for low porosity ($\phi = 0.5$–0.6) they became highly tortuous ($\tau > 1.3$). Among the tested methods, the A* and Greedy BFS algorithms proved to be the most computationally efficient, confirming the effectiveness of heuristic approaches in modeling transport phenomena in porous structures.

Introduction

The concept of tortuosity was introduced by KOZENY (1927), who observed that when calculating pressure drops associated with fluid flow through porous media, it is necessary to consider not the layer thickness L_0, but the actual path

Correspondence: Wojciech Sobieski, Wydział Nauk Technicznych, Uniwersytet Warmińsko-Mazurski, ul. Oczapowskiego 11, 10-710 Olsztyn, wojciech.sobieski@uwm.edu.pl

length traveled by the fluid L_p. The degree of curvature of this path relative to the measurement section was defined as tortuosity:

$$\tau = \frac{L_p}{L_0} \tag{1}$$

where τ is a dimensionless quantity, whose value is always greater than or equal to 1.

A few years later, CARMAN (1937) pointed out that a similar correction is also required for flow velocity, since in tortuous channels the fluid moves faster than it would along a straight path. Both concepts led to the formulation of the equation now known as the Kozeny-Carman equation:

$$\frac{dp}{dx} = C_{KC} \cdot \tau^2 \cdot S^2_{0,Carman} \cdot \frac{(1-\phi)^2}{\phi^3} \cdot \left(\mu \cdot v_f\right) \tag{2}$$

where: p – pressure [Pa], x – a coordinate along which the pressure drop occurs [m], τ – tortuosity [–], C_{KC} – a model constant, usually equal to 5.0 (CARMAN 1937), $S^2_{0,Carman}$ – specific surface of the porous body in Carman meaning [1/m] (it is the inner surface of the solid body divided by the total volume of the solid part), ϕ porosity [–], μ – fluid viscosity [Pa·s], v_f – filtration velocity [m/s].

In the literature, one can also encounter the concept of the tortuosity factor ($\tau_f = \tau^2$), which is sometimes used interchangeably with the square of tortuosity. However, these quantities should be clearly distinguished to avoid misunderstandings and computational errors.

A key challenge in determining tortuosity is the evaluation of L_p. This task is non-trivial, and numerous approaches have been proposed in the literature. The complexity of the problem is illustrated in the study by FU et al. (2021), in which the authors reviewed and summarized 203 publications devoted to methods of determining tortuosity in porous media. Tortuosity determined directly from the shape of pore channels is referred to as geometric tortuosity. There are also other methods based on the analysis of transport, diffusion, electrical, or acoustic properties, but these approaches are not considered in the present work.

The studies presented in this work focus on methods for calculating geometric tortuosity in two-dimensional geometries represented in a binary form. This representation involves creating a regular grid of nodes (a matrix of size $L \times L$), each assigned a value of 0 or 1. A value of 0 indicates that the node belongs to the pore space, while 1 represents a part of the solid skeleton. Skeleton elements can have a size of $w \times w$, where w denotes the number of nodes forming a single obstacle in a given direction. The analyses were conducted using 125 combinations of the parameters L, ϕ and w, with each configuration repeated 50 times. The primary

objective of the study was to analyze the influence of these parameters on path tortuosity. The presented results extend and continue previous research (SOBIESKI 2019, 2020).

The concept of applying pathfinding algorithms to binary representations of porous media is not new (KOPONEN et al. 1996, 1997, MATYKA et al. 2008). However, the novelty of this work lies in the selection of algorithms, their original implementation, and the development of an approach enabling large-scale data analysis and processing. Additionally, a rarely considered factor in the literature – the size of the elementary skeleton element – was incorporated, allowing for a more faithful representation of real porous structures. The approach used not only broadens the scope of tortuosity research but also opens new possibilities for its practical application, particularly in the context of numerical modeling of transport processes in complex porous media.

Materials and methods

Materials

The subject of the study is a set of porous media represented in binary form (Fig. 1), generated randomly using proprietary software. It was assumed that the computational domain is always square, as is the unit skeleton element. Periodic boundary conditions were applied to the porous structures, meaning that if a skeleton element is located at the edge of the domain and does not fit entirely within it, its continuation is automatically added on the opposite side (an example of such a situation is shown in the figure). The significance of the parameters L and w was discussed in an earlier section of the work.

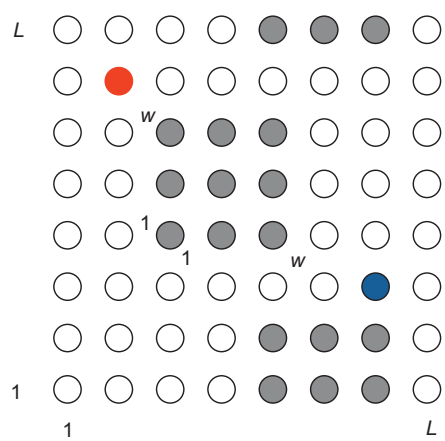


Fig. 1. Schematic of a porous medium in binary representation
Source: own work.

In each generated structure, a starting node (marked in red in the figure) and an ending node (marked in blue) were defined, between which the path is searched. An additional assumption was made that the path cannot cross the domain boundary.

A* algorithm

The A* algorithm is a standard method for solving optimal pathfinding problems, combining Dijkstra's approach with a heuristic estimator of the distance to the goal, which significantly speeds up the search process. The method guarantees finding a solution with minimal cost. Its operation is based on evaluating each node using the cost function $f(n) = g(n) + h(n)$, where $g(n)$ is the actual cost from the start node, and $h(n)$ is the heuristically estimated cost to the goal.

In the implemented version, due to the restriction to orthogonal movement only, the Manhattan metric was used for the heuristic estimation: $h(x, y) = |x - x_{goal}| + |y - y_{goal}|$. In each iteration, the node with the smallest $f(n)$ is selected from the priority queue (value 58 in Fig. 2). The node's neighbors are then validated, excluding obstacles and violations of boundary constraints. If a new path to a neighbor has a lower $g(n)$ cost, its parameters are updated, and the node is added to the queue with the new priority (Fig. 3). The process continues until the goal is reached, after which the optimal path is reconstructed by tracing backward from the end node to the start node (Fig. 4).

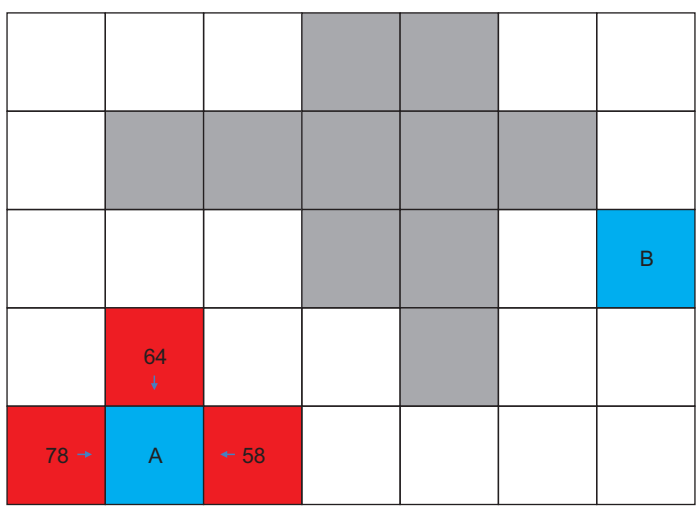


Fig. 2. Visualization of the A* algorithm in a sample 2D geometry during the first iteration
Source: own work.

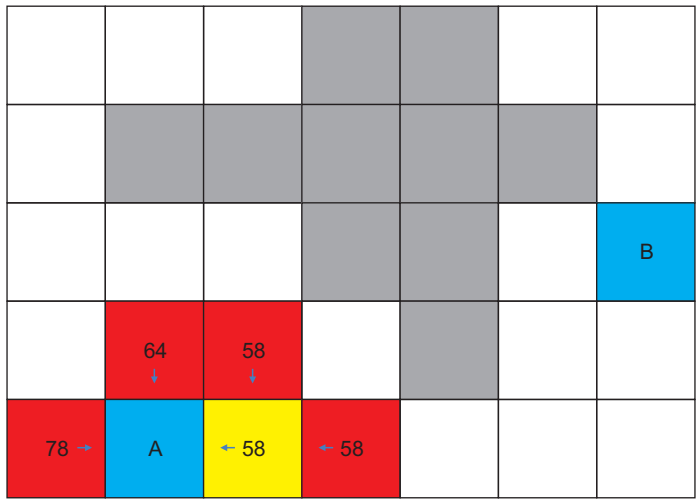


Fig. 3. Visualization of the A* algorithm in a sample 2D geometry during the second iteration
Source: own work.

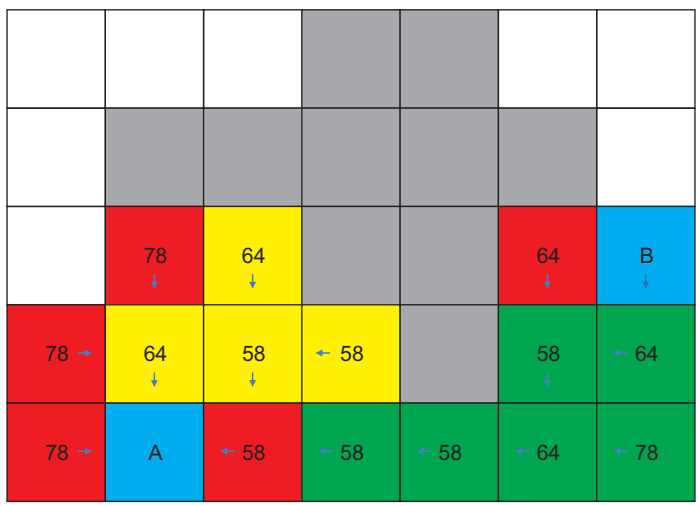


Fig. 4. Result of the A* algorithm in a sample 2D geometry
Source: own work.

Dijkstra's algorithm

Dijkstra's algorithm is one of the most widely used methods for finding the shortest paths in a graph, relying solely on the minimization of the actual travel cost. Unlike the A* algorithm, Dijkstra's method does not use a heuristic

function. The procedure begins by assigning all nodes, except the start node, a cost of infinity. In each iteration, the node with the smallest current cost is removed from the priority queue. Its neighbors are then analyzed, and a new travel cost is calculated as the sum of the current node's cost and the transition weight (in the adopted grid environment, a constant cost of 1 is assigned for cardinal movement). If the new cost is lower than the previously known value, it is updated, and the current node is set as the parent.

In the presented implementation, although the algorithm is based on Dijkstra's principles, it intentionally includes modifications that result in suboptimal search efficiency, such as omitting the formal structure of the closed node set. The primary purpose of this approach was to create a reference algorithm with increased computational cost, allowing for a reliable comparison of results with those obtained using other properly implemented and optimized search methods.

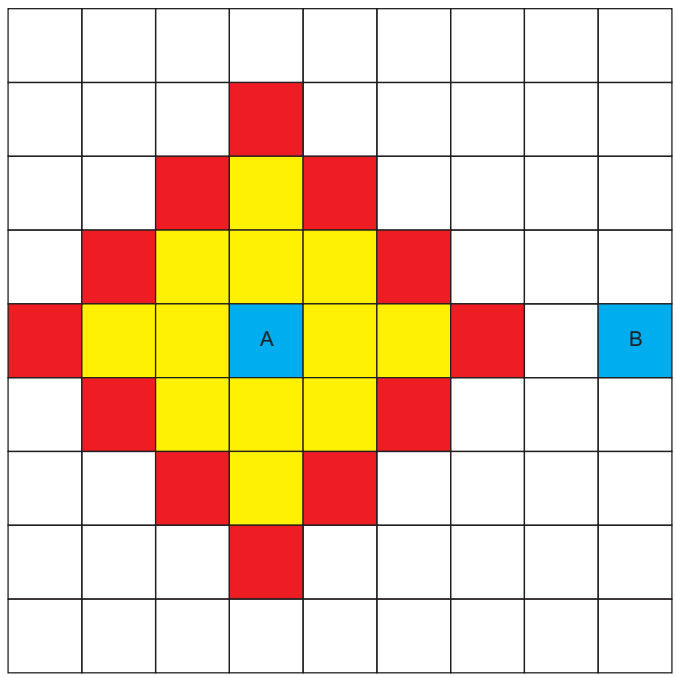


Fig. 5. Appearance of the BFS graph after the third iteration of the algorithm in a 2D obstacle-free space
Source: own work.

BFS algorithm

The BFS (Breadth-First Search) algorithm is another graph search method, which, in the context of pathfinding, can be considered a simplification of Dijkstra's algorithm. It is particularly efficient in environments where all transitions between nodes have the same cost (equal to 1). Under this assumption, there is no need to use a heuristic, resulting in a uniform wave-like expansion of the search in all directions from the start node, exploring the state space layer by layer.

Instead of a priority queue (used in Dijkstra and A*), BFS uses a standard FIFO (First In, First Out) queue. In each iteration, a node is dequeued, and its immediate neighbors (only in cardinal directions) are analyzed. Nodes that are not obstacles and have not yet been visited are immediately added to the queue, with the current node set as their parent. Due to the constant transition cost, BFS always finds the shortest path in terms of the number of edges (minimum number of steps).

The procedure continues until the target node is reached. The optimal path is then reconstructed by tracing back the parent-child relationships. This search strategy is computationally efficient, although in large, open spaces it may explore more nodes than heuristic-based algorithms. For example, in an unbounded space, the number of visited nodes after n steps grows proportionally to $4n$, as illustrated in Figure 5.

Greedy BFS algorithm

The Greedy BFS algorithm implements a greedy exploration strategy and can be considered a variant of the Breadth-First Search (BFS) algorithm, with a greedy approach. Unlike A*, the priority of a node in the queue is determined solely by the heuristic function $h(n)$, which estimates the distance from the node to the goal. The algorithm deliberately ignores the actual cost $g(n)$ from the start node. This strategy prioritizes speed of finding a path over path length, meaning that Greedy BFS does not guarantee an optimal path, but significantly reduces computation time.

The procedure begins with the initialization of a priority queue, where a node's priority is determined exclusively by $h(n)$. In the implemented version, the Manhattan heuristic was used due to cardinal movement. In each iteration, the node with the lowest $h(n)$ value is dequeued, and its neighbors are analyzed. Nodes that pass the same validation checks as in the A* algorithm (boundary verification and obstacle avoidance) are added to the queue, with their parent node recorded, which is essential for later path reconstruction. The process continues iteratively until the goal node is reached. The efficiency of this method depends directly on the quality of the chosen heuristic.

Results and discussion

Experimental design

The study adopted the following parameter space: number of grid nodes (L) in the X and Y directions: 100, 125, 150, 175, 200; size of the unit skeleton element (w): 1, 4, 7, 10, 13; porosity (ϕ): 0.9, 0.8, 0.7, 0.6, 0.5 [–]. For each combination of parameters, 50 independent repetitions were performed. A trial limit of 1000 was imposed to prevent infinite computation time in cases where a continuous porous channel between the start and end points could not be generated. The results were averaged and stored in a database. In total, 6250 individual results were obtained, excluding cases where a given parameter combination made it impossible to generate a valid path (e.g., when no connection existed between the start and end points).

Results

A summary of the results is presented in Tables 1–5. For all porosity values, $\phi \geq 0.6$ the algorithm consistently found valid solutions within the imposed limit of 1000 trials per iteration. In all 50 iterations for each parameter configuration, a grid satisfying the path existence condition was obtained. For porosity $\phi = 0.5$, in some cases it was not possible to generate a geometrical structure containing a connected pore space enabling path construction between the start and end points. The tables include only realizations in which a valid path was successfully obtained. This result indicates the presence of a distinct percolation threshold between $\phi = 0.5$ and $\phi = 0.6$ in the adopted structure generation model. This finding is consistent with previous observations (SOBIESKI 2019) and the literature. NEWMAN and ZIFF (2000) showed that for square lattices, the percolation threshold is 0.59275, which aligns with the values obtained in this study. A value of 0.595 is also reported by WANG and ZHOU (2013). Preliminary data analysis further revealed significant differences in the efficiency of the tested algorithms, which will be discussed in the following section of the work.

Influence of porosity and obstacle size on tortuosity

Figure 6 shows the relationship between tortuosity and porosity for five different obstacle widths and the smallest grid resolution. The analysis reveals a clear increasing trend – decreasing porosity leads to a systematic increase in tortuosity, regardless of the transverse dimensions of the obstacles. This result is consistent with literature data (KOPONEN et al. 1996, 1997, SOBIESKI 2019).

Table 1

Algorithm research results for grid size equal to 100

No.	Grid size [node]	Obstacle size [node]	Porosity [–]	Dijkstra		A*		BFS		Greedy BFS	
				path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]
1	100	1	0.9	147	186.078	106	4.425	106	36.100	112	13.482
2	100	1	0.8	296	154.579	117	9.104	117	39.602	149	20.853
3	100	1	0.7	320	77.430	134	10.095	134	28.194	186	15.716
4	100	1	0.6	301	21.636	214	12.993	214	16.256	223	11.064
5	100	1	0.5	–	–	–	–	–	–	–	–
6	100	4	0.9	126	895.159	104	5.042	104	49.491	118	27.956
7	100	4	0.8	185	331.086	110	6.191	110	34.315	140	18.509
8	100	4	0.7	242	85.730	119	7.743	119	29.542	157	15.706
9	100	4	0.6	276	75.561	130	9.027	130	24.673	183	15.502
10	100	4	0.5	316	41.194	159	13.995	159	23.107	194	15.908
11	100	7	0.9	132	633.255	104	3.639	104	35.000	124	18.774
12	100	7	0.8	169	373.211	111	5.690	111	30.878	144	17.595
13	100	7	0.7	215	254.786	119	12.215	119	41.594	178	24.650
14	100	7	0.6	208	103.418	127	7.610	127	22.952	167	14.437
65	100	7	0.5	261	49.085	149	14.645	149	24.465	189	18.186
16	100	10	0.9	136	466.954	104	4.760	104	45.369	123	18.390
17	100	10	0.8	160	236.603	109	6.753	109	38.374	158	19.816
18	100	10	0.7	220	258.996	116	8.109	116	35.816	177	24.730
19	100	10	0.6	253	81.446	127	10.141	127	28.714	177	18.996
20	100	10	0.5	250	73.964	137	10.783	137	22.575	177	16.021
21	100	13	0.9	165	307.895	105	5.180	105	48.138	134	21.141
22	100	13	0.8	174	321.279	110	4.829	110	30.574	161	17.859
23	100	13	0.7	193	188.177	118	7.565	118	27.804	181	17.521
24	100	13	0.6	241	80.946	133	10.905	133	25.556	189	16.905
25	100	13	0.5	246	57.430	144	11.336	144	21.818	183	15.653

Source: own work.

Table 2

Algorithm research results for grid size equal to 125

No.	Grid size [node]	Obstacle size [node]	Porosity [–]	Dijkstra		A*		BFS		Greedy BFS	
				path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]
26	125	1	0.9	179	788.127	134	6.517	134	53.742	143	24.493
27	125	1	0.8	353	447.409	146	15.678	146	70.720	181	35.750
28	125	1	0.7	449	116.180	167	14.766	167	44.077	280	24.997
29	125	1	0.6	418	36.753	262	18.043	262	22.801	297	16.138
30	125	1	0.5	–	–	–	–	–	–	–	–
31	125	4	0.9	188	2806.941	130	8.233	130	84.799	157	53.486
32	125	4	0.8	212	990.880	136	6.779	136	47.545	161	29.188
33	125	4	0.7	380	255.600	146	9.860	146	43.449	197	23.270
34	125	4	0.6	404	173.662	161	13.105	161	35.812	242	23.898
35	125	4	0.5	408	57.951	195	16.648	195	28.451	262	18.309
36	125	7	0.9	164	1001.839	130	4.756	130	50.731	159	20.144
37	125	7	0.8	233	1337.778	137	12.030	137	78.496	192	42.213
38	125	7	0.7	315	422.705	145	9.483	145	42.070	204	25.662
39	125	7	0.6	368	172.124	160	12.313	160	36.639	229	22.550
40	125	7	0.5	350	104.290	176	14.232	176	28.838	245	18.271
41	125	10	0.9	165	1431.209	130	4.673	130	51.935	167	24.086
42	125	10	0.8	236	627.226	135	6.457	135	47.081	203	21.921
43	125	10	0.7	335	389.660	151	12.048	151	45.141	239	27.161
44	125	10	0.6	368	270.374	156	12.331	156	39.166	230	26.568
45	125	10	0.5	375	102.714	187	17.277	187	31.225	263	20.905
46	125	13	0.9	185	442.933	130	5.218	130	50.572	193	17.743
47	125	13	0.8	236	769.926	137	7.979	137	47.630	211	29.043
48	125	13	0.7	305	362.424	144	9.165	144	41.408	237	23.432
49	125	13	0.6	343	334.322	154	17.277	154	54.289	230	36.107
50	125	13	0.5	352	114.928	180	15.798	180	29.175	254	19.571

Source: own work.

Table 3

Algorithm research results for grid size equal to 150

No.	Grid size [node]	Obstacle size [node]	Porosity [–]	Dijkstra		A*		BFS		Greedy BFS	
				path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]
51	150	1	0.9	252	3840.325	160	13.294	160	112.450	175	63.585
52	150	1	0.8	450	1497.159	174	22.188	174	102.042	224	58.627
53	150	1	0.7	532	646.859	200	34.794	200	104.060	322	57.032
54	150	1	0.6	566	99.747	311	45.231	311	56.804	374	36.629
55	150	1	0.5	–	–	–	–	–	–	–	–
56	150	4	0.9	202	5283.292	156	10.501	156	119.872	172	61.715
57	150	4	0.8	438	1503.535	163	14.410	163	107.144	220	57.384
58	150	4	0.7	491	1075.695	174	19.549	174	94.454	260	61.834
59	150	4	0.6	513	445.292	194	26.465	194	87.621	315	54.228
60	150	4	0.5	529	187.395	232	38.501	232	72.356	334	46.668
61	150	7	0.9	189	4035.617	156	9.818	156	115.950	188	49.214
62	150	7	0.8	393	1056.714	174	21.175	174	99.482	269	52.171
63	150	7	0.7	435	1591.322	172	19.757	172	91.818	268	59.163
64	150	7	0.6	507	420.038	191	27.729	191	88.474	323	54.769
65	150	7	0.5	498	243.271	217	33.977	217	71.333	323	47.385
66	150	10	0.9	226	4821.258	157	11.642	157	117.984	217	58.921
67	150	10	0.8	305	2215.373	162	14.683	162	113.467	248	55.858
68	150	10	0.7	367	994.337	174	19.423	174	96.204	276	50.055
69	150	10	0.6	436	285.669	189	26.327	189	87.144	297	47.499
70	150	10	0.5	452	271.708	218	37.148	215	70.872	323	44.243
71	150	13	0.9	213	2749.467	155	9.410	155	123.530	216	47.948
72	150	13	0.8	310	1388.168	164	16.376	164	109.893	274	51.064
73	150	13	0.7	387	1268.553	171	19.704	171	101.231	330	56.642
74	150	13	0.6	440	1028.456	185	27.289	185	90.740	354	68.543
75	150	13	0.5	462	191.971	209	30.004	209	65.807	317	43.412

Source: own work.

Table 4

Algorithm research results for grid size equal to 175

No.	Grid size [node]	Obstacle size [node]	Porosity [–]	Dijkstra		A*		BFS		Greedy BFS	
				path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]
76	175	1	0.9	356	1952.004	186	11.196	186	101.997	199	40.775
77	175	1	0.8	625	1376.919	203	20.192	203	91.945	289	55.841
78	175	1	0.7	768	1335.113	233	46.799	233	129.268	413	70.072
79	175	1	0.6	684	111.989	381	38.707	381	47.644	439	32.763
80	175	1	0.5	–	–	–	–	–	–	–	–
81	175	4	0.9	318	5203.649	182	8.031	182	96.244	212	50.189
82	175	4	0.8	543	2810.027	192	13.769	192	94.752	274	55.546
83	175	4	0.7	670	1340.078	203	33.736	203	154.482	334	110.286
84	175	4	0.6	650	1196.186	222	39.226	222	126.308	391	82.473
85	175	4	0.5	704	196.961	266	28.774	266	58.906	415	34.703
86	175	7	0.9	259	4505.749	182	8.209	182	96.618	238	42.693
87	175	7	0.8	404	4549.956	189	11.488	189	89.395	265	56.451
88	175	7	0.7	559	1979.501	200	17.663	200	91.370	339	56.783
89	175	7	0.6	588	837.868	222	25.786	222	88.736	382	51.180
90	175	7	0.5	627	532.867	251	45.649	251	101.496	398	71.076
91	175	10	0.9	241	5339.327	182	11.743	182	157.116	231	57.849
92	175	10	0.8	387	4419.289	193	22.715	193	143.389	309	91.381
93	175	10	0.7	612	1623.940	204	18.869	204	91.021	344	54.554
94	175	10	0.6	614	1205.097	216	36.368	216	123.030	397	81.410
95	175	10	0.5	594	228.558	247	30.772	247	65.405	399	41.690
96	175	13	0.9	288	7369.531	182	10.954	182	140.150	269	61.908
97	175	13	0.8	417	4367.959	192	23.044	192	148.367	316	79.121
98	175	13	0.7	546	3610.179	202	29.587	202	137.070	382	87.899
99	175	13	0.6	602	1574.586	220	38.785	220	114.996	428	76.860
100	175	13	0.5	600	433.813	245	26.740	245	60.823	366	44.492

Source: own work.

Table 5

Algorithm Research Results for Grid Size Equal to 200

No.	Grid size [node]	Obstacle size [node]	Porosity [–]	Dijkstra		A*		BFS		Greedy BFS	
				path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]	path length [node]	exploration time [ms]
101	200	1	0.9	474	6148.226	214	23.283	214	207.503	233	89.099
102	200	1	0.8	806	1419.232	233	40.302	233	191.537	308	103.132
103	200	1	0.7	901	1449.975	267	59.114	267	164.672	471	93.050
104	200	1	0.6	820	235.636	449	86.703	449	104.244	560	65.660
105	200	1	0.5	–	–	–	–	–	–	–	–
106	200	4	0.9	430	11178.171	208	12.395	208	164.142	252	87.704
107	200	4	0.8	666	3047.156	218	17.104	218	125.301	300	66.408
108	200	4	0.7	799	1161.343	232	28.278	232	137.336	403	78.758
109	200	4	0.6	741	1294.794	254	41.800	254	147.416	456	96.472
110	200	4	0.5	781	570.108	292	51.392	292	117.524	462	73.010
111	200	7	0.9	310	7764.951	207	14.178	207	190.489	273	72.928
112	200	7	0.8	608	8773.648	218	23.919	218	179.233	338	112.161
113	200	7	0.7	842	2217.360	229	32.176	229	165.990	395	104.249
114	200	7	0.6	838	741.352	250	33.438	250	109.040	450	70.982
115	200	7	0.5	676	931.822	280	52.031	280	124.680	445	72.715
116	200	10	0.9	369	9751.953	208	15.591	208	193.267	298	78.819
117	200	10	0.8	490	10008.315	219	27.189	219	183.422	382	107.373
118	200	10	0.7	618	6252.599	227	34.646	227	185.004	374	120.178
119	200	10	0.6	826	823.885	249	44.382	249	145.356	469	91.554
120	200	10	0.5	755	410.693	289	40.907	289	89.949	471	52.608
121	200	13	0.9	386	9170.718	210	13.204	210	144.228	354	64.694
122	200	13	0.8	483	8471.353	217	24.611	217	196.207	374	99.439
123	200	13	0.7	770	2829.703	235	39.312	235	174.989	480	105.015
124	200	13	0.6	711	1569.180	247	44.779	247	157.380	446	93.647
125	200	13	0.5	797	850.787	285	58.767	285	128.317	526	97.281

Source: own work.

For high porosity (ϕ = 0.9), tortuosity reaches values close to the minimum ($\tau \approx 1.03 - 1.06$), indicating nearly straight paths. In the case of low porosity ($\phi = 0.5 - 0.6$), tortuosity rises sharply to values above 1.3. It is worth noting that structures with point obstacles (w = 1) exhibit significantly higher tortuosity compared to configurations with larger obstacles. For obstacles of size 4 to 7 nodes, tortuosity values are similar, whereas for sizes 10 and 13, the relationship becomes more complex and nonlinear, particularly at low porosity levels.

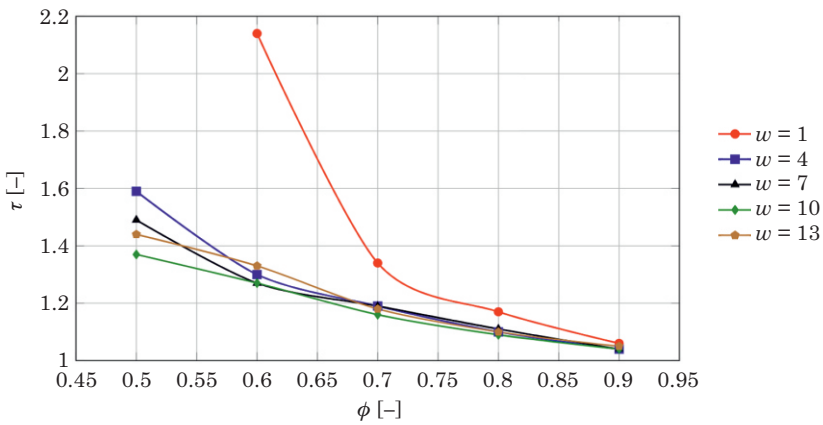


Fig. 6. Influence of porosity ϕ on tortuosity τ for different obstacle widths w (grid 100 ×100)
Source: own work.

Figure 7 shows the relationship between tortuosity and obstacle size for fixed porosity values. For low porosities ($\phi = 0.5 - 0.6$), a clear decrease in tortuosity is observed when transitioning from point obstacles to wider structures, followed by stabilization of the values. For higher porosities ($\phi = 0.7 - 0.9$), this relationship is less pronounced, and tortuosity remains at a relatively low, stable level.

An interesting phenomenon is observed for obstacle width w = 10, where tortuosity reaches a local minimum for most porosity values, suggesting the existence of an optimal scale of heterogeneity that minimizes path tortuosity. For w = 13, a slight increase in tortuosity is observed, particularly for ϕ = 0.5 and ϕ = 0.6, indicating a more complex dependency for larger obstacles.

Figure 8 presents the combined influence of grid size, porosity, and obstacle size on tortuosity for nine selected parameter sets. For most configurations, a slight but systematic decrease in tortuosity is observed with increasing grid size. This trend is particularly noticeable for low-porosity structures, where tortuosity is significantly higher. For higher porosities ($\phi = 0.8 - 0.9$), tortuosity values are lower and exhibit less variability. It is worth noting that for the configuration ϕ = 0.6 and w = 1, a slight increase in tortuosity with grid size was observed, which may indicate specific transport properties for this particular

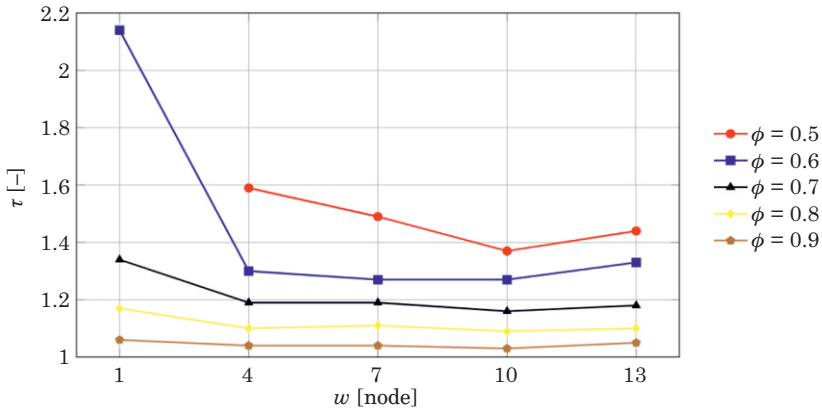


Fig. 7. Influence of obstacle width w on tortuosity τ for different porosity values (grid 100 × 100)
Source: own work.

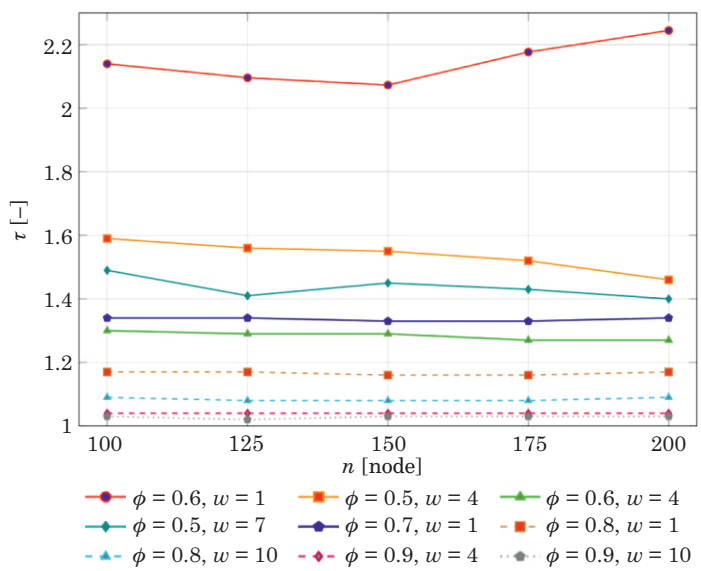


Fig. 8. Influence of grid size n on tortuosity τ or different parameter combinations
Source: own work.

parameter combination. A particularly interesting behavior occurs for $\phi = 0.5$ and $w = 7$, where a significant deviation from the general trend appears at a grid size of 125 nodes. This phenomenon does not occur for the similar configuration $\phi = 0.5$, $w = 4$, suggesting high sensitivity to obstacle geometry. The cause of the observed anomaly is not fully explained. It is suspected that it does not result from grid generation errors but may arise from a specific combination

of parameters, which around a grid size of 125 nodes leads to a local change in medium properties. This effect may be amplified by nonlinear interactions between porosity and obstacle geometry, where a small change in obstacle size (from $w = 4$ to $w = 7$) at constant porosity $\phi = 0.5$ can cause disproportionately large changes in path tortuosity. Similar unusual cases may also occur for other parameter configurations, but identifying them would require finding specific parameter combinations or dependencies, which can be very difficult to detect in standard analyses. In contrast, for the highest porosity ($\phi = 0.9$), tortuosity remains at a nearly constant, low level, regardless of grid size or obstacle width.

In addition to studying the influence of obstacle size and porosity on tortuosity in a porous medium, the objective of this work was also to identify the most optimal algorithm for this type of simulation and to compare it with competing algorithms in terms of both the quality of the found paths (tortuosity) and the computation time required to determine them. To identify the most efficient algorithm, two diagrams were created showing the pathfinding times for random porous media of various sizes (Fig. 9 and 10). The presented results represent the average times calculated from all data contained in Tables 1–5 for each grid size.

The Dijkstra algorithm proved to be the least efficient in the tested environment. The average time required to find a solution for a 200 × 200 node grid was over 100 times longer than for the A* algorithm. This significant discrepancy arises from the fact that Dijkstra explores nodes solely based on the criterion of minimizing the actual travel cost $g(n)$, i.e., the distance from the start node. The main factor contributing to its drastic reduction in efficiency, however, was a deliberate implementation modification. The algorithm was implemented suboptimally, including the omission of the formal structure of the

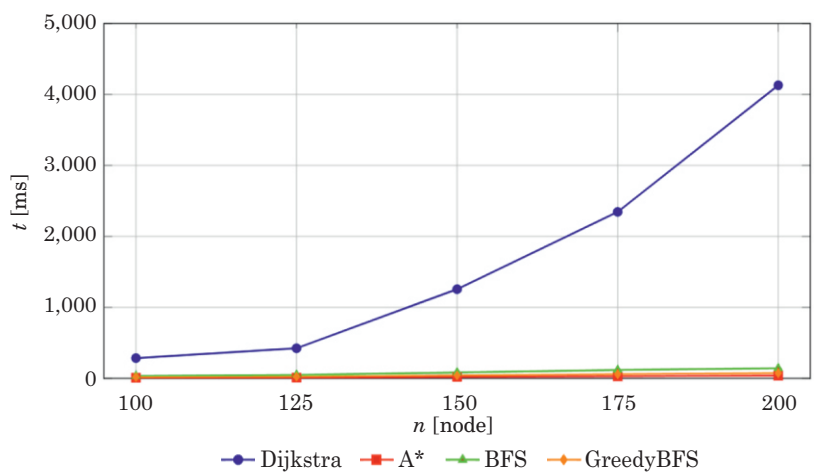


Fig. 9. Comparison of average execution times for all tested algorithms
Source: own work.

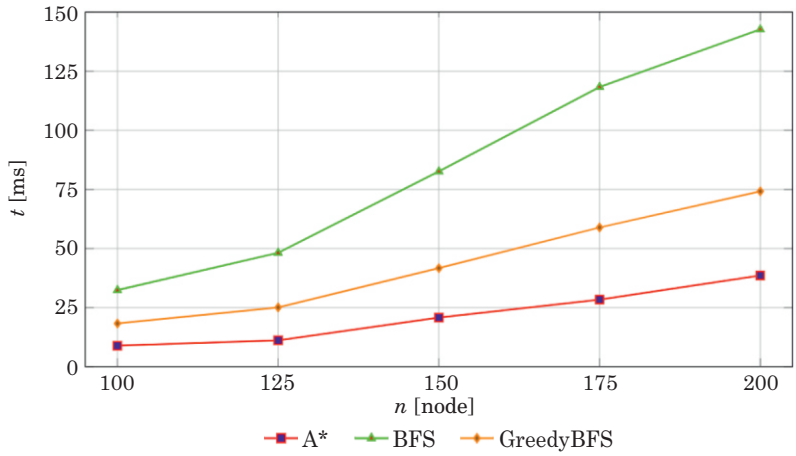


Fig. 10. Detailed comparison of average execution times for A*, BFS and Greedy BFS algorithms
Source: own work.

closed node set. Combined with the absence of any mechanism guiding the search toward the goal, this resulted in excessive, unguided exploration of the state space. This approach fulfilled the purpose of creating a reference algorithm with intentionally increased computational cost.

The A* and Greedy BFS algorithms, which use a Manhattan heuristic, were the fastest. The heuristic, by estimating the distance to the goal, effectively guided the search, drastically reducing the number of nodes explored. The BFS algorithm, lacking a heuristic but using a simpler and faster FIFO queue, performed intermediately – slower than the heuristic-guided algorithms but significantly faster than Dijkstra, which was burdened by the overhead of the priority queue.

Conclusion

The study investigated the influence of grid size, obstacle width, and porosity on the tortuosity of paths in porous media and evaluated the efficiency of various pathfinding algorithms. Numerical experiments were performed for grids of 100–200 nodes, obstacle widths of 1–13 units, and porosity values from 0.9 to 0.5. Each parameter combination was tested in 50 independent iterations, resulting in 6250 valid datasets.

For porosity values $\phi \geq 0.6$, valid paths were always found within the iteration limit, while for $\phi = 0.5$ no continuous path could be generated, indicating a percolation threshold between $\phi = 0.5$ and $\phi = 0.6$, consistent with literature values (≈ 0.59275). Decreasing porosity led to a systematic increase in tortuosity, reaching values above 1.3 for low porosity. Configurations with point obstacles

(w = 1) exhibited the highest tortuosity, whereas larger obstacles produced smoother, less tortuous paths. For most cases, increasing grid resolution slightly reduced tortuosity, though specific parameter sets (e.g., ϕ = 0.5, w = 7) showed localized anomalies due to nonlinear geometric effects.

Among the tested algorithms, A* and Greedy BFS achieved the shortest computation times, effectively guided by the Euclidean heuristic, while Dijkstra's algorithm was significantly slower due to its exhaustive, non-heuristic exploration strategy. The results confirm both the critical percolation behavior and the superior computational efficiency of heuristic algorithms in simulating transport phenomena in porous structures.

Note: This article is based on the results of the Filip Klimek Master's thesis, defended at the University of Warmia and Mazury in Olsztyn in 2025.

Author contributions: Filip Klimek: selection, implementation, and description of algorithms; software development; execution of computations; compilation, processing, analysis and interpretation of results. Wojciech Sobieski: formulation of the research topic; literature review; preparation of the introduction and conclusion; description of materials; assistance with interpretation of results; assistance with manuscript editing.

References

CARMAN P.C. 1937. *Fluid flow through a granular bed*. Transactions of the Institute of Chemical Engineers, Jubilee Supplement, 75: 32-48.

FU J., THOMAS H.R., LI C. 2021. *Tortuosity of porous media: Image analysis and physical simulation*. Earth-Science Reviews, 212, 103439: 1-30.

KOPONEN A., KATAJA M., TIMONEN J. 1996. *Tortuous flow in porous media*. Physical Review E, 54: 406-410.

KOPONEN, A., KATAJA, M., TIMONEN, J. 1997. *Permeability and effective porosity of porous media*. Physical Review E, 56, 3319-3325.

KOZENY J. 1927. *Über kapillare Leitung des Wassers im Boden*. Akademie der Wissenschaften in Wien, Sitzungsberichte, 136(2a): 271-306.

MATYKA M., KHALILI A., KOZA Z. 2008. *Tortuosity-porosity relation in porous media flow*. Physical Review E, 78: 026306.

NEWMAN M.E.J., ZIFF R.M. 2000. *Efficient Monte Carlo algorithm and high-precision results for percolation*. Physical Review Letters, 85(19): 4104-4107.

SOBIESKI W. 2019. *Numerical investigations of tortuosity in randomly generated pore structures*. Mathematics and Computers in Simulation, 166: 1-20.

SOBIESKI W. 2020. *Calculating the binary tortuosity in DEM-generated granular beds*. Processes, 8(9): 1-19.

WANG J., ZHOU Z. 2013. *Site percolation thresholds in two dimensions*. Physical Review E, 87(5): 052107.