International Journal of Data Warehousing and Mining
Volume 15 • Issue 2 • April-June 2019

A Clustering Approach to Path Planning for Big Groups

Jakub Szkandera, University of West Bohemia, Pilsen, Czech Republic

Ondřej Kaas, University of West Bohemia, Pilsen, Czech Republic

Ivana Kolingerová, University of West Bohemia, Pilsen, Czech Republic

ABSTRACT

The article introduces a new method of planning paths for big groups in dynamic environments represented by a graph of vertices and edges, where the edge weight as well as the graph topology may change, but the method is also applicable to environments with a different representation. The utilization of clustering enables the use of a computed path for a group of agents. In this way, a speed-up and memory savings are achieved at a cost of some path sub-optimality. Examples of proper applications of the suggested approach are crowd simulation in urban environments or path-planning-based tasks in molecular biology. The experiments showed good behaviour of the method to speed-up, relative error and online computation.

KEYWORDS

Agent Based Model, Clustering, Crowd Simulation, Graph Representation, Molecular Biology Simulation, Path Planning

INTRODUCTION

The path planning or more specifically the task of finding a path between two or more spots in some environment is an important research problem, useful in many different applications, e.g., in robotics, molecular biology or simulations of crowds. The environment can be static or dynamic and is represented by a graph of vertices and edges. In the dynamic environment, the environment properties or its topology may change over time. The static environment does not allow any changes. The paths are planned for entities, which are often called agents.

While the path planning for one agent in the static environment can be considered as a solved problem, the situation is different for more agents or even huge groups in a dynamic environment: a repeated recomputation of path for many changes and many agents may be too slow. For a huge number of agents, the optimality of the paths is less important than the speed of computation as the application for this version of path planning problem usually rather needs the crowd to look well and realistic than to compute and use optimal paths. A similar problem is currently in molecular biology where water molecules have to be navigated through proteins. Therefore, there is a research space for algorithms, sacrificing optimality to speed.

We proposed an algorithm that takes advantage of path similarity of some agents (Szkandera, Kolingerová, & Maňák, 2017). The agents are grouped according to their initial and target position and the path is found only once for the whole group. The method brought a speed-up and memory savings but there was still space for an improvement. Later (Szkandera, Kaas, & Kolingerová, 2017) we proposed to incorporate a more sophisticated approach of groups formation based on clustering.

DOI: 10.4018/IJDWM.2019040103

In this way it is possible to increase the speed-up without fatal consequences on the relative error of the produced paths.

The algorithm proposed here is a generalized version of the clustering-based solution, enabling to include also new-coming agents.

This paper is organized as follows. Section Related Work outlines existing path planning methods, which are suitable for the simulation of big groups, and clustering methods. Section Proposed Solution describes the new algorithm. Section Experiments and Results contains the results of experiments performed on real environment data, e.g. town and molecular environments, and a comparison to classic path planning approach. Section Conclusion concludes the paper.

RELATED WORK

Path Planning

The models for the movement simulation and planning of big groups can be divided into two main categories – the agent-based model and the continuum model.

The more natural for human beings is the first mentioned model where the path is planned for each agent separately. Every agent can have his or her individual requirements and the biggest advantage is that the agent-based model respects them. On the other hand, the path planning for this model may be quite challenging in terms of time and memory complexity. Moreover, with an increasing number of agents this approach stops to be a real time and may become unsuitable.

Agent-based methods can be divided into local and global. The global methods, which help to locate possible evacuation critical spots in buildings and to simulate an emergency scenario, belong to fire-escape algorithms (Okazaki & Matsushita, 1993; Fang, Zong, Li, Li, & Xiong, 2011). Algorithms (Bonabeau, 2002; Singh, Kapadia, Hewlett, Reinman, & Faloutsos, 2011) similar to the fire-escape algorithms have been proposed but with a focus on the bottlenecks. Also, human behaviour have been examined by many interesting approaches (Pellegrini, Ess, Schindler, & Van Gool, 2009; Guy, Kim, Lin, & Manocha, 2011).

The graph representation of the environment is often used for the global agent-based path planning. The most widespread A* algorithm (Hart, Nilsson, & Raphael, 1968), which uses heuristics to speed up the planning, is proposed for a static environment as well as the basic algorithms – Breadth First Search (BFS), Depth First Search (DFS), Dijkstra's algorithm or Floyd-Warshall algorithm (Floyd, 1962; Warshall, 1962), which rank among all-pair algorithms. The minimal cost path for all pairs of vertices is found in the memory complexity $O\left(n^2\right)$ and time complexity $O\left(n^3\right)$ in the worst case for a graph with n vertices. In the case of dynamic graphs, where weights of vertices and edges may change over time, the D* algorithm (Stentz, 1994) and its improvement D* Focused (Stentz et al., 1995) are more suitable. The D* Lite (Koenig & Likhachev, 2002), which modifies the backwards algorithm LPA* (Koenig, Likhachev, & Furcy, 2004), may yield even better results than D* Focused. The memory complexity $O\left(k\left(n+m\right)\right)$ for k agents, n vertices and m edges can be easily reached by these algorithms because each agent needs its own graph ranking. Hierarchical Annotated A* (Harabor & Botea, 2008) creates a hierarchical graph to find an almost optimal path and Anytime D* (Likhachev, Ferguson, Gordon, Stentz, & Thrun, 2005) finds a sub-optimal path in a limited time. The path planning for a moving target solves Moving Target D*-Lite (Sun, Yeoh, & Koenig, 2010) or Generalized Fringe-Retrieving A* (Sun, Yeoh, & Koenig, 2010).

Local methods are less diverse because they are mostly focused on the collision detection between agents themselves or between agents and obstacles. Methods of collision avoidance have been developed including grid-based rules (Loscos, Marchal, & Meyer, 2003) or Bayesian decision processes (Metoyer & Hodgins, 2004). The parallel computation of the collision avoidance has also been proposed (Guy, et al., 2009). The smoothing of the planned path for autonomous vehicles (Chu, Lee, & Sunwoo, 2012) or robots (Vadakkepat, Tan, & Ming-Liang, 2000) belongs into another bigger group of local approaches.

The continuum path planning model which is suitable for dense crowds solves the disadvantages of the agent-based model. The Navier-Stokes equations (Glowinski, Ciarlet, & Lions, 2003) can describe the movement in a dense crowd which is similar to the fluid flow (Darken & Burgess, 2004) as well as the car traffic (Bretti, Natalini, & Piccoli, 2007). A continuum model that introduces an evolving dynamic potential function and describes a crowd as a continuous density field was proposed by Hughes (Hughes, 2002). These partial differential equations describe the continuous density field. This field can be changed into a particle description of the crowd (Treuille, Cooper, & Popović, 2006), which reproduces emerging phenomena of real crowds. The smooth movement of the agents in a complex environment can be achieved with the sophisticated continuum model (Jiang et al., 2010) that was expanded from (Treuille, Cooper, & Popović, 2006). Moreover, the continuum model problems can be solved in parallel (Mao, et al., 2010). Although the continuum model is suitable for the dense crowds, the disadvantage is that the fluid simulation is ruled by the laws of physics. Therefore, individual requirements of crowd members are neglected.

Creating Groups

The clustering algorithms (Jain, 1988; Jain, Murty, & Flynn, 1999) divide similar elements (called clients) into groups (called clusters) represented by one centroid (called a cluster centre or a facility). The nature of the elements are based on the application and in a general case, they could be any objects, which can be described by a characteristic N-dimensional vector. Depending on the area of application, a transformation into the N-dimensional space may be needed to express elements for clustering. For example, for processing a point in a N-dimensional space its N spatial coordinates are used. A similarity of elements is measured by a distance function. The function can be modified based on application requirements and basically might not fulfil the properties of the metrics such as the non-negativity, symmetry or the triangle inequality. However, the Euclidean distance is used most often.

The literature shows several clustering methods such as single-link (Sokal, 1985; Rohlf, 1982), complete-link (King, 1967), sweep-line (Žalik & Žalik, 2009), k-means (MacQueen et al., 1967), parallel c-means (Kwok, Smith, Lozano, & Taniar, 2002) and facility location (Charikar & Guha, 1999).

Now we will define the clustering task more precisely. As we have chosen the facility location algorithm for our purpose, we will present the definition suitable for this type of algorithm.

A given data set of N input data points $x_1, x_2, \ldots, x_N$ is subdivided into k disjoint subsets $F_i, i=1,\ldots,k$ each containing n_i data points, $0<n_i<N$ by minimizing the following mean-square-error (MSE) clustering cost:

$$J_{MSE} = \sum_{i=1}^{k} \sum_{x_j \in F_i} x_j - f_i^2 \tag{1}$$

where x_j is a vector representing the j-th data point in the cluster F_i and f_i is the cluster centre of the cluster F_i. The J_{MSE} function in Equation (1) represents the distance between the data point x_j and the cluster centre f_i.

As we can see, the number of clusters has to be determined before clustering, which is not suitable in all scenarios. Moreover, if k equals to N, J_{MSE} loses its measuring property and the result of clustering is not correct. The facility location algorithm keeps a reasonable amount of clusters by minimization of the following clustering cost:

$$J_{FL} = \sum_{f_i \epsilon F} fc + \sum_{j \epsilon C} c_{f_i j} \tag{2}$$

where $c_{f_i j}$ is the distance between a data point $j \in C$ and its facility $f_i \in F$. The set C contains all data points. To open a new cluster center, a cost fc have to be paid, this way a quantity of cluster centers can be controlled.

The clustering task is an NP-hard problem, so most algorithms produce only approximate results or have some restrictions. On the other hand, in many scenarios the approximate solution suffices.

Clustering methods are used in technical as well as non-technical disciplines, such as data analysis (Dubes & Jain, 1980; Ball & Hall, 1965), data mining (Fayyad, Piatetsky-Shapiro, & Smyth, 1996), pattern recognition (Baraldi & Blonda, 1999) and information retrieval (Rasmussen, 1992).

There are several ways how to categorize the clustering algorithms (Jain, Murty, & Flynn, 1999). One of possible subdivisions is into partitional and hierarchical methods. The result of partitional ones is a flat structure of data distributed into a given number of clusters (partitions) instead of the hierarchical ones which create a tree structure by grouping small clusters into larger ones.

The clustering is called hard if each element is assigned into exactly one cluster. Fuzzy clustering determines for each element a degree of membership in several clusters.

Clustering methods can be further subdivided based on algorithm approaches. The agglomerative approach is buiding the result by merging elements. Therefore, at the beginning each element is a center of a cluster. The divisive approach starts with one cluster containing all elements and performs splitting into smaller clusters. Both agglomerative and divisive algorithms stop when a stopping criterion is satisfied.

To achieve at least some approximation of the best clustering result, the stochastic clustering methods adopt randomized algorithms which are more suitable for processing larger amounts of data due to their smaller time complexity. The deterministic ones usually use some heuristic.

PROPOSED SOLUTION

This section describes the proposed path planning approach for many agents in an environment represented by an undirected graph $G = \left(V, E\right)$, where V is a finite, non-empty set of vertices, and E is a set of unordered pair of vertices $\left(E \subset \binom{V}{2}\right)$ called edges. First, we will summarize the idea of the groups path planning from (Szkandera, Kolingerová, & Maňák, 2017) and then we propose a better and faster solution with the use of clusters.

Let $P = \left\{p_1, p_2, ..., p_c\right\}$ be a set of agents. Each $p_i \in P$ needs to individually rate vertices of an undirected graph. The graph represents the dynamic environment, the dynamics of which can be interpreted as a set of pairs $D = \left\{\left(d_1, t_1\right), \left(d_2, t_2\right), \ldots, \left(d_r, t_r\right)\right\}$, where d_i is a graph change and t_i is a simulation time. Each vertex of the graph describes a point in $\mathbb{R}^w$. Moreover, every agent p_i has a starting vertex with the position $s\left(p_i\right) \in \mathbb{R}^w$ and a goal vertex with $e\left(p_i\right) \in \mathbb{R}^w$.

The two following sections (Summarized group approach and Clustering approach) suppose that all agents are present in the system for the whole time of simulation. Section Incorporating new-coming agents drops this supposition and explains how new-coming agents are processed. This modification allows solving an online version of the given path planning problem.

Summarized Group Approach

Let $g \subseteq P$ be a group of agents. For each group we find a leader $p_m \in g$ who will be followed by others. Figure 1 illustrates the idea. First, any standard path planning algorithm is used to compute the path of the leader from the start $s\left(p_m\right)$ to the end $e\left(p_m\right)$. After that, the path of every agent $p_i \in g \setminus \left\{p_m\right\}$ is planned: First from $s\left(p_i\right)$ to $s\left(p_m\right)$, then the path of the leader p_m is reused, and finally the part from $e\left(p_m\right)$ to $e\left(p_i\right)$ is computed.

In some cases, the path requires further optimization, which will be discussed next, because the path cost is not necessarily optimal. For example, the nearest vertex of the leader's path for p_i may be other vertex than $s\left(p_m\right)$, then it has no sense to go to $s\left(p_m\right)$ and back again. Therefore, the path planning is modified in the following way to handle such situations. When the agent p_i reaches the path of the leader p_m, the path planning to the start position $s\left(p_m\right)$ is stopped and the essential part of the path of the leader p_m is used instead (Figure 2b). Once the agent p_i reaches the destination position $e\left(p_m\right)$, path planning algorithm from the position $e\left(p_m\right)$ to $e\left(p_i\right)$ should be started. However, Figure 2c illustrates that a situation similar to Figure 2a may happen. The change of the path planning direction transforms this problem to the already solved problem (Figure 2b).

The criteria for creating groups of agents with similar start positions and destinations and choosing a leader are as follows. A list P of agents and a grouping threshold τ (will be discussed later) are taken as the input of the group approach and a set $Gr = \left\{g_1, g_2, \ldots, g_q\right\}$ of non-empty distinct groups of agents is produced, where q is the number of groups created implicitly by the algorithm. A path planning strategy is also needed because the computation of the path of an agent from the group g_j depends on the path of its leader. Any classic path planning algorithms, e.g., A*, D* or D* Lite, may be used.

The groups are created as follows: The group algorithm iterates over all agents and tries to add them to all groups created so far. The first agent added to the group is always the leader p_m of the group. For every agent p_i its positions $s\left(p_i\right)$ and $e\left(p_i\right)$ are compared with the start and the destination

Figure 1. The idea of path planning for a group of agents. The path of each agent p_i starts in $s\left(p_i\right)$ and ends in $e\left(p_i\right)$, p_m is the leader. The others can reuse leader's path.

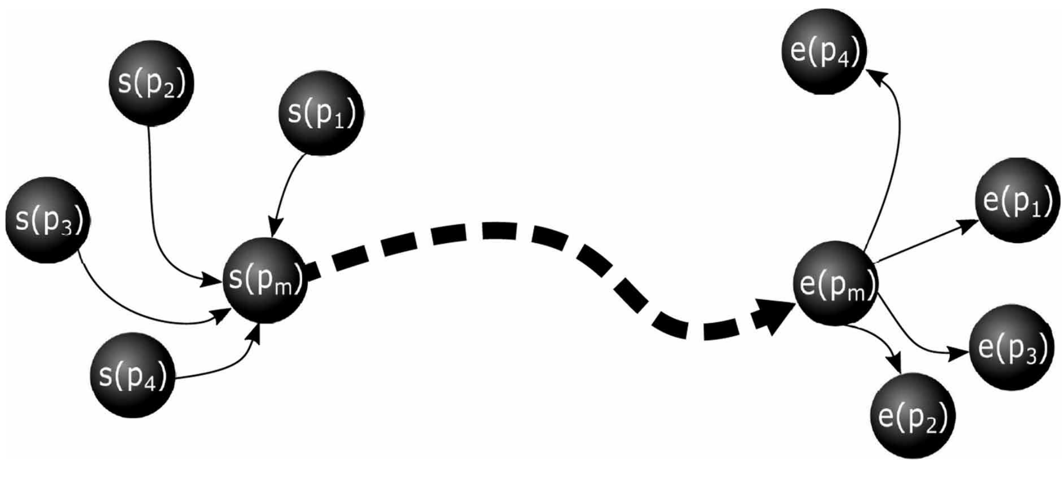
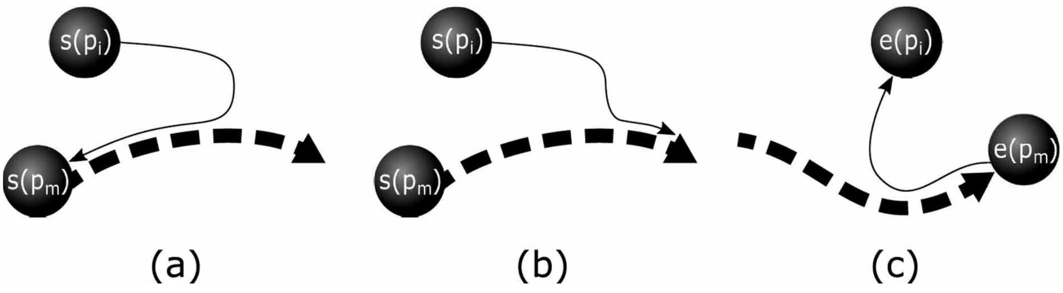


Figure 2. Group path planning problems and solutions: (a) Problem near the start position; (b) Solution for the case (a); (c) Problem near the destination



of the leader $p_m \in g_j$. The agent p_i is added to the group g_j if both distances are less than the error threshold τ. Otherwise the agent p_i is set as the leader of a new group and the group is added to the set Gr.

The worst-case time complexity is $O\left(\left|P\right|\left|Gr\right|\right)$ or $O\left(\left|P\right|^2\right)$ if each group contains only one agent. The performance of the algorithm and the quality of results depend on the threshold τ. The threshold τ is a boundary of the maximal allowed relative error. Each member of the group (except the leader) may have its path 1.1 times longer than the path of the leader when $\tau = 10\%$ is chosen. The higher τ the faster the path computation is. On the other hand with the growing threshold τ the path inaccuracy grows.

Clustering Approach

The output groups of the group approach are sensitive to the order of the input data because the algorithm has a greedy character - the first possible solution is accepted and never reconsidered. An additional optimization of the found groups would improve the final paths but deteriorate the complexity which is already $O\left(n^2\right)$. What is more, although the group approach speeds up the path computation, there is still room for acceleration. Therefore, we incorporate to the group creation a clustering by a non-modified local search algorithm (Skála & Kolingerová, 2010; Okazaki & Matsushita, 1993) with relevant parameters setup discussed in Section *Experiments and results*. The algorithm implicitly optimizes the agent groups and their found paths. Moreover, the clustering even speeds up the computation because the clustering can be done in $O\left(n \log n\right)$.

For clustering purposes the agent p in the input set $P = \left\{p_1, p_2, \dots, p_N\right\}$, where N is a number of agents, is described as the following $2w$-dimensional vector:

$$v_i = \left(s\left(p_i\right), e\left(p_i\right)\right)^T, v_i \in \mathbb{R}^{2w} \tag{3}$$

where $s\left(p_i\right) \in \mathbb{R}^w$ is the starting and $e\left(p_i\right) \in \mathbb{R}^w$ the destination position of the agent.

The main idea of the proposed improvement is to incorporate Equation (2) as a heuristic to avoid checking so many possible cases compared to the original group approach at a price of a lowered accuracy of the clustering result.

Let us describe the used *local search* clustering algorithm steps in detail. Algorithm starts by creating an initial coarse solution. A cluster center is always created at the first point and further points are then taken in the random order. A point v_i is connected to the closest already open cluster center based on the measured distance d between the point v_j and the open cluster center. With the probability d / fc (or one if $d > fc$) a new cluster center is opened at the point v_j. This initial coarse solution is improved by $O\left(n \log n\right)$ iterations of the following local search step. The explanation for $O\left(n \log n\right)$ steps of iterations is given in (Skála & Kolingerová, 2011).

A single local search step can be described as a random selection of a point $v_i \in C \cup F$ (it does not matter whether it is a cluster center or not) and it is computed whether the point v_i can improve the current solution (if v_i is not already a cluster center, the facility cost would have to be paid for its opening). Some clients (points) may be closer to the investigated (new) cluster center f_{v_i} than to their current facility. All such clients can be re-assigned to f_{v_i}, it decreases the connection cost. If these changes result in some cluster center having only a few clients remaining, the cluster center could be closed and its facility cost spared.

A possible improvement of the current solution by declaring the point v_i a new cluster center f_{v_i} and reassigning all near clients from their cluster centers to f_{v_i} is determined by a gain function according to the following relation:

$$gain\left(v_i\right) = -fc + \sum_{c_i \epsilon C} ds_i + \sum_{f_j \subseteq F} cs_j \tag{4}$$

where fc is the facility cost, or zero if the cluster center f_{v_i} is already open, ds_i is the distance spared by reassigning the client c_i from its current cluster center to the cluster center candidate f_{v_i} and cs_j (close spare) is the facility cost minus expenses for reassigning all remaining clients from their current cluster center f_j to f_{v_i}. If the current cluster center f_j lies closer to v_i than f_{v_i} then $ds_i < 0$ and ds_i needs to be set to 0. Again, if $cs_j < 0$ (cluster center f_j has enough clients, so no spare can be achieved by closing the cluster center and reassigning all their clients to the new cluster center f_{v_i}) then cs_j is set to 0. If $gain\left(v_i\right) > 0$, opening the cluster center at the point v_i improves the current solution and therefore reassignments and closures are saved.

The algorithm of the group approach with the facility location clustering is summed up as Algorithm 1. When the clusters, which represent the groups of agents, are computed (Alg. 1, line 1) from the input set of agents P, the path of each agent is planned by any standard path planning strategy, e.g., Dijkstra, A* or D*. First the path of the leader p_m of the group g_i is found (Alg. 1, line 4). Then two paths are computed each member of the group g_i - first between vertices $s\left(p_j\right)$ and $s\left(p_m\right)$ and second between $e\left(p_j\right)$. and $e\left(p_m\right)$ (Alg. 1, lines 6-7). Finally, these paths and the path of the leader are joined (Alg. 1, line 8). This process goes in cycle for each group $g_i \in Gr$. The procedure *cluster_agents* generates first a coarse solution of agent groups, then it reatedly tries to improve it by reassignment of a randomly chosen agent to another group. If the new connection improves the overall clustering cost J_{FL}, then it is preserved and another randomly chosen agent is investigated. The improvement phase is repeated $O\left(n \log n\right)$ times.

Algorithm 1. The clustering path planning approach

```
Data: A list $P$ of agents, a path planning strategy find_path(...)
Result: The list $P$ of agents with computed paths
Algorithm cluster_approach
1 $Gr$ ←cluster_agents($P$);   // groups of agents
2 foreach $g_i \in Gr$ do
3 $p_m$ ← the first member of $g_j$; // the leader of the group
4 $p_m$.path ← find_path($s(p_m)$, $e(p_m)$);
5 foreach $p_j \in \{g_i \; \{p_m\}\}$ do
6 path_start ← find_path($s(p_i)$, $s(p_m)$);
7 path_end ← find_path($e(p_i)$, $e(p_m)$);
8 $p_j$.path ← Join paths(path_start, $p_m$.path, path_end)
end
end
return $P$;
Procedure cluster_agents(the list of agents $P$)
1 Generate a coarse solution of groups
2 repeat $O(n \log n)$ times
3 Pick $v_i \in P$ at random;
4 if $gain(v_i) > 0$ then
5 Perform reassignments and closures.
end
return groups;
```

Incorporating New-Coming Agents

Let us drop the condition that all agents are present in the system for the whole simulation time. A new agent has basically two options. It can either become part of the nearest existing group from those having a similar goal, or it can create a totally new group. Every clustering solution and calculated paths stay the same until the end of the simulation, so some calculation for new agents can be spared.

The whole modification for dynamic agent clustering is carried out in the procedure *cluster_agents*. A coarse solution is generated at the beginning in the same way as in Algorithm 1. This step assigns new agents to previously created groups that have the same goal of journey, or creates new groups with different goals. Then the vertex v_i is randomly chosen from the new block of agents B_j and the function $gain$ calculates the possible improvement. The function now does not consider the option to close groups which were created during processing the block B_{j-1}. This step is followed by reassignments within all agents and closes only newly created groups from the block of agents B_j.

Modification for dynamic clustering is summarized in the procedure *cluster_agents* in Algorithm 2.

Algorithm 2. The dynamic clustering approach

```
Data: Block of agents $B_j$
Result: The list $Gr$ of groups of agents
Procedure cluster_agents(block of agents $B_j$)
```

```
1 Generate a coarse solution of groups
2 repeat $O(n \log n)$ times
3 Pick $v_i \in P$ at random;
4 if $gain(v_i) > 0$ then
5 Perform reassignments in all existing groups $Gr$
6 Perform closures only in the set $B_j$
end
return groups;
```

EXPERIMENTS AND RESULTS

The proposed method was implemented in C# and all experiments were performed on a computer with the CPU Intel Core i7-950 (8MB Cache, 3.07GHz) and 12GB 668MHz RAM. The proposed solution was tested on two types of the environments – a city and a protein. Moreover, each of the environments was tested with two types of agent data.

We were unable to determine the time and space complexity generally because they highly depend on the distribution of the agents and the chosen path planning algorithm. Therefore, the experiments focus on the computational time and especially on the path correctness of the proposed solution to at least show the behavior of the proposed approach in detail. For instance, the A* algorithm as the path planning method will not save the computational space because it does not need to store a unique graph rating for each agent. However, dynamic algorithms such as D* Lite algorithm, as mentioned in Section Related Work, are able to save up to $O(kn)$ space for k similar paths and n vertices.

Crowd Simulation

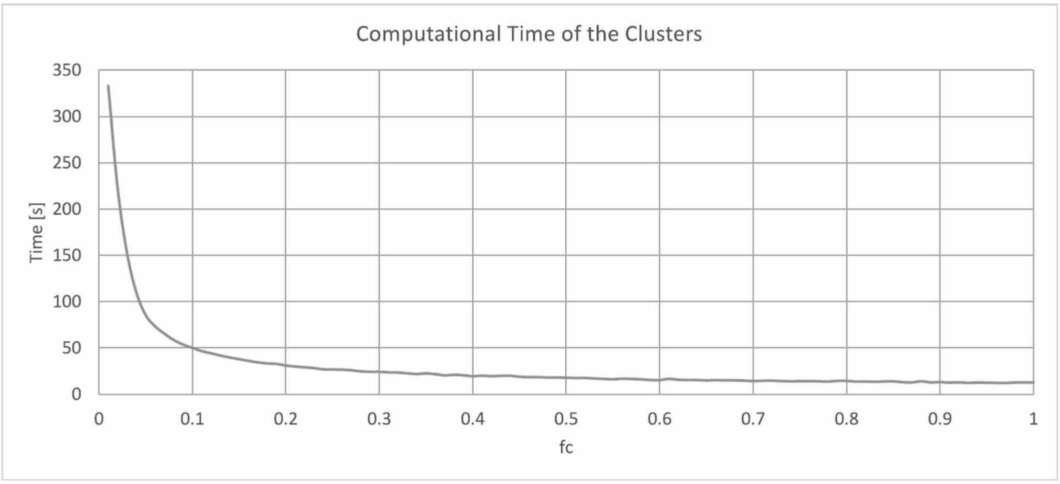
The proposed solution was tested on two types of the testing data. First type ('unsuitable data') contains agents generated at random positions and the second type ('suitable data') contains groups of agents (many agents with similar paths). The environment is represented by real data – the 2D Open Street Map of Pilsen, Czech Republic.

The Computational Time of the clustering method without the path planning is shown in Figure 3, where the time dependences on the facility cost fc are depicted. The higher value of fc produces bigger clusters and speeds up the computation of the clusters. The reasonable facility cost is for $fc > 0.1$, where the computational time becomes relatively very low (Figure 3) and with the growing fc the time still descends.

Figure 4 shows the computational time of the standard A* algorithm, the A* with the group approach and with the proposed clustering approach (Alg. 1). The experiments were made for $100k$ randomly generated agents and the measured computational time of both approaches includes the groups creation and path planning strategy. The group approach depicted in Figure 4 uses $\tau = 10\%$. and the clustering approach is showed with the facility cost $fc = 0.5$. The proposed clustering approach is the fastest. The random character of the clustering algorithm causes small fluctuations on the time curve. The curve of the clustering approach is $O(n \log n)$, the group approach curve is $O(n^2)$ and the A* curve is $O(n)$. Although the A* is the slowe in the graph, obviously it thanks to its better algorithmic complexity for some number of agents overruns both the group and the clustering approaches. For the given size of the environment it will be $\approx 300k$ agents for the group approach and over millions of agents for the clustering approach. This number is relatively high and so it is not such a big problem as it might seem.

However, the data of randomly generated agents are the worst possible for the proposed approach. The clustering approach is most suitable for the groups of agents where the clustering approach is

Figure 3. Computational time of the clustering method for 100k agents



a clear choice because it is able to reuse many paths and save a huge amount of the computational time (Figure 5). The same computational time of the A* algorithm and the clustering approach with A* algorithm is for billions of agents for the data of the groups of agents.

The Path Correctness of the found path shows how large is the difference between the minimal path and the path found by the proposed solution. The path correctness is measured by a relative error (5):

$$\delta = \frac{length\left(group.path\right) - length\left(minimal.path\right)}{length\left(minimal.path\right)} \tag{5}$$

where $group.path$ is the path found with the group or the clustering approach including existing path planning method and $minimal.path$ is the path found with the same path planning algorithm without the proposed solution. Note that $\delta \geq 0$.

The average relative error dependency on the chosen facility cost is shown in Figure 6 for two above mentioned types of the testing data. The upper curve, which reaches the relative error up to 15%, represents $100k$ randomly generated agents position and the lower curve represents data with generated groups of agents (a lot of similar paths). The average relative error of the groups does not exceed 2%. This is an expected result because this type of data is most suitable for the proposed clustering approach. Moreover, the result of the worst case (randomly positioned agents) is acceptable and for the facility cost up to the value 0.1 is comparatively very good. It can be also seen that the relative error does not change too much if $fc > 0.5$. It means that if a higher relative error is acceptable, number of clusters can be kept relatively small. The parameter fc can be understood as percentage expression number of clusters, where approximately $100\left(1 - fc\right)$ per cent of agents from the input data become cluster centers, see Section *Clustering approach*.

Figure 7 shows the dependency of the relative error on the relative number of the created groups and the difference between the original group approach (without clustering) and the group approach with clustering (Alg. 1). The x-axis describes the relative number of groups $\lambda = \frac{\left|Gr\right|}{\left|P\right|}$, where $\left|Gr\right|$ is the number of created groups and $\left|P\right|$ the number of tested agents. The experiments have been

Figure 4. Computational time of single A*, A* with group and clustering approaches for the data of randomly generated agents

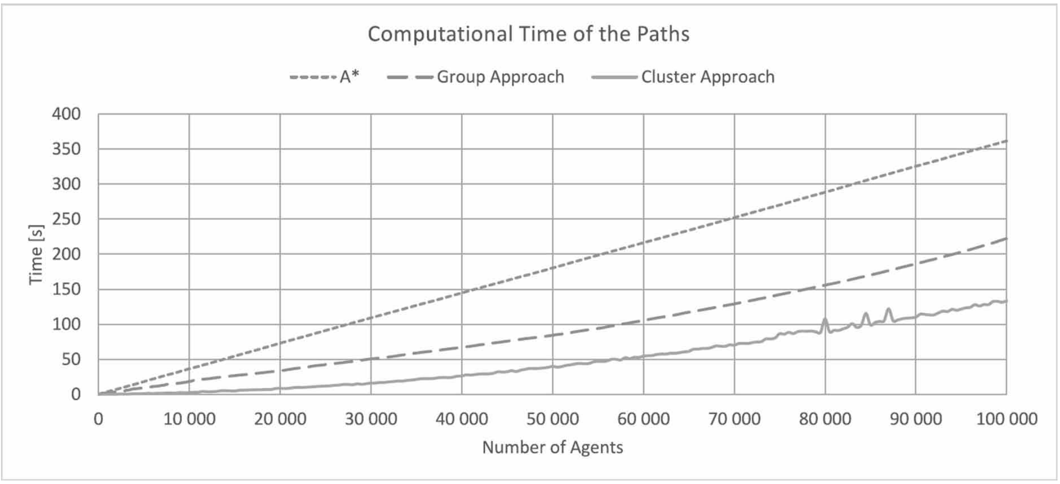
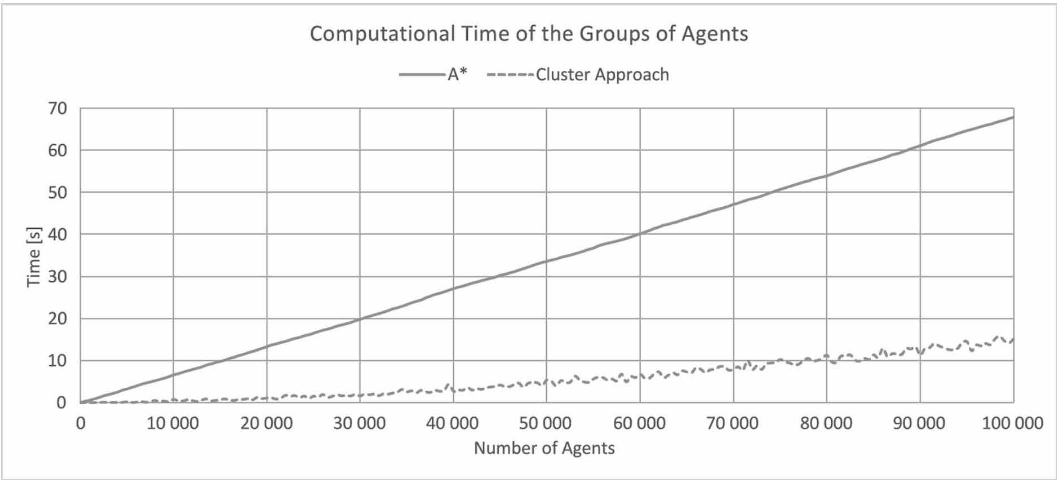


Figure 5. Computational time of single A* and A* with the clustering approach for the data of the groups of agents



performed on the randomly generated data of $100k$ agents. The higher the number of groups is, the fewer agents each group contains. For example, when the proposed solution creates $100k$ groups (clusters) from $100k$ agents, the average relative error will be zero because every agent is a leader of a group. The other extreme is only one group (a cluster) created from 100k agents. In that case, relative error will be enormous because everyone has to follow the leader. The average relative error of both approaches grows with the decreasing number of groups, as can be expected. Although both approaches are almost the same for a high number of groups, the clustering approach gives better results for a smaller number of groups, which contain a higher number of agents. The average relative error of the clustering approach oscillates for the small number of groups because it contains randomized operations which are more visible for greater groups.

In the graphs, attention was drawn to the average relative error. The maximal relative error is influenced by the setting of the parameters - how big attraction of the group leaders for other agents

Figure 6. The average relative error δ for $100k$ generated agents, either random or in groups, in dependence on facility cost fc

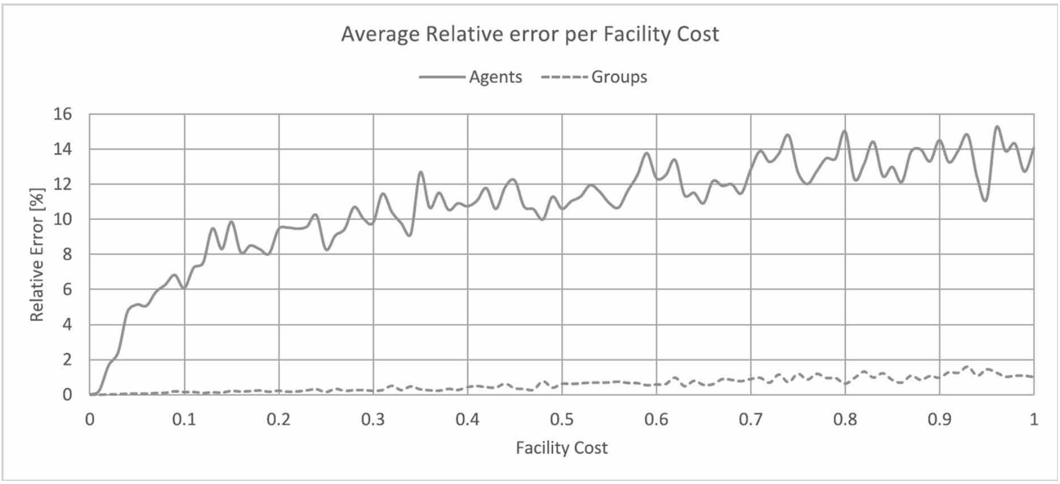
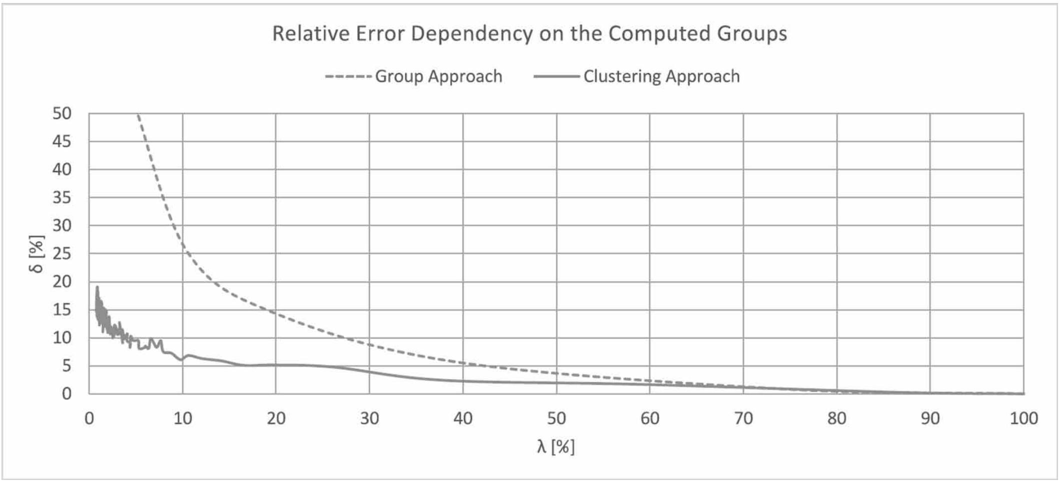


Figure 7. The average relative error δ of the group and clustering approaches in dependence on the relative number of groups λ



is allowed. If big then even more distant agents are pushed to group with a leader which may result in a high suboptimality of the path for the given agent. Such a path then has a high relative error and so contributes to a high maximal error. Fortunately, maximal errors concern individuals while the approach is aimed at big groups, so one strongly non-optimal path is not important in the intended applications and will be amortized by good behavior of the whole group.

The Online Computation was tested on the same kind of the data because we were simulating a new scenario where new agents may appear during simulation. At this situation, the proposed approach does not have information of all agents at the start of the simulation t_0 because new agents may appear during simulation time $t_i, i \in \mathbb{N}$ where $t_i > t_0$. Therefore, the clustering and the path planning of the agents cannot be computed at the same simulation time t_0 as in the previous experiments. The experiments were performed for $100k$ agents. First x agents are processed (clustering and path planning) and then every processed agent moves along his found path by one edge. After that the next x agents are processed and $2x$ agents move. This is repeated up to $100k$ agents and the x is called a block of agents. The tested size of the blocks was $1k$, $2k$, $4k$, $5k$ and $10k$ of agents.

Figure 8 shows the online computation of the average relative error dependency on the chosen facility cost for the randomly generated positions of agents. The experiment shows that with the growing size of the blocks of agents the relative error decreases. The reason is that the clustering algorithm has a bigger pool of the independent agent data and has a higher chance to create more precise agent groups.

On the other hand, the depicted result in Figure 9 for the randomly generated groups of agents is opposite in comparison with Figure 8. In contrast with the previous result, where the agents were independent, there is dependency between agents because they are generated in groups. Therefore, the clustering algorithm with a growing size of the blocks of agents (input data) has a higher chance to break the implicit group dependency. Although the result for the randomly generated groups of agents is exactly opposite than the result of the randomly generated agent positions, from an overall point of view the groups of agents have many times more precise paths. The relative error is up to 9% while the relative error of the randomly generated agents grows up to 80%.

Molecular Biology Simulation

The 3D molecular biology environment is represented by real protein data – $1cqw.pdb$ from the RCSB Protein Data Bank. The proposed solution was tested on the graph representation created from the protein data. Similar to the crowd simulation, we tested two types of input data – the randomly generated agents (spherical probes or ligands) positions and the randomly generated positions of groups of agents.

The Computational Time of the standard A* algorithm and the A* with the proposed clustering approach (Alg. 1) is shown in Figure 10. Similar to the crowd simulation the experiments were made for $100k$ randomly generated agents. The clustering approach is showed with the facility cost $fc = 0.3$ and it is still faster than the standard A* algorithm.

As we mentioned in Section *Crowd Simulation*, the clustering approach is most suitable for the groups of agents where the clustering approach is able to save a huge amount of the computational

Figure 8. The average relative error δ of the online computation for 100k randomly generated agents in dependence on facility cost fc

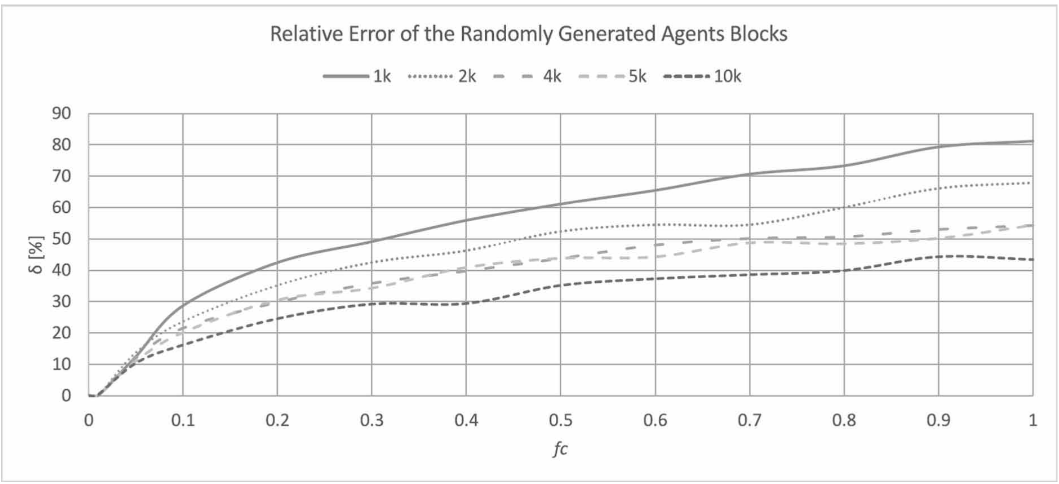
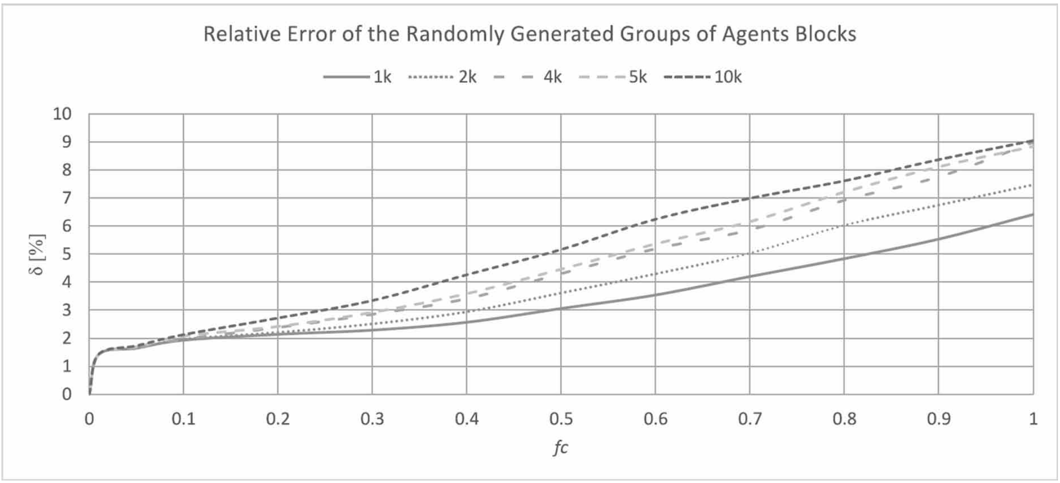


Figure 9. The average relative error δ of the online computation for $100k$ generated agents in groups in dependence on facility cost fc



time (Figure 5). Obviously, there is a huge speed-up when we compare results of the single A* and A* with clustering approach in Figure 10 and Figure 11.

The Path Correctness is measured by a relative error given by Equation (5) described in Section *Crowd Simulation.* Figure 12 shows the average relative error dependency on the chosen facility cost for the two above mentioned types of the testing data. The upper curve represents 100k randomly generated agents positions with the relative error up to 43% and the lower curve represents data with generated groups of agents. The average relative error of the groups does not exceed 20%. This result is slightly worse than the path correctness of the crowd simulation. The reason is that we were searching the path for the worst input agent data – the agents with zero radii. When the agents have very small radii, the graph G is dense, there is a huge path diversity, and the relative error accumulates. With the growing agent radius, the relative error decreases because the graph edges start to be gradally impassable.

We would like to remind that the parameter fc can be understood as a cost for opening a new cluster. The smaller parameter fc is, the more expensive the opening is. Therefore, the choice of a smaller and more convenient parameter fc, e.g. $fc = 0.3$ or less, also helps to reduce the relative error.

The Online Computation was tested the same way as in Section Crowd Simulation. First the proposed approach processes x agents and after that every processed agent moves by one edge on the found path. This is repeated up to $100k$ agents.

The online computation of the average relative error dependency on the chosen facility cost for the randomly generated positions of agents is shown in Figure 13. The behavior of the proposed approach for the molecular data is obviously similar to the behavior of the crowd simulation data. The bigger blocks of agents produce more paths that are accurate because the proposed clustering approach has higher probability to optimize the generally independent data.

Figure 14 shows the results of the randomly generated groups of agents. Results in the crowd simulation are slightly different from the molecular simulation results but the overall trend is the same. The difference is that the relative error is almost negligible for the facility cost fc range from 0 to 0.5. On the other hand, the proposed approach still has a greater probability of destroying data dependence for larger blocks. Therefore, the relative error is the smallest for the smallest blocks of agents.

Figure 10. Computational time of single A* and A* with clustering approach (fc = 0.3) for the data of the randomly generated agents

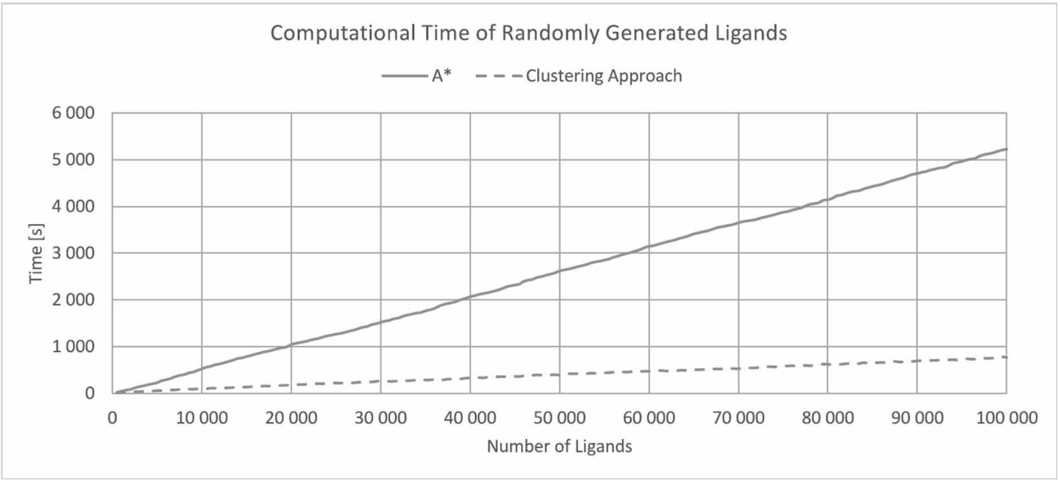
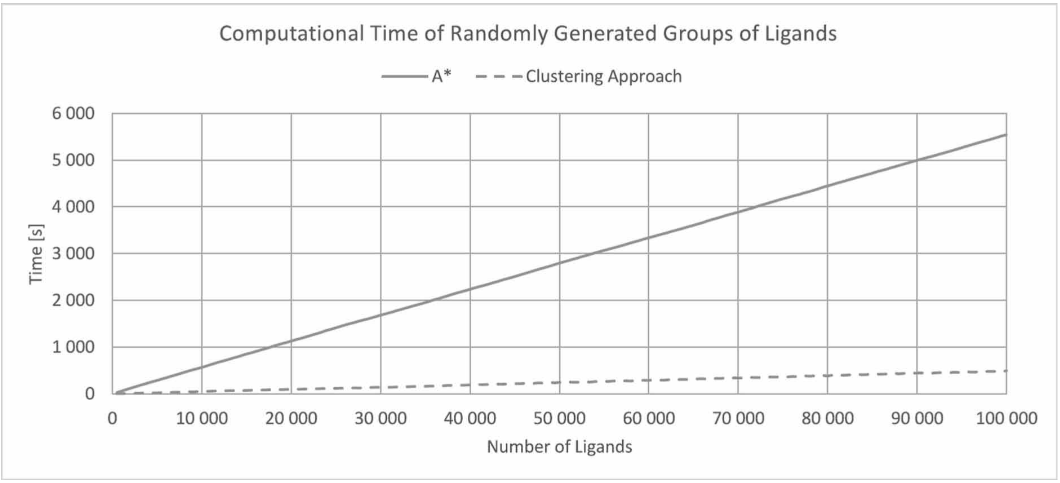


Figure 11. Computational time of single A* and A* with clustering approach (fc = 0.3) for the data of the groups of agents



CONCLUSION

In this paper, we introduced a path planning approach for many agents in an environment represented by a graph of vertices and edges. The approach can be also used for other types of environment, e.g., grids or polygonal meshes. Our approach creates groups of agents based on their start and target positions and optimizes the size of the created groups by clustering. Any standard graph-based path planning algorithm, e.g., A*, D* or D* Lite is then used to plan the paths. The substantial part of the computed path of group leaders is shared with the rest of the group. In this way, computation time and memory can be saved at a price of non-optimality of paths. The independence of the proposed approach of the dimension and the data types, e.g. data of cities or proteins, has been confirmed by extensive testing. The proposed approach was tested with the use of the A* algorithm but it could be easily extended to employ other path planning methods. The running time and the influence of the clustering on the quality of the paths have been measured. The proposed approach proved to be suitable

Figure 12. The average relative error δ for $100k$ generated agents, either random or in groups, in dependence on facility cost fc

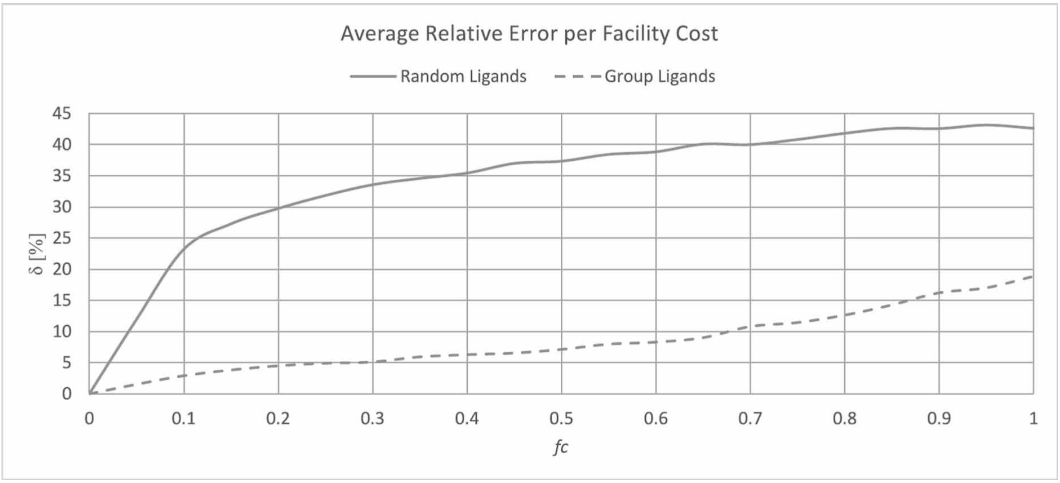
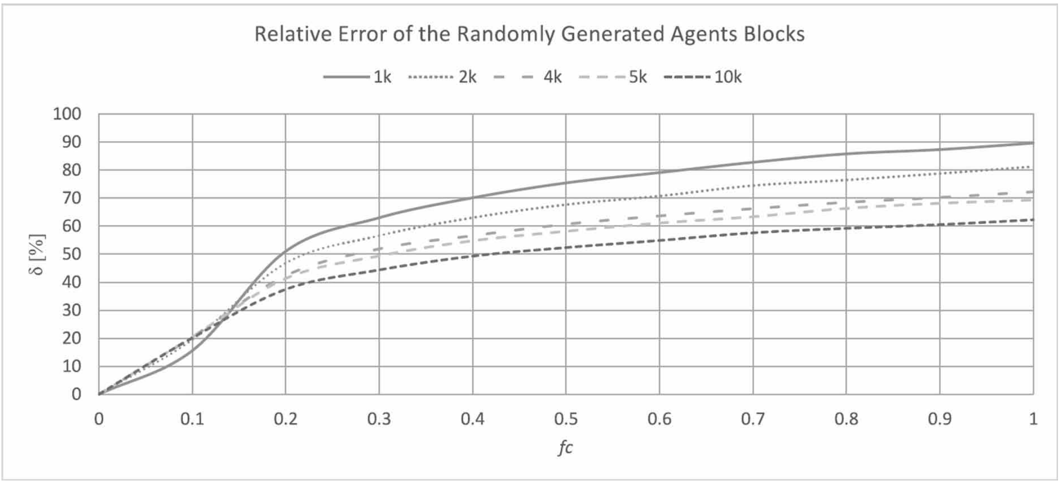


Figure 13. The average relative error δ of the online computation for $100k$ randomly generated agents in dependence on facility cost fc

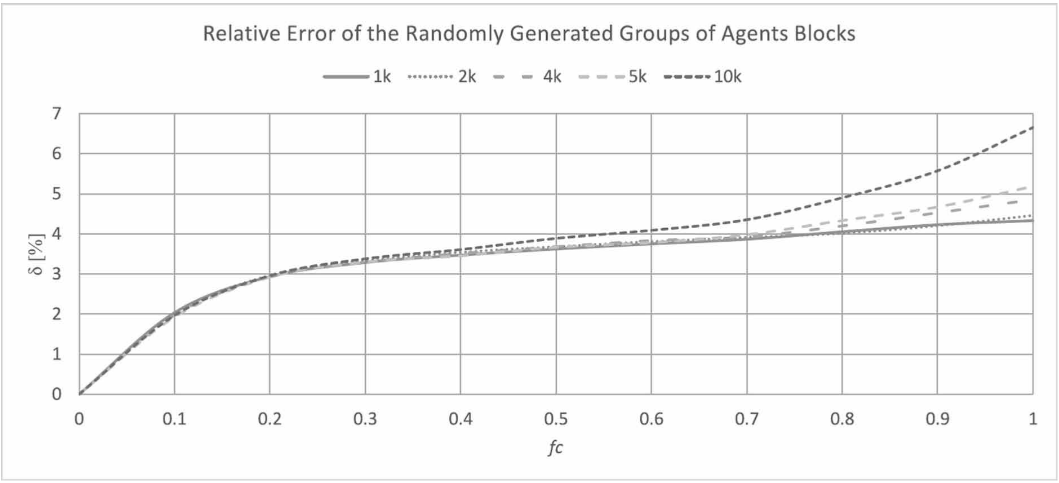


for the data with a significant number of potentially similar paths. Moreover, the experiments proved that it is possible to use the online computation with a relatively small inaccuracy for this type of data.

ACKNOWLEDGMENT

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, project SGS-2016-013 Advanced Graphical and Computing Systems, and by the Czech Science Foundation, project No. 17-07690S.

Figure 14. The average relative error δ of the online computation for $100k$ generated agents in groups in dependence on facility cost fc



REFERENCES

Ball, G. H., & Hall, D. J. (1965). *ISODATA, a novel method of data analysis and pattern classification.*

Baraldi, A., & Blonda, P. (1999). A survey of fuzzy clustering algorithms for pattern recognition. I. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, *29*(6), 778–785. doi:10.1109/3477.809032 PMID:18252357

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(Suppl. 3), 7280–7287. doi:10.1073/pnas.082080899 PMID:12011407

Bretti, G., Natalini, R., & Piccoli, B. (2007). A fluid-dynamic traffic model on road networks. *Archives of Computational Methods in Engineering*, *14*(2), 139–172. doi:10.1007/s11831-007-9004-8

Charikar, M., & Guha, S. (1999). Improved combinatorial algorithms for the facility location and k-median problems. In *40th Annual Symposium on Foundations of Computer Science* (pp. 378-388). IEEE.

Chu, K., Lee, M., & Sunwoo, M. (2012). Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Transactions on Intelligent Transportation Systems*, *13*(4), 1599–1616. doi:10.1109/TITS.2012.2198214

Darken, C. J., & Burgess, R. G. (2004). Realistic Human Path Planning using Fluid Simulation. Army War College Carlisle Barracks, PA.

Dubes, R., & Jain, A. K. (1980). Clustering methodologies in exploratory data analysis. [). Elsevier.]. *Advances in Computers*, *19*, 113–228. doi:10.1016/S0065-2458(08)60034-0

Fang, Z., Zong, X., Li, Q., Li, Q., & Xiong, S. (2011). Hierarchical multi-objective evacuation routing in stadium using ant colony optimization approach. *Journal of Transport Geography*, *19*(3), 443–451. doi:10.1016/j.jtrangeo.2010.10.001

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, *17*, 37.

Floyd, R. W. (1962). Algorithm 97: Shortest Path. *Communications of the ACM*, *5*(6), 345. doi:10.1145/367766.368168

Glowinski, R., Ciarlet, P. G., & Lions, J. L. (2003). Handbook of numerical analysis: Numerical methods for fluids.

Guy, S. J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., & Dubey, P. (2009). Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (pp. 177-187). doi:10.1145/1599470.1599494

Guy, S. J., Kim, S., Lin, M. C., & Manocha, D. (2011). Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 43-52). doi:10.1145/2019406.2019413

Harabor, D., & Botea, A. (2008). Hierarchical path planning for multi-size agents in heterogeneous environments. In *2008 IEEE Symposium On Computational Intelligence and Games* (pp. 258-265). doi:10.1109/CIG.2008.5035648

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics, 4*, 100-107.

Hughes, R. L. (2002). A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, *36*(6), 507–535. doi:10.1016/S0191-2615(01)00015-7

Jain, A. K. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys*, *31*(3), 264–323. doi:10.1145/331499.331504

Jiang, H., Xu, W., Mao, T., Li, C., Xia, S., & Wang, Z. (2010). Continuum crowd simulation in complex environments. *Computers \& Graphics, 34*, 537-544.

King, B. (1967). Step-wise clustering procedures. *Journal of the American Statistical Association*, *62*(317), 86–101. doi:10.1080/01621459.1967.10482890

Koenig, S., & Likhachev, M. (2002). D* Lite. In AAAI/IAAI (pp. 476-483).

Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence*, *155*(1-2), 93–146. doi:10.1016/j.artint.2003.12.001

Kwok, T., Smith, K., Lozano, S., & Taniar, D. (2002). Parallel fuzzy c-means clustering for large data sets. In Euro-Par 2002 parallel processing (pp. 27-58).

Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., & Thrun, S. (2005, June). Anytime Dynamic A*: An Anytime, Replanning Algorithm. In ICAPS (pp. 262-271).

Loscos, C., Marchal, D., & Meyer, A. (2003). *Intuitive crowd behavior in dense urban environments using local laws. In Theory and Practice of Computer Graphics* (pp. 122–129).

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281-297).

Mao, T., Jiang, H., Li, J., Zhang, Y., Xia, S., & Wang, Z. (2010). Parallelizing continuum crowds. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology* (pp. 231-234). doi:10.1145/1889863.1889914

Metoyer, R. A., & Hodgins, J. K. (2004). Reactive pedestrian path following from examples. *The Visual Computer*, *20*(10), 635–649. doi:10.1007/s00371-004-0265-z

Okazaki, S., & Matsushita, S. (1993). A study of simulation model for pedestrian movement with evacuation and queuing. In *International Conference on Engineering for Crowd Safety*.

Pellegrini, S., Ess, A., Schindler, K., & Van Gool, L. (2009). YouÍl never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, (pp. 261-268).

Rasmussen, E. M. (1992). Clustering Algorithms. *Information retrieval: data structures & algorithms, 419*, 442.

Rohlf, F. J. (1982). 12 Single-link clustering algorithms. In *Classification Pattern Recognition and Reduction of Dimensionality* (Vol. 2, pp. 267–284). Elsevier. doi:10.1016/S0169-7161(82)02015-X

Singh, S., Kapadia, M., Hewlett, B., Reinman, G., & Faloutsos, P. (2011). A modular framework for adaptive agent-based steering. In *Symposium on Interactive 3D Graphics and Games*. doi:10.1145/1944745.1944769

Skála, J., & Kolingerová, I. (2010). Accelerating the local search algorithm for the facility location. In *Proceedings of the 12th WSEAS international conference on Mathematical and computational methods in science and engineering* (pp. 98-103).

Skála, J., & Kolingerová, I. (2011). Faster facility location and hierarchical clustering. *International Journal of Computers*, *5*, 132–139.

Sokal, R. R. (1985). The principles of numerical taxonomy: twenty-five years later. *Computer-assisted bacterial systematics*, *15*, 1-20.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. *Proceedings 1994 IEEE International Conference on Robotics and Automation* (pp. 3310-3317).

Stentz, A. et al. (1995). The focussed D* algorithm for real-time replanning. *IJCAI (United States)*, *95*, 1652–1659.

Sun, X., Yeoh, W., & Koenig, S. (2010). Generalized Fringe-Retrieving A*: faster moving target search on state lattices. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems* (Vol. 1, pp. 1081-1088).

Sun, X., Yeoh, W., & Koenig, S. (2010). Moving target D* lite. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems* (Vol. 1, pp. 67-74).

Szkandera, J., Kaas, O., & Kolingerová, I. (2017). A Clustering Approach to Path Planning for Groups. In *International Conference on Computational Science and Its Applications* (pp. 465-479). Springer. doi:10.1007/978-3-319-62395-5_32

Szkandera, J., Kolingerová, I., & Maňák, M. (2017). Path Planning for Groups on Graphs. *Procedia Computer Science*, *108*, 2338–2342. doi:10.1016/j.procs.2017.05.040

Treuille, A., Cooper, S., & Popović, Z. (2006). Continuum crowds. *ACM Transactions on Graphics*, *25*(3), 1160–1168. doi:10.1145/1141911.1142008

Vadakkepat, P., Tan, K. C., & Ming-Liang, W. (2000). Evolutionary artificial potential fields and their application in real time robot path planning. In *Proceedings of the 2000 Congress on* Evolutionary Computation (Vol. 1, pp. 256-263). doi:10.1109/CEC.2000.870304

Warshall, S. (1962). A Theorem on Boolean Matrices. *Journal of the Association for Computing Machinery*, *9*(1), 11–12. doi:10.1145/321105.321107

Žalik, K. R., & Žalik, B. (2009). A sweep-line algorithm for spatial clustering. *Advances in Engineering Software*, *40*(6), 445–451. doi:10.1016/j.advengsoft.2008.06.003