New path planning method for computation of constrained dynamic channels in proteins

Petr Beneš
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
xbenes2@fi.muni.cz

Ondřej Strnad
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
xstrnad2@fi.muni.cz

Jiří Sochor
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
sochor@fi.muni.cz

ABSTRACT

Collision-free paths in the geometric model of a protein molecule reveal various dependencies between the structure of the molecule and its function. The paths which connect a biochemically important part of the protein molecule with the surface of the molecule can serve as egression or access paths for small molecules which react in the active site. The geometric method introduced in this paper is designed to compute such paths in the dynamic models of protein molecules. The paths have to satisfy additional constraints such as valid flow of time which allows us to distinguish between access and egression paths, minimum width and others. Possibly, the method may be used not only for protein molecules but also for similar environments with high density of spherical obstacles. The method was tested on real protein data and the results indicate that if there is a path present, it is detected by our method.

Keywords: protein, path planning, collision-free path, constrained dynamic channel

1 INTRODUCTION

Investigating the behaviour of protein molecules is one of the main tasks that help biochemists fully understand how the real micro-world works. One of the real applications is the development of new and the improvement of existing drugs. These days, we can utilize the computational power and perform simulations instead of wasting time in the laboratory.

With the increase in computational power the molecular dynamics simulation which simulates the behaviour of the protein molecule is capable of capturing longer time intervals. The resulting vast amount of data has to be processed and for chemists, it is desired to reveal the dependency between structure and function of a protein molecule. One of the crucial analyses is to find the collision-free paths connecting the active site of the protein molecule with its surface. These paths are called channels and may be used by a small molecule as access paths to or egression paths from the active site. Chemists need to focus on these paths to assess whether the active site is well or poorly accessible.

In the protein molecule, each atom is represented geometrically as a sphere on given three-dimensional coordinates with appropriate Van der Waals radius. To reflect the real situation, we have to consider the movement of atoms in the molecule. This movement certainly influences channels and their properties. The movement is typically obtained by a molecular dynamics simulation and is stored as a set of snapshots – states of the molecule over time. The simulation may generate thousands of snapshots. The set of snapshots is denoted by chemists as a trajectory.

Previous approaches to channel computation were able to compute channels in a single snapshot only. Even if the approach was applied to each snapshot, the solution did not exactly reflect the real situation in which the small molecule (substrate or product) passes into or leaves the active site. The ongoing research demanded that collision-free paths which satisfy certain constraints and connect the active site with the surface of the molecule are determined. Although other possibilities exist, in this paper, the surface of the molecule is understood as the convex hull boundary.

In this paper, we introduce new method, which joins the information about an empty space in different snapshots into one large graph and finds collision-free paths connecting the active site with the surface of the molecule. These paths are called dynamic channels. Applying additional constraints on these paths allows us for instance to consider the flow of time and distinguish between access and egression paths. These paths are referred to as constrained dynamic channels.

The method we propose is based on computational geometry principles and could be used to find collision-free paths for a spherical robot in an environment with high density of moving obstacles. We expect these obstacles to be spherical too. Our specific implementation is used for solving the issue of finding dynamic channels in a protein molecule.

2 BASIC DEFINITIONS

A channel in a protein molecule is defined by the centerline and the volume [MBS07]. The centerline is a three-dimensional continuous curve and the volume is formed by the union of spheres with centers on this centerline and with appropriate radii so that none of the spheres intersects any atom in the molecule. The bottleneck radius of the channel (i.e. the minimum sphere inserted) limits the size of a molecule which is able to pass through a channel. In addition to previous definition, a dynamic channel contains for each point on the centerline the reference to a particular snapshot in the trajectory.

Definition 1: Let $M = \{m_1, ..., m_n\}$ be a time-ordered set of snapshots. Dynamic channel T is defined as $T = \bigcup_{x \in a_T} s(x,r,i)$ where a_T is the centerline of T (a three-dimensional curve) and $s(x,r,i)$ is a sphere with center x and radius r which does not intersect nor contains inside any atom in the reference snapshot m_i.

The definition of a dynamic channel is general. For the purposes of this paper there is a need to apply various constraints. The constraints may be geometrical (such as minimal width) or time (such as the order of snapshots). For each dynamic channel we are able to determine if it satisfies the given constraints. For example, if a dynamic channel with the centerline a_T, connecting an active site with the surface is to be an egression path, the basic constraint which should be satisfied is that the reference snapshot numbers should be increasing from the active site. More formally, for each $a, b \in a_T$ such that a is closer in a_T to the active site than b, if the reference snapshot number for a is smaller than reference snapshot number for b, then the path satisfies the egression path constraint. Otherwise, the constraint is not satisfied.

3 RELATED WORK

3.1 Motion planning methods

In the environments with static obstacles, there are various approaches that can be used to find a collision-free path from a starting position to a destination position. Possible methods are cell decomposition [Lin04], visibility graphs [JLK95], Minkowski sum [Gho90] and potential fields [HA92]. The issue of path planning gets more complicated when considering environments with moving obstacles.

In the dynamic environments, there are approaches which are designed for the two dimensional case. As an example, spatial indexing [FS88], velocity obstacles method [FS98] and hierarchical strategies [FS89] can be mentioned.

The methods designed for three-dimensional environments with moving obstacles are much rarer. As an example the following methods introduced in [YJSHJ06], [BKZ$^+$07] and [HKcLR00] can be mentioned.

Our solution of path planning is done in an environment with different settings. Firstly, obstacles in our environment are always spherical and all of them are moving. Secondly, all obstacles are of similar size and their movement is known a priori. In this environment, our method tries to find paths connecting the starting position with a destination position which satisfy user-defined constraints. For each path the maximum size of the spherical robot which is able to pass through can be determined.

The proposed solution utilizes computational geometry principles Voronoi diagram and Delaunay triangulation [SU00] and is based on the breadth first search algorithm (BFS) [CLRS01].

3.2 Channels

The issue of computation of channels in one snapshot of the molecule was described in [POB$^+$06] and the improved approach based on computational geometry was proposed in [MBS07]. Other approaches which appeared later [PKKO07, YFW$^+$08] are based on similar concepts.

The extension to dynamics where channels are computed separately in each snapshot was proposed in [BMS09]. It should be stressed that the cluster analysis which was utilized in this issue is not directed to find a dynamic channel as defined above since there is no explicit notion of continuous movement through the set of channels clustered together.

4 PROPOSED METHOD

The input data for the proposed method include the geometric model of the molecule which is composed of a set of spheres in three dimensions. The molecule contains thousands of atoms; each atom corresponds to a sphere in the model. Additionally, the movement of atoms is sampled with a fixed frequency, producing a set of snapshots (a trajectory). Each snapshot represents the positions of atoms in a certain instance of time and in fact is a static model. The capture of snapshots is dense enough so that no important movements of atoms are missed. This implies that there is a certain limit for the translation of each atom between consecutive snapshots.

In addition, it is necessary to specify the starting point and the set of destination points. For protein analysis, starting point should be located in the active site cavity and destination points may be automatically determined as points lying on the surface of the molecule.

It should be noted that the trajectory typically describes local movement of atoms only, while the global

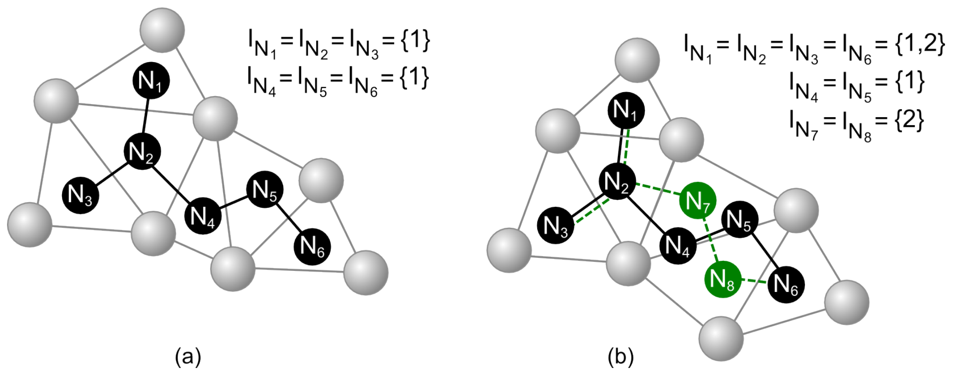

$I_{N_1} = I_{N_2} = I_{N_3} = \{1\}$
$I_{N_4} = I_{N_5} = I_{N_6} = \{1\}$
N_1 N_2 N_3 N_4 N_5 N_6
(a)
$I_{N_1} = I_{N_2} = I_{N_3} = I_{N_6} = \{1,2\}$
$I_{N_4} = I_{N_5} = \{1\}$
$I_{N_7} = I_{N_8} = \{2\}$
N_1 N_2 N_3 N_4 N_5 N_6 N_7 N_8
(b)

Figure 1: Construction of a graph from the Delaunay triangulations. (a) Graph after processing of the first snapshot. (b) In the next snapshot, the movement of atoms caused changes in the Delaunay triangulation. New edges (dashed green) and nodes (green) were added into a graph after processing this snapshot.

movement of the molecule is not present. Various alignment tools can be used in case the data do not satisfy this condition.

The proposed method is purely geometrical. It processes the geometric model of the molecule which consists of a set of spheres in three dimensions; each atom corresponds to one sphere. No biochemical properties are considered. This makes the method general and applicable in similar situations and similar dense environments in which feasible paths are to be computed. For protein analysis, the method can be adjusted so that biochemical properties are taken into account to find the most prospective and biochemically relevant paths.

The method is composed of two main phases – the graph creation phase and the graph processing phase. In the first phase, the Voronoi diagrams (or dual Delaunay triangulations) are merged into a large graph. This graph captures the behaviour of the geometric model over time. In the second phase the graph is processed searching for feasible paths which satisfy defined constraints.

4.1 Phase 1: create graph

Recall that $M = \{m_1, ..., m_n\}$ be a time-ordered set of snapshots. For each snapshot m_i, we compute the Delaunay triangulation DT_i that maintains the space partitioning of the molecule. The Delaunay triangulation is computed using the QuickHull algorithm [BDH96]. It should be stressed that each tetrahedron in the triangulation contains the indices of four atoms in the molecule. In general, all triangulations DT_i for M are merged into one multi-edge graph G.

Due to the fact that there is a fixed number of different atom radii and that the radii do not vary greatly, the Delaunay triangulation for atom centers can be used. For general situations in which the size of spheres (obstacles) would differ more, the additively weighted Delaunay triangulation would have to be computed.

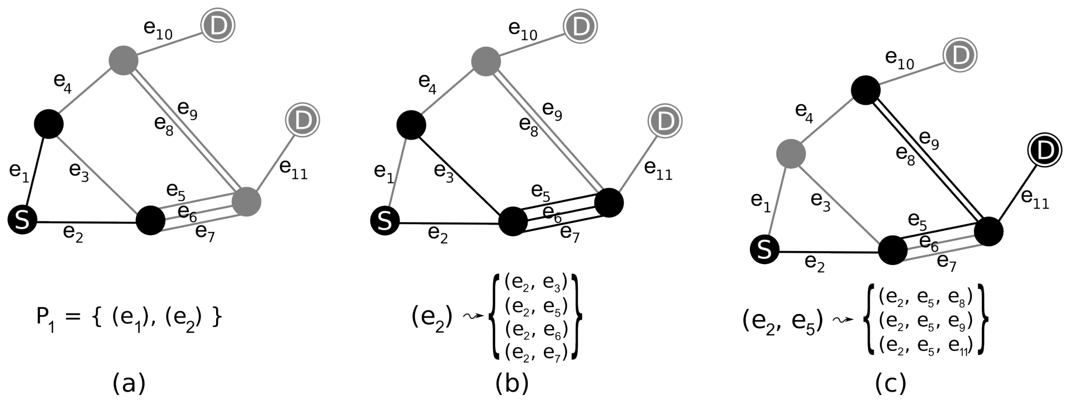
The algorithm starts with an empty graph G. For each snapshot m_i and for each tetrahedron $t \in DT_i$, a new node N containing a reference to t, is created, $Ref(N) = t$. Two nodes are considered equal if the atom indices of their reference tetrahedra are equal. If G does not contain an equal node, then N is inserted into G. This ensures that the associated tetrahedra for nodes are unique.

Moreover, for each node N a set I_N of indices of snapshots is maintained. If $i \in I_N$, then $Ref(N)$ was a tetrahedron in the Delaunay triangulation in the snapshot m_i.

Let N_1 and N_2 be the two nodes in G. For each snapshot m_i in which tetrahedra $Ref(N_1)$ and $Ref(N_2)$ share a face, a new edge e connecting N_1 and N_2 is inserted into G, then $e = (N_1, N_2, i, v)$ where i is the snapshot number and v is the representation of width. For short, let $Snap(e)$ denote the snapshot number of e, i.e. $Snap(e) = i$. The process of computing the width representation follows.

For each edge $e = (N_1, N_2, i, v)$ in G, a Voronoi edge exists which is dual to the shared face between two tetrahedra $Ref(N_1)$ and $Ref(N_2)$ in DT_i. The value of v can be computed as the distance between the corresponding Voronoi edge associated with e and the surface of the nearest atom in i-th snapshot. The issue of computation of the values on edges for a Delaunay triangulation from only one snapshot was addressed in [MBS07]. The referred value v represents the maximum possible radius of a spherical probe which is able to pass along the corresponding Voronoi edge without colliding with any of the atoms. As a result, a multi-edge graph G is created. Note that there may be more than one edge between two nodes in G if associated tetrahedra for these nodes were present in the Delaunay triangulation in more than one snapshot or even no edge if both associated tetrahedra did not share a face in any of the triangulations DT_i. The example of such a graph is shown in Fig. 1.

To keep the number of edges in G reasonable, it is convenient to filter the edges in G by their value. If the computed value v for an edge is less or equal to



$P_1 = \{ (e_1), (e_2) \}$
$(e_2) \rightsquigarrow \{ (e_2, e_3), (e_2, e_5), (e_2, e_6), (e_2, e_7) \}$
$(e_2, e_5) \rightsquigarrow \{ (e_2, e_5, e_8), (e_2, e_5, e_9), (e_2, e_5, e_{11}) \}$
(a)
(b)
(c)

Figure 2: Modified BFS. Starting node (S), destination nodes (D). (a) First step of the algorithm in which paths of length one are created. (b) Extension of a path $p = (e_2)$ with edges emanating from the last node of p. (c) Extension of path $p = (e_2, e_5)$ during which a path between starting and destination node $p_{new} = (e_2, e_5, e_{11})$ is found.

user-defined limit, the edge is not added to G. Without filtering, all edges are used regardless of their value.

Once the multi-edge graph is computed, it can be stored for further processing. This graph represents the discrete evolution of the proximity information about the geometric scene over time.

The starting node in G from which the search for paths begins is selected by its associated reference tetrahedron. For the purposes of protein analysis, a node whose tetrahedron encloses three-dimensional coordinates of the active site is selected. In other applications, an arbitrary node may be selected.

For the purposes of protein analysis, the paths which end on the convex boundary of the molecule are important. Each node in G whose associated tetrahedron appeared on such boundary of the molecule in at least one snapshot is marked as a destination node. In other applications, a set of destination nodes may be selected according different criteria.

4.2 Phase 2: process the graph and find constrained paths

Given multi-edge graph G and starting node and a set of all destination nodes we search for paths satisfying a set of constraints. The straightforward solution in which no path is excluded is to generate all possible paths which emit from a starting node. To find such paths, G is processed using the modified breadth-first search (BFS) algorithm.

A path p consists of edges from G, $p = (e_1, ..., e_m)$ where $e_i \in G$. The length of a path is denoted as the number of edges in p. The modified BFS generates paths in steps; in each step, paths of length increased by one are created.

The paths which consist of consecutive edges in G and connect the starting node with one of the destination nodes can be mapped to a centerline of a dynamic channel. The mapping is described later.

The modified BFS search algorithm works as follows. In the first step, paths of length one (containing a single edge which originates from the starting node) are created. In the next step, all paths from previous step are extended with edges emanating from the last node in the path being extended. To prevent cycles, a path is extended with edges whose end node is not already present in the path. With this approach the set of all feasible paths of variable length from the starting node to all other nodes are computed.

The generation of new paths may terminate once a path connecting the starting node with any of the destination nodes is found. Alternatively, the generation of paths may continue to find more than one path. Also, if multiple destination nodes were selected, it may be crucial to find all suitable paths ending in some of these nodes. Otherwise, the computation terminates after paths to all reachable nodes are checked.

The described generation of feasible paths produces a vast number of paths. If there was an unlimited computational power and storage, it would be possible to search the whole graph for all paths connecting starting and destination nodes. Currently, this is not possible and various restrictions have to be applied to keep the number of paths reasonable. The description of various techniques which help us to cope with the number of paths follows, including the concept of constraints.

As mentioned above, paths are generated in steps. In the first step, the set of paths P_1 of length one is created; each of these paths has exactly one edge which emanates from the starting node. Then, in $(i+1)$-th step, each path $p = (e_1, ..., e_i)$ from P_i is extended with every edge e_{new} which emanates from the last node in the path p (except for cases when p already contains the end node of e_{new} to prevent cycles) resulting in a new path $p_{new} = (e_1, ..., e_i, e_{new})$ which is inserted into P_{i+1}. Finally, the set P_i is discarded and no longer maintained.

The example progress of the algorithm is shown in Fig. 2.

To keep the number of paths reasonable, it is convenient to filter non-prospective paths during the extension process. More precisely, in the i-th step, each path of length i can be extended by many edges to form paths of length $i+1$. The concept of constraints is utilized to decide whether an extension is appropriate instead of blindly creating extended paths and filtering them afterwards.

Let us now explain various constraints and their application in the process of extending a path $p = (e_1, ..., e_m)$ with an edge e_{new}, resulting in a new path $p_{new} = (e_1, ..., e_m, e_{new})$. The following list describes important constraints which take into account the minimum clearance of the path, proper flow of time, maximum speed of a robot, waiting of a robot and a limited curvature of the path. The list, however, is not exhaustive and one can design other constraints that can be applied. The selection of constraints below is motivated by the computation of the widest straight-leading paths which are safe to pass along over time.

- **Minimum radius of a path**. If the width of e_{new} is lower than the minimum value specified, the extension does not happen and p_{new} is not created. See Fig. 3 (b).

- **Increasing snapshot number**. For paths, which should serve as egression paths from the active site, the snapshot numbers for edges in the path have to be increasing. If $Snap(e_{new}) < Snap(e_m)$, the extension does not happen since the extended path would not satisfy the constraint. Additionally, for the access paths, the decreasing order of snapshot numbers has to be used. This constraint is the most important for the validity of the path over time. See Fig. 3 (c).

- **Speed of the robot**. If there is more than a certain number of edges having the equal snapshot number in the path, it would be impossible for a robot to pass through in such a short time. Let $time$ be the time between the each two consecutive snapshots in the trajectory. If there is a certain number of edges having the equal snapshot number in the path, the sum of lengths of these edges divided by the speed of the robot cannot exceed $time$. In case all edges in the multi-edge graph are of approximately the same length, the constraint can be formulated in the following simpler form. If the extension of a path causes that a last certain number (experimentally derived) of edges have equal snapshot number, the extension does not happen. See Fig. 3 (d).

- **Waiting of a robot in a graph node**. For this constraint, the additional information stored for each node in I_N is utilized. Note that I_N contains snapshot numbers in which it was safe for the robot to wait in node N (Fig. 1). Let N be the node shared by e_m and e_{new}. If I_N contains all integer numbers from interval $< Snap(e_m), Snap(e_{new}) >$ and the radius of the inscribed sphere to a tetrahedron $Ref(N)$ is larger than the size of the robot, the transition between e_m and e_{new} is safe and the robot is able to wait in N. See Fig. 3 (e).

- **Curvature of the path**. For every new edge e_{new} to be added we are able to compute an angle α between e_{new} and e_m. If α is above a user specified value the extension does not happen. If desired, this allows us to exclude paths exceeding a user-defined curvature threshold. See Fig. 3 (f). Note that this definition limits the angle between each pair of consequent edges in the path only. For our purposes this definition is sufficient. However, other definitions which consider the whole path can be used.

For the practical case, the number of paths may be extremely large for huge multi-edge graphs even if previously mentioned constraints are applied. One of the other possibilities to reduce the number of paths is adding new constraints. There are also other techniques which do not belong to the category of constraints but still help us to reduce the number of paths so that the computation is feasible when only a limited amount of memory is available. The techniques typically result in the loss of paths.

In i-th step, when paths are created, we can stop the extension and proceed to the next step if the number of generated paths of length i is larger than a user defined value. It is obvious, that this approach prefers such paths which are to be found earlier in the process of generating paths.

The alternative approach we propose is to leave only one path for each node in G. This way, a number of paths is limited by the number of nodes in G. It is crucial that the best candidate path is kept for each node to reduce the number of lost paths. Let $P(N)$ be a set of paths whose last node is N. The selected candidate path $p \in P(N)$ for a certain node N is such a path $p = (e_1, ..., e_m)$ for which $Snap(e_m)$ is minimal. This is the safest way which ensures that if there exists a path satisfying the most important time flow constraint (increasing / decreasing order of snapshots) it will be generated.

Our algorithm is general enough that the output of paths can also be done during the extension process. If the ending node of e_{new} is marked as destination, it is clear that a path which connects the starting node with the destination node exists. The path is stored and is no longer extended. However, the process of generating paths may continue to find other different paths connecting the starting node with a destination node.

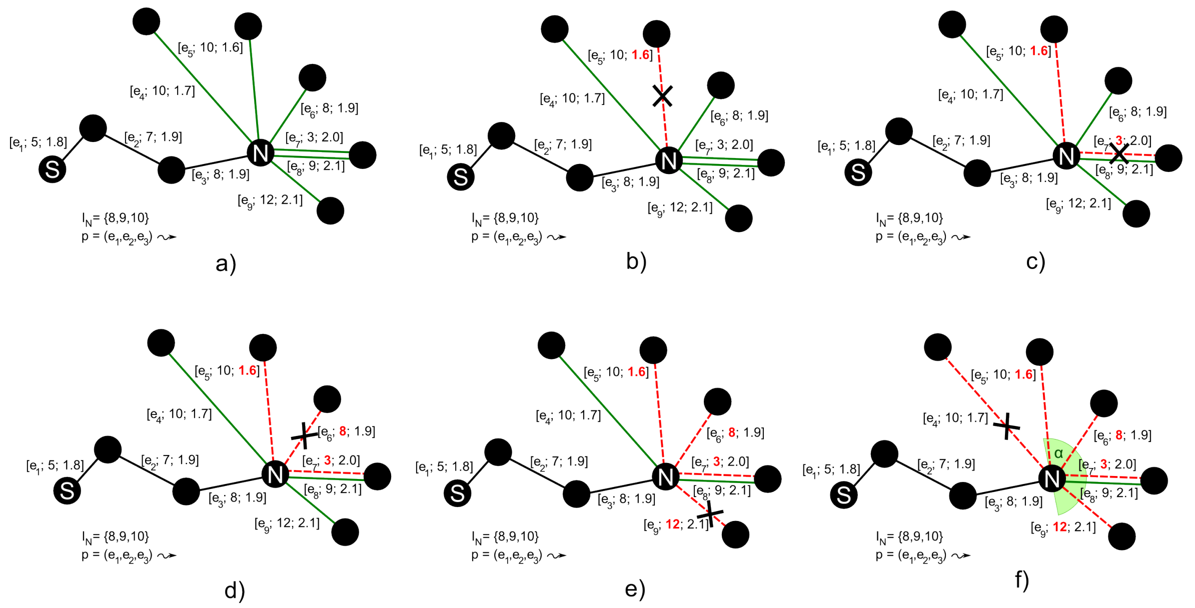


Figure 3: Extension of a path $p = (e_1, e_2, e_3)$ with e_4, e_5, e_6, e_7, e_8, e_9 and applying constraints. Crossed dashed lines denote the edges which do not satisfy given constraints. (a) Extension without constraints. (b) Minimum-width constraint with value of 1.7. Edge e_5 does not meet the minimum width requirement. (c) Time-flow constraint. The path is treated as egression and the numbering of snapshots should be increasing. Newly added edge e_7 does not satisfy the constraint. (d). Speed of the robot. In this case, we do not allow two consecutive edges to have the same snapshot number. Edge e_6 does not satisfy the constraint. (e) Waiting of the robot. Edge e_9 does not satisfy the constraint as some of the snapshot numbers between 8 and 12 is missing in I_N. (f) Angle constraint. Edge e_4 does not satisfy the constraint of maximum angle of 90 degrees shared by e_3 and e_4. The angle α which satisfies the constraint is emphasized.

5 BIOCHEMICAL APPLICATION: MAPPING A PATH TO A DYNAMIC CHANNEL

A path connecting the starting node with a destination node is mapped to a dynamic channel in the following way. The centerline of a dynamic channel consists of the Voronoi edges dual to the faces shared by tetrahedra associated with the nodes in G and new edges are added to ensure that the resulting centerline is continuous to meet the definition of a dynamic channel.

More precisely, each edge $e_1 = (N_1, N_2, s_1, w_1) \in G$ after which follows $e_2 = (N_2, N_3, s_2, w_2) \in G$ is mapped to a centerline as a Voronoi edge ve_1 dual to the face shared by $Ref(N_1)$ and $Ref(N_2)$ in snapshot s_1. If both e_1 and e_2 originate from different snapshots, additional edges have to be mapped to the centerline. For each pair of successive i_1, i_2 in the ordered set of integer numbers from $I_{N_2} = \{s_1, ..., s_2\}$, a new edge ge_i is added to the centerline which connects Voronoi vertex dual to $Ref(N_2)$ in snapshot i_1 and the Voronoi vertex dual to $Ref(N_2)$ in snapshot i_2. The whole process is demonstrated in Fig. 4

6 RESULTS

The method was tested on trajectories which were computed using a computer simulation of a protein molecule. In the visualization of these data, it can be clearly visible that a small molecule (a ligand) leaves the active site defining an egression path. In addition, the molecule does not remain open during the egression which means that recent approaches which compute channels separately in each snapshot fail to detect such a path.

For the analysed trajectories, the ligand was naturally excluded from the computation. The above mentioned constraints were used searching for egression paths; their settings were identical for all analysed trajectories. After constrained dynamic channels were found, these channels were compared against the egression path of a ligand. Table 1 shows the results of the comparison. We have inspected the bottleneck radius (in Angstrom, Å, 1 Å$= 10^{-10}$ m) of the dynamic channel and the distance between dynamic channel and the ligand position in the appropriate snapshot. Average value of this distance is shown in the table. In the trajectories 1-3, the egression path was computed by our method which was almost identical with the egression path of the ligand. For trajectory 4, our method has detected wider possible egression path which exists, whereas the ligand in the simulation chose different path. In this case, the size of the ligand (Table 1) is smaller and more possible paths exist. In all cases, the bottleneck radius of computed dynamic channel was the same or slightly smaller than

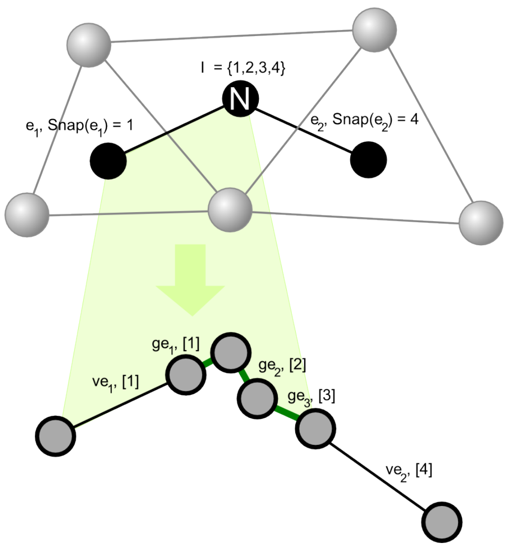


Figure 4: The example mapping of an edge e_1 in a path (upper part) to the part of a dynamic channel centerline (lower part). For the edge e_1, the Voronoi edge ve_1 is added to the centerline together with edges ge_1, ge_2 and ge_3. The ge edges show the waiting of the robot and ensure that the resulting centerline is continuous.

the radius of a ligand bounding sphere. This was caused by the non-spherical shape of the ligand which can pass through a channel only when properly oriented. It can be noticed that for trajectory 4, the number of computed paths was significantly larger than for other trajectories. Since all these paths satisfied the used constraints, additional filtering is convenient to select the most important of them.

The example visualization of a dynamic channel in selected snapshots using pyMol visualization software [Del02] is shown in Fig. 5. In each snapshot, only a part of the dynamic channel which is valid in this particular snapshot is visualized. Standard visualization method of insertion of empty spheres on the centerline is used. Notice that the path is blocked by atoms in the second and fourth selected snapshot – this is exactly the case in which methods that compute channels in each snapshot independently fail to detect the channel.

It should be stressed that in the testing data the behaviour of the molecule over time is sampled densely enough so that the translation of each atom in the two consecutive snapshots is kept within a reasonable limit and no crucial movement is omitted. This ensures that it would be feasible for a small molecule to pass through a computed path without any collision.

The time required to find paths satisfying the constraints depends on the number of edges in G and on the constraints applied. However, in the worst case the modified BFS algorithm may process all paths which may require vast amount of time and memory. More precisely, let c be the number of nodes in G and k be the number of snapshots. Then, the maximum number of edges between a pair of nodes is k. In the worst case, the number of all paths of length d is $\prod_{i=1}^{d} k(c-i)$. With the constraints and the cutting of paths applied after each step, the number of paths generated is significantly smaller. For protein trajectories, the time required to find paths with all mentioned constraints was hours on a common desktop computer (single core 2GHz, 2GB of memory). With the reduction of paths in each step to the number of nodes in G, the time even decreased rapidly to approx. 20 minutes still generating reasonable paths. It should be noted that the processing time is not crucial since the molecular dynamics simulation takes days to compute.

7 CONCLUSION

We have presented a new geometric method, which is capable of finding constrained paths in a three dimensional space with a huge number of moving spherical obstacles. These obstacles may not only move arbitrarily, but also may overlap. Various constraints were designed which can be applied to filter non-prospective paths. We have presented the modified breadth-first search algorithm which generates feasible paths with the constraints applied during the generation process.

We have also demonstrated the application of the method to an interesting biochemical problem of finding a dynamic channel in a trajectory. Due to limited computing resources currently available, trajectories do not cover large period of protein life and that egression may not happen in the short interval of simulation. Despites this, we expect the method to have a great potential once the data is available. We have also shown that for the available data with egression paths confirmed by other means, these collision-free paths were found by our method.

8 FUTURE WORK

In the future, we would like to focus on the validation of biochemical relevance of computed paths on large protein trajectories. Moreover, in cooperation with biochemists it would be necessary to include various additional biochemical constraints to improve the relevance of the results if still a large number of paths exists.

So far, the method assumes that the radii of obstacles do not vary greatly. The modification of the method for environments with variable-size spherical obstacles could be accomplished using the additively weighted Voronoi diagram and the corresponding triangulation.

9 ACKNOWLEDGMENTS

This work was supported by The Ministry of Education of The Czech Republic, Contract No. LC06008 and by The Grant Agency of The Czech Republic, Contract

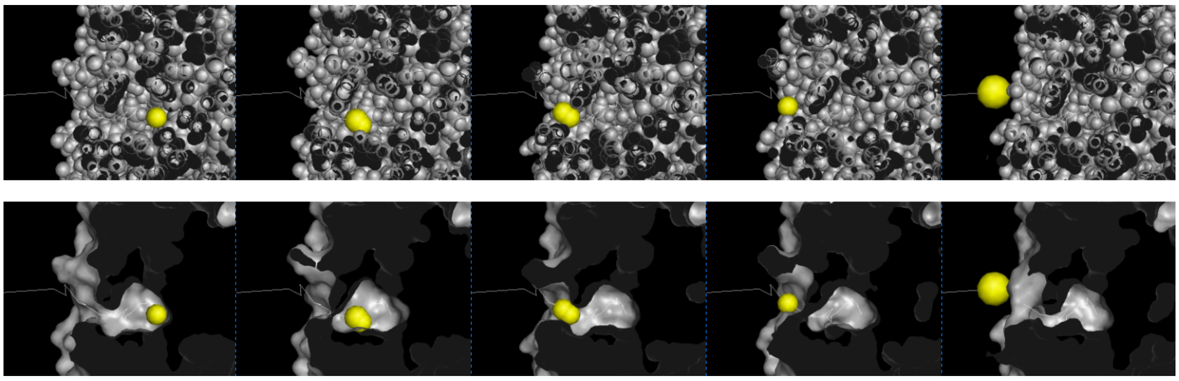


Figure 5: Example visualization. Side view of a protein molecule, cut with a cutting plane perpendicular to the view. The resulting egression path is visualized using yellow spheres defining the safe collision-free positions of a spherical robot inside the molecule over time (example for six selected snapshots). The molecule is visualized using two basic visualization methods (upper panel – sphere model, lower panel – surface model).

id	molecule (codename)	ligand egression	dynamic channel bottleneck radius	number of paths found	average distance between ligand egression and one of the paths computed
1	LinB WT	cyclohexanol (OCX)	2.7 Å	59	0.8 Å
2	LinB L177W	cyclohexanol (OCX)	1.4 Å	43	3.8 Å
3	LinB L177W	2-bromoethanol (BEO)	1.3 Å	1	2.01 Å
4	LinB L177W	bromide ion (Br-)	1.4 Å	1588	7.7 Å

Table 1: The properties of computed constrained dynamic channels for selected trajectories.

No. P202/10/1435. We would also like to acknowledge Lada Biedermannova from Heidelberger Institut for Theoretical Studies for the supplied trajectories.

REFERENCES

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[BKZ+07] P. Broz, I. Kolingerova, P. Zitka, R. Apu, and M. Gavrilova. Path planning in dynamic environment using an adaptive mesh. *Spring Conference on Computer Graphics, ACM proceedings*, pages 172–178, 2007.

[BMS09] P. Beneš, P. Medek, and J. Sochor. Computation of channels in protein dynamics. *In Proceedings of the IADIS International Conference Applied Computing 2009*, 2:251–258, 2009.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[Del02] W. L. Delano. The pymol molecular graphics system, 2002.

[FS88] K. Fujimura and H. Samet. Path planning among moving obstacles using spatial indexing. *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 1662 –1667 vol.3, apr. 1988.

[FS89] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles [mobile robot]. *Robotics and Automation, IEEE Transactions on*, 5(1):61 –69, feb. 1989.

[FS98] Paolo Fiorini and Zvi Shillert. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

[Gho90] Pijush K. Ghosh. A solution of polygon containment, spatial planning, and other related problems using minkowski operations. *Comput. Vision Graph. Image Process.*, 49(1):1–35, 1990.

[HA92] Y.K. Hwang and N. Ahuja. A potential field approach to path planning. *Robotics and Automation, IEEE Transactions on*, 8(1):23 –32, feb. 1992.

[HKcLR00] David Hsu, Robert Kindel, Jean claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles, 2000.

[JLK95] J.A. Janet, R.C. Luo, and M.G. Kay. The essential visibility graph: an approach to global motion planning for autonomous mobile robots. volume 2, pages 1958 –1963 vol.2, may. 1995.

[Lin04] F. Lingelbach. Path planning using probabilistic cell decomposition. volume 1, pages 467 – 472 Vol.1, apr. 2004.

[MBS07] P. Medek, P. Beneš, and J. Sochor. Computation of tunnels in protein molecules using delaunay triangulation. *Journal of WSCG*, 15(1-3):107–114, 2007.

[PKKO07] Martin Petřek, Pavlína Košinová, Jaroslav Koča, and Michal Otyepka. Mole: A voronoi diagram-based explorer of molecular channels, pores, and tunnels. *Structure*, 15(11):1357–1363, November 2007.

[POB+06] Martin Petřek, Michal Otyepka, Pavel Banáš, Pavlína Košinová, Jaroslav Koča, and Jiří Damborský. Caver: a new tool to explore routes from protein clefts, pockets and cavities. *BMC Bioinformatics*, 7(1):316+, June 2006.

[SU00] J.-R. Sack and J. Urrutia. *Handbook of computational geometry*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2000.

[YFW+08] Eitan Yaffe, Dan Fishelovitch, Haim J. Wolfson, Dan Halperin, and Ruth Nussinov. Molaxis: Efficient and accurate identification of channels in macromolecules. *Proteins: Structure, Function, and Bioinformatics*, 73(1):72–86, 2008.

[YJSHJ06] Kwon Kyoung Youb, Cho Jeongmok, Kwon Sung-Ha, and Joh Joongseon. Collision avoidance of moving obstacles for underwater robots. *Journal of Systemics, Cybernetics and Informatics*, 4(5):86–91, 2006.