Discovering molecules: Pass planning through a gap

Mgr. Jan Byška*
Supervised by: doc. Ing. Jiří Sochor, CSc.†

Department of Computer Graphics and Design
Faculty of Informatics, Masaryk University
Brno, Czech Republic

Abstract

We present a new algorithm for a molecular pass planning through a circle. Our algorithm can solve the given problem with the significant improvement of accuracy for arbitrary shaped molecules in comparison with the method using a minimal bounding sphere. This accuracy is gained by eliminating the overestimation of the substrate size by the bounding volume approaches. Our approach is particularly beneficial in cases where the bounding volume fits poorly to the substrate geometry as is the case with oblong shaped substrates. We are using a sampling-based version of the motion path planning and the Delaunay triangulation to arrange the substrate for the space search. The successor configurations are then computed incrementally from the already known configurations until we find a connected path of the substrate through a circle or we can claim that such path does not exist.

Keywords: motion path planing, Delaunay triangulation, computational chemistry

1 Introduction

Proteins are irreplaceable parts of every life form. They are involved in many vital processes, for instance they catalyze various chemical reactions, work as antibodies or even transfer signals between distant cells. The detailed analysis of the complex protein structures using methods of computational geometry can significantly help understanding the purpose and chemical principles of specific protein molecules.

One of the most researched properties of the protein molecule is the existence of a protein channel. The channel represents an empty space connecting the inner part of the protein, called *active site*, with its surface. They allow, upon certain conditions, other molecules (*substrates* or *ligands*) to reach the active site, where the reaction between protein amino acids and the substrate can undergo. For biochemists, the knowledge, whether the substrate can enter the protein or not, is important for instance in the process of drug design.

*xbyska@fi.muni.cz
†sochor@fi.muni.cz

As was shown by Petr Medek et al. [3], protein channels can be represented as circular-profile shaped tunnels. The current methods use a minimal bounding sphere (*MBS*) approach to compute whether a substrate can pass through the protein channel. This method is simple because we only need to compute the MBS enclosing all atoms of the substrate and then compare its radius with the narrowest cross-section radius of the channel. However, it is not suitable in cases where the substrate has an oblong or polymer-like shape. In these cases the protein channel has to be significantly bigger to let through the overestimated MBS in comparison with the real scenario, where the oblong substrate can pass through a narrower channel.

The method could be improved by using different types of bounding volumes. The real molecules, however, have often very complicated structures and therefore this solution will not suffice in general cases. We decided to develop a new algorithm based on the motion path planning. Our algorithm can solve the given problem for arbitrary shaped molecules since it is not using any bounding volume but the substrate geometry itself. It computes a path of a set of spheres (which represent atoms of a substrate) through a gap approximated by a circle. Moreover, the algorithm can be generalized to use a cross-section of the arbitrary shape.

As such it can be used in the process of detection whether the ligand can pass through the protein channel. If we make an assumption that these channels consist of large and wide corridors connected by narrow gaps we can then, without loss of generality, anticipate that these wide channel segments are large enough for the substrate to pass through in an arbitrary configuration. Hence, we can use previous inaccurate but fast algorithms for these parts of the channel and focus rather on the narrow holes connecting them. The narrow parts of the protein channel can be, for instance, sampled by a set of circles and our algorithm can handle each circle individually.

The goal of this paper is to present the new algorithm that can detect whether a given substrate can pass through a circle. In the section 3 we briefly remind some of the basic terms of motion planning theory. In the section 4 we focus on the detail description of the algorithm. The evaluation of the complexity of the algorithm is presented in the section 5 and we conclude with the section 6.

2 Previous work

One of the most important task in the computational chemistry is called a channel detection. The approach based on the computational geometry that uses the Voronoi diagram and the Delaunay triangulation to compute a protein channel as a set of spheres, was described by Petr Medek et al. [3]. The connected channel can be created from this set by an interpolation. The problem of a substrate passing through the protein channel can then be solved by comparing the bounding sphere of the substrate with the smallest radius of the channel.

Another method for solving this problem was presented by Haranczyk and Sethian [1]. They sampled the high-dimensional configuration space to find a path through a protein.

There are generally two possible approaches to handle the motion path planning problem. The *combinatorial approach* [2] or the *sampling-based approach* [2]. We will describe them in detail in the next section, after the necessary background is given. The motion path planning theory was initially developed for the navigation of the robot through the complex environment. In molecular chemistry, however, several major problems were successfully solved by motion planning approaches, such as *protein folding* or *ligand docking*.

The protein folding is the process when a polypeptide is folded and connected into a final protein structure. It was solved by the motion planning by Song and Amato [5]. The second problem – the ligand docking – refers to the issue of a substrate inserting itself into a protein cavity while satisfying other constraints, such as maintaining the low energy. An algorithm using the probabilistic motion planning to compute the most energetically favorable path between any initial and goal ligand shape was presented by Singh et al. [4].

3 Motion planning concept

Since the motion planning theory was initially developed for the navigation of a robot through a complex environment, it is using terms such as a *robot* or a *path*. In this section we briefly remind some of this basic terms and show how to adapt them for a substrate passing through a protein channel (for more information about motion planning see LaValle 2006 [2]).

The basic element of the motion planning theory is a *state*. Each state represents a specific transformation that can by applied to the robot and thus defines a specific position and orientation of the robot in a *world space* $\mathcal{W}$. In our case, the world space is three-dimensional and the robot $\mathcal{R} \subset \mathcal{W}$ represents the substrate moving in it. Therefore the state representing the substrate will be actually a six-dimensional vector $\vec{q}(x_t, y_t, z_t, u_t, v_t, w_t)$ with three positional and three rotational coordinates.

The set of all states is called the *configuration space* and it is often denoted as $\mathcal{C}$. Formally, for each state holds $\vec{q} \in \mathcal{C}$ and the configuration space of a fixed robot in three-dimensional space creates a six-dimensional manifold.

Since $\mathcal{C}$ contains all possible configurations of the substrate (robot $\mathcal{R}$) in $\mathcal{W}$ it as well contains the states that represent situations in which the substrate will collide with a channel wall. The motion planning theory divides the configuration space into two subspaces, which are called the *obstacle space* and the *free space* denoted as $\mathcal{C}_{obs}$ and $\mathcal{C}_{free}$ respectively. The obstacle space contains all states that represent cases when the robot have a collision with an obstacle and the free space contains the rest of them.

Formally, let the $\mathcal{O} \subset \mathcal{W}$ be an obstacle; $\mathcal{R} \subset \mathcal{W}$ be a substrate and $\vec{q} \in \mathcal{C}$. If the substrate position in the world space is denoted by $\mathcal{R}(\vec{q})$ then the $\mathcal{C}_{obs}$ and $\mathcal{C}_{free}$ can be defined as:

$$\mathcal{C}_{obs} = \{\vec{q} \in \mathcal{C} \mid \mathcal{R}(\vec{q}) \cap \mathcal{O} \neq \emptyset\} \quad (1)$$

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs} \quad (2)$$

To solve the motion path planning problem it is essential to find a connected path $\mathcal{P}$ from a starting state $\vec{q}_s$ to a goal state $\vec{q}_g$. Furthermore, the path has to satisfy $\forall \vec{p} \in \mathcal{P} \mid \vec{p} \in \mathcal{C}_{free}$ otherwise the solution would not be valid (see Figure 1).

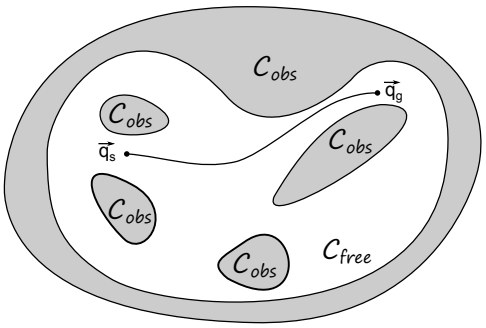

Figure 1: The connected path from $\vec{q}_s$ to $\vec{q}_g$.

Generally, there are two possible approaches to solve this problem. The *combinatorial approach* solves the problem without any approximations. The basic idea is to evaluate the whole free and obstacle space and then find a connected path $\mathcal{P}$ from $\vec{q}_s$ to $\vec{q}_g$ that lies completely in $\mathcal{C}_{free}$. Unfortunately this method has a high computational complexity and therefore is not sufficient for solving practical problems.

Most of the current algorithms are therefore using the *sampling-based approaches*. The configuration space is sampled (usually in a deterministic way) into a finite set of samples. The main concept of this approach is to find only a limited set of points on the path $\mathcal{P}$ instead of the explicit construction of the whole configuration space. The connected path is computed by an interpolation between the nearby samples. On one hand, the complexity of this method is lower but on the other hand the accuracy is lower as well. Both factors usually depend on the number of

samples – the lower number produces more errors while the higher number increases the computational time.

4 Algorithm

In this section we present a new algorithm based on the motion path planning that alleviates the need for a bounding volume approximation of a substrate.We will first describe the algorithm background, then provide an overview of the algorithm after which the necessary conditions for boundary configurations and three running modes of the algorithm will be described. The primary goal of the algorithm is to detect whether a given substrate can pass through a circle. The input of the algorithm is a set of spheres defining the substrate geometry and the radius of the circle through which the substrate should pass.

4.1 Algorithm Background

As shown in Figure 2, the radius of a protein channel varies over its length. Our measurements indicate that some parts of the channel will be wide enough for the substrate to pass through in an arbitrary configuration $\mathcal{R}(\vec{q})$. However, in the real scenario this does not hold for the whole length of the channel. The narrower parts of channel (*gaps*) create channel bottlenecks where the substrate will collide with the channel "wall" (the atoms of the protein) in some configurations or will not pass through them at all. This basically means that if the molecule can pass through all these gaps it can as well pass through the whole channel.

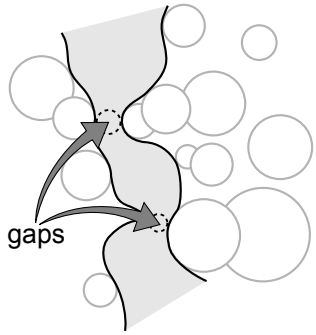

Figure 2: Gaps in the protein channel (adapted from [3]).

Hence we can approximate the problem of substrate passing through the whole channel to the problem of substrate passing through a set of gaps. Moreover, the protein channel can be approximated as a circle-profile shaped tunnel [3] and we can simplify the previous problem to the problem of the substrate passing through a set of circles.

Our algorithm solves the passing problem for each circle individually. The basic idea is to find a sequence of rotations and shifts that will result in the successful passing of the given substrate through the circle, from one side to the other. In this section, we will show that it can be done by using the motion path planning. Note, that if there are two or more circles close enough, we can generalize the problem and for the second circle start in the middle of the passage provided that we have appropriately modified the collision detection and marked all already passed atoms.

The problem of passing substrate through the circle, however, can be inverted to a problem of pulling the circular ring over the static substrate, from one side to the other. Both concepts are almost identical except that the transformations will be inverted. The second view is easier for explanation of the algorithm and therefore will be used in this paper.

Note that $\mathcal{R}$ from now on will denote the new circle robot and the obstacles set $\mathcal{O}$ refers to the spheres defining the substrate geometry. We can also use the fact that the circle robot $\mathcal{R}$ is invariant under the rotation around one of the axis. Hence, the state defining the circle position $\mathcal{R}(\vec{q})$ in the space can be actually reduced to five-dimensional vector $\vec{q}(x_t, y_t, z_t, u_t, v_t)$. Obviously, the reduction of the state-space dimensionality leads to the significant speedups.

4.2 Molecular Pass Planning

We will start with an overview of the algorithm and describe it in detail later. The main idea of this algorithm is to incrementally compute the continuous path of the circle over the substrate from already known configurations.

Pseudocode 1 Molecular pass planning algorithm.

Input: a set of spheres (connected by red and green lines), the circle radius
Output: a connected path of the circle over the substrate

1: find the starting configuration $\mathcal{R}(\vec{q}_s)$
2: $\mathcal{S} \leftarrow \mathcal{R}(\vec{q}_s)$ {*S is a stack*}
3: **loop**
4: **if** $\mathcal{S}$ is empty **then**
5: $exit \longrightarrow fail$: the circle is to small
6: **else**
7: $\mathcal{R} \leftarrow \mathcal{S}.top()$
8: $\mathcal{R}.markAsUsed()$
9: $\mathcal{R}.previous.successor \leftarrow \mathcal{R}$
10: **if** $\mathcal{R}$ satisfy conditions for $\mathcal{R}(\vec{q}_g)$ **then**
11: $end\ loop \longrightarrow success$: there is a path $\mathcal{P}$
12: **else**
13: find the next possible configurations $\mathcal{N}$ from $\mathcal{R}$ according to the mode and put each $n \in \mathcal{N}$ onto the stack $\mathcal{S}$
14: **end if**
15: **end if**
16: **end loop**
17: create a connected path $\mathcal{P}$ using $.successor$ attributes

We have observed that: If there is a connected collision-free path $\mathcal{P}$ of the circle $\mathcal{R}$ from one side of the substrate $\mathcal{O}$ to the other we can cut the substrate at any time by the hyperplane containing $\mathcal{R}$. The cut will then produce a set

of additional circles, which represent the crosscuts of the atoms lying in the cut plane. These circles must not collide with the circle $\mathcal{R}$ otherwise the path $\mathcal{P}$ would not be collision-free (see Figure 3).

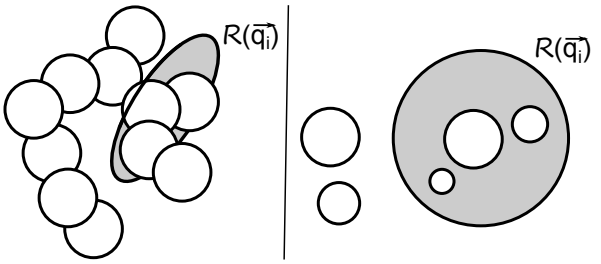

Figure 3: The crosscuts of atoms lying in the hyperplane containing $\mathcal{R}$.

We utilize the fact that all atoms of the substrate have to pass through the circle. In other words each atom has to cross the hyperplane containing the circle $\mathcal{R}$. Therefore we can sample the path $\mathcal{P}$ at this moments which with the previous observation allows us to use the sampling-based version of the motion path planning.

To create a connected path $\mathcal{P}$ of the circle $\mathcal{R}$ over the substrate we need to find a sufficient number of samples and then interpolate between them. The interpolation between two configurations has to be collision-free as well. To achieve that we need to test whether the computed path of the circle will not collide with any atom of the substrate.

This can be done either by the continuous collision detection or by using the sampling-based approach. The continuous collision detection is very computationally demanding in this case because we have to compute the intersections of a collapsed torus with a set of spheres. Hence, we decided to use the sampling approach based on collision detection between the circle and a set of spheres.

Additionally, for a valid detection whether the substrate can pass through the protein channel we need to test it for collisions with the atoms of the protein as well.

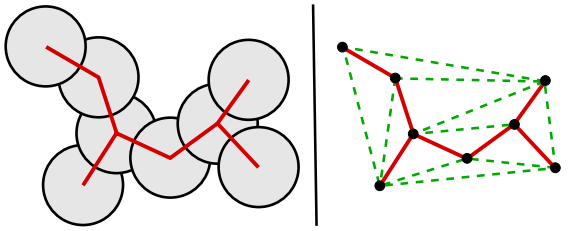

Figure 4: Left: The substrate molecule as a set of spheres connected by red (solid) lines. Right: The ball-and-stick model with the Delaunay triangulation represented by green (dashed) lines.

The approximation of the substrate as a set of spheres is simple enough for computing all collisions during the interpolation process. However, this model is not suitable for the other part of the algorithm. To be able to compute all samples efficiently we need to consider the structure of the substrate, namely bonds between atoms. The appropriate data structure is based on the ball-and-stick model with additional information stored in *green lines* (see Figure 4). The extra lines represent the edges of the Delaunay triangulation computed on the atoms of the substrate and are used for the samples computing.

Our algorithm employs the sample-based approach. As we are only interested in the knowledge whether the substrate can pass through the circle, we can use an incremental sampling. In other words, we do not compute all samples in advance but we compute the first configuration $\mathcal{R}(\vec{q}_s)$ and then compute a next possible configuration $\mathcal{R}(\vec{q}_{s+1})$. If there are more then one successor configurations we will pick one randomly and push the rest onto the stack for further processing. The whole process is then repeated until we reach the goal configuration $\mathcal{R}(\vec{q}_g)$ that represents the case when the substrate had passed through the circle, or we end if there is no valid successor of the previous configuration.

The algorithm is similar to a depth-first search for traversing a tree structure. However, the algorithm does not need the tree structure itself. The only two needed structures are the stack S and the hash table for remembering already visited configurations, to avoid cycling.

The successor configurations are computed incrementally from the known configurations. The successor configuration $\mathcal{R}(\vec{q}_{i+1})$ can be created from the previous configuration $\mathcal{R}(\vec{q}_i)$ by replacing one of it's atoms. The configuration can be represented either by one, two or three atoms, see section 4.4. The conditions for the replaced atom in $\mathcal{R}(\vec{q}_i)$ and for the new one in $\mathcal{R}(\vec{q}_{i+1})$ are as follows:

For the atom that will be replaced, there is no limitation at all. In other words we can replace any atom in the set. From practical point of view, however, the choice can influence whether the incrementally built path $\mathcal{P}$ will lead to the successful passing of the circle over the substrate. Therefore, during the search, we need to compute the whole set of successor configurations for every atom. Then we randomly pick one and store the remaining configurations for eventual processing in the future.

The new atom has to be connected by either red or green line with the atom which will be replaced. This condition will assure that we will not try to make too long jumps. The short steps are necessary because between two distant configurations we cannot easily interpolate to create a connected path.

4.3 Boundary configurations

The motion path planning solves the problem of finding a connected path from $\vec{q}_s$ to $\vec{q}_g$. The starting configuration $\mathcal{R}(\vec{q}_s)$ has to satisfy two conditions. Firstly, the plane containing the circle in starting configuration $\mathcal{R}(\vec{q}_s)$ should cut only one atom. This condition is not compulsory, its purpose is to simplify the problem of correct orientation of

the first configuration in the space. Secondly, all remaining atoms of the substrate have to lie on the same side from the plane defined by the circle.

The problem, that can occur when violating the second condition, is illustrated in Figure 5. If we select the red atom $\mathcal{A}$ as a starting point, the configuration would assume that some atoms had already passed through the gap. To avoid this, we reduce the set of possible starting atoms to atoms lying on the convex hull of the substrate (see Figure 5 right).

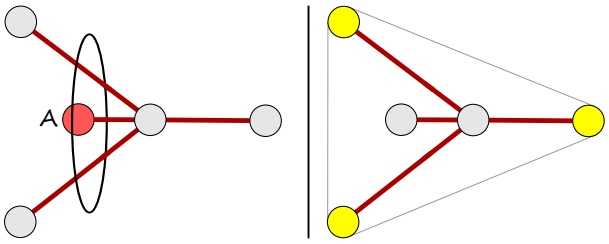

Figure 5: Left: The unappropriate atom choice. Right: The convex hull of the substrate.

However this will not solve the problem completely. We also have to be careful about the size of the circle. Because even though that we choose the starting atom at the convex hull a problem can occurs. There can be another atom $\mathcal{B}$ such that the intersection of its connection line and the hyperplane containing the circle $\mathcal{R}$ will lie inside the circle $\mathcal{R}$ (see Figure 6). This is the case where the second condition will be violated.

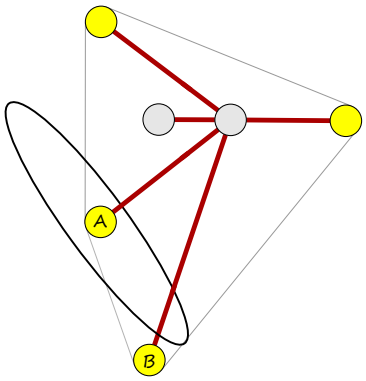

Figure 6: The problem with the convex hull.

In this case, we have to explicitly check whether all atoms of the substrate are lying on one side of the circle defined by the configuration $\mathcal{R}(\vec{q}_s)$.

Similar to the starting configuration there can be more than one end configuration. The conditions and rules that affect the end configuration are almost the same. The difference is that all atoms have to be on the other side of the circle than at the beginning.

4.4 Modes

The algorithm has to deal with all types of molecules. Therefore we propose to use it in three modes, *1D, 2D* and *3D*, each named by a number of atoms they are using for a configuration representation. The 4D or higher modes are not necessary because in real scenarios there are only rare occasions when four or more nearby atoms are lying in one plane and if they do, the situation can be handled by two or more subsequent 3D configurations. Algorithm selects the current mode according to the complexity of the substrate in the specific area. It is possible to switch between modes by adding/removing one atom to/from the current configuration. In one algorithmic step, it is possible to switch between 1D and 2D, or 2D and 3D modes only.

4.4.1 1D Mode

The first mode is responsible for handling polymer-like molecules or parts of complex molecules where atom bonds can be represented as a polygonal chain (Figure 7).

There is always at least one starting configuration represented by a single atom (see section 4.3). Let the current configuration $\mathcal{R}^{1D}(\vec{q}_i)$ be such configuration. At this point the algorithm has two basic possibilities how to compute the successor configuration. It can be another configuration represented by one atom $\mathcal{R}^{1D}(\vec{q}_{i+1})$ or the algorithm can switch from the 1D mode to the 2D mode. The 2D configuration $\mathcal{R}^{2D}(\vec{q}_{i+1})$ is then created by adding an additional atom into the current configuration $\mathcal{R}^{1D}(\vec{q}_i)$.

Pseudocode 2 The 1D mode traversing.

Input: $\mathcal{C}_i$ – current configuration $\mathcal{R}^{1D}(\vec{q}_i)$ from stack S
Output: a successor configuration $\mathcal{C}_{i+1}$

```
1:  A ← C_i.getAtom()
2:  N ← A.getAllNeighbours()
3:  for all B in N do
4:      if B is a complex atom then
5:          C_{i+1} ← the new 2D configuration represented by
            the atoms A and B
6:      else
7:          C_{i+1} ← the new 1D configuration placed in B
8:      end if
9:      if C_{i+1} was not used and a collision-free path from
        C_i to C_{i+1} exists then
10:         C_{i+1}.previous ← C_i
11:         put C_{i+1} onto the stack S for further processing
12:     end if
13: end for
```

In this mode the configuration $\mathcal{R}(\vec{q}_i)$ is represented by one atom $\mathcal{A}$. It is clear that one atom (or rather one point) cannot define the orientation of the circle $\mathcal{R}$ in the three-dimensional space. Hence, we need to define normal vector $\vec{n}$ of the circle $\mathcal{R}$. We utilize the fact that atoms of the substrate's molecule are connected. This mean that every

atom $\mathcal{A}$ has at least one neighbor $\mathcal{B}$ and we denote this connection by the red line. We call the vector $\overrightarrow{\mathcal{AB}}$ a *segment*. The normal vector $\vec{n}$ of the circle $\mathcal{R}$ then corresponds to the segment originating from the atom by which the configuration is represented – in this case atom $\mathcal{A}$ (see Figure 7).

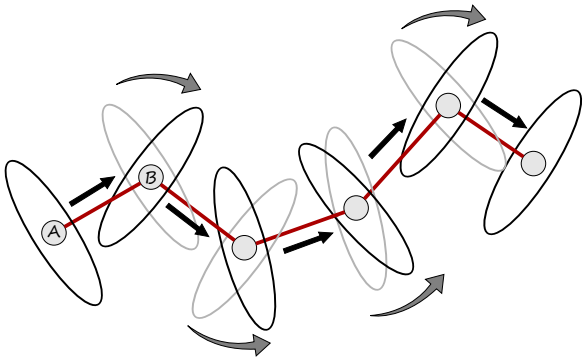

Figure 7: The 1D mode traversing.

As shown in Figure 7, the interpolation between two 1D configurations is a simple translation along the segment and a rotation at the end. The rotation is necessary to orient the circle properly before traversing to the next segment of the polynomial curve defining the substrate.

Whether the successor configuration will be 1D or 2D is determined by the complexity of the molecular structure in the specific part of the substrate. More precisely, if the current atom $\mathcal{A}$ has more then two neighbors or there is at least one green line connecting $\mathcal{A}$ with another atom, then our algorithm will switch to the 2D mode. In this case we call the atom $\mathcal{A}$ a *complex atom* (see Figure 8). Otherwise the algorithm will stay in the current mode and will compute another 1D configuration. The switch from 1D to 2D is done simply by including one of the neighbor atoms connected to $\mathcal{A}$ with red (solid) or green line (dashed).

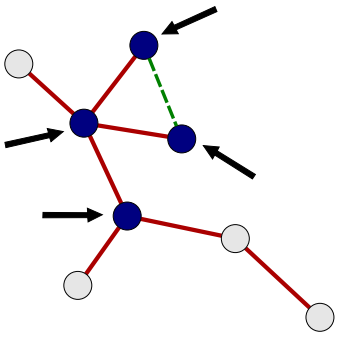

Figure 8: Complex atoms in substrate model.

4.4.2 2D Mode

While in the 1D mode the circle moves from one configuration to another by translation, in the 2D mode the path is built by rotations. The example of such rotational move is shown in Figure 9. The algorithm proceeds from the state $\vec{q}_i$ that represents the position $\mathcal{R}^{2D}(\vec{q}_i)$ of the circle in space and searches for a configuration $\mathcal{R}^{2D}(\vec{q}_{i+1})$. The successor configuration in the 2D mode can be computed by replacing one of the atoms $\mathcal{A}$ or $\mathcal{B}$ by a new atom $\mathcal{C}$. If the smallest circle containing the triangle $\triangle\mathcal{ABC}$ is smaller than $\mathcal{R}$, the algorithm will switch to the 3D mode.

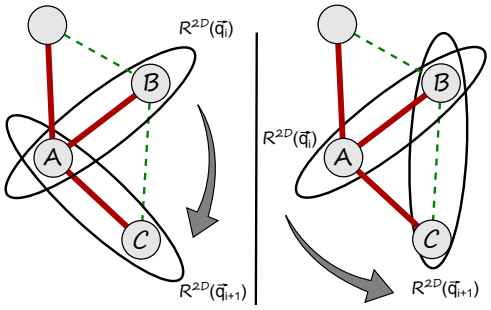

Figure 9: The 2D mode traversing.

Obviously, in 2D mode there may be more then one successor configuration $\mathcal{R}^{2D}(\vec{q}_{i+1})$ If such situation occurs, the algorithm randomly selects one of the possible configurations and others are stored for later processing.

Pseudocode 3 The 2D mode traversing.

Input: $\mathcal{C}_i$ – current configuration $\mathcal{R}^{2D}(\vec{q}_i)$ from stack S
Output: a set of successor configurations $\mathcal{C}_{i+1}$

```
Atoms ← C_i.getAllAtoms()
for all A in Atoms do
    {note that the second atom in C_i is denoted B}
    N ← A.getAllNeighbours()
    for all C in N do
        if the smallest circle containing △ABC is smaller
        then R then
            C_{i+1} ← the new 3D configuration represented
            by the atoms A, B and C
        else
            C_{i+1} ← the new 2D configuration represented
            by the atoms B and C
        end if
        if C_{i+1} was not used and a collision-free path
        from C_i to C_{i+1} exists then
            C_{i+1}.previous ← C_i
            put C_{i+1} onto the stack S
        end if
    end for
end for
```

The 2D configuration $\mathcal{R}^{2D}(\vec{q}_i)$ is represented by two atoms $\mathcal{A}$ and $\mathcal{B}$. The center is situated in the middle of the line connecting the atoms. To represent a circle orientation in the three-dimensional space, we need to compute the normal vector $\vec{n}$.

Let the current configuration $\mathcal{R}^{2D}(\vec{q}_i)$ be defined by two atoms $\mathcal{A}$ and $\mathcal{C}$ and the previous configuration $\mathcal{R}^{2D}(\vec{q}_{i-1})$

be defined by atoms $\mathcal{A}$ and $\mathcal{B}$. The normal vector $\vec{n}$ of the circle $\mathcal{R}^{2D}(\vec{q}_i)$ can be then computed as a cross product $\overrightarrow{\mathcal{AC}} \times \vec{R}$, where $\vec{R}$ is a vector of the rotation from $\mathcal{R}^{2D}(\vec{q}_{i-1})$ to $\mathcal{R}^{2D}(\vec{q}_i)$ and can be computed as $\overrightarrow{\mathcal{AB}} \times \overrightarrow{\mathcal{AC}}$. That gives us the equation for the normal vector:

$$\vec{n} = \overrightarrow{\mathcal{AC}} \times (\overrightarrow{\mathcal{AB}} \times \overrightarrow{\mathcal{AC}}) \tag{3}$$

Note that the orientation of this normal depends on the order of the vectors in the equation. However, if we assume that the rotation from $\mathcal{R}^{2D}(\vec{q}_{i-1})$ to $\mathcal{R}^{2D}(\vec{q}_i)$ is done with the smallest possible angle then the correct orientation of the $\vec{n}$ can than be ensured by a simple rule. We denote the half-spaces induced by the plane containing $\mathcal{R}$ according to Figure 10 and let a point p be the center of the line connecting the centers of both configurations. In order to ensure the correct orientation of the normal $\vec{n}$ the point p has to lie in a differently signed half space.

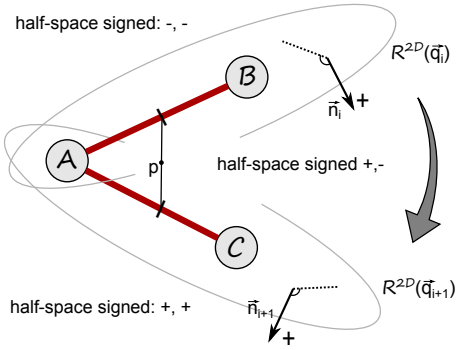

Figure 10: Space divided by two configurations.

Similarly to the 1D mode, it is necessary to apply a small position correction between each step. While in the 1D mode we needed a rotation to put the circle into the right orientation before we could traverse the next segment, here it is a shift which solves the problem of different emplacement of the circle centers. If we simply rotate from $\mathcal{R}^{2D}(\vec{q}_i)$ to $\mathcal{R}^{2D}(\vec{q}_{i+1})$ we would get a different circle center than we have originally computed by replacing atom $\mathcal{B}$ by atom $\mathcal{C}$ (see Figure 11). Hence, before we rotate or after rotation we have to shift the appropriate configuration into the right position.

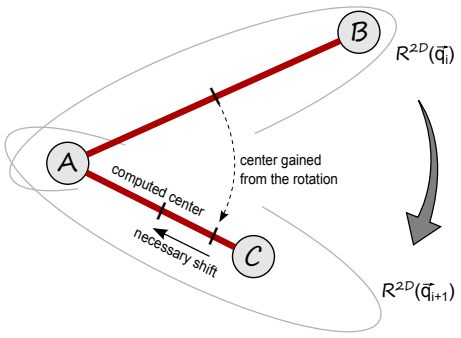

Figure 11: The position correction between each step.

4.4.3 3D Mode

The difference of the 3D mode compared to the previous is that circle in this mode is represented by three atoms $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. This mode provides more degrees of freedom for the rotations between configurations then the previous ones. Additionaly, we do not need to compute the normal vector $\vec{n}$. The center of the circle is defined as a center of the smallest circle containing the triangle $\triangle ABC$. The problem caused by center difference during the passing from $\mathcal{R}^{3D}(\vec{q}_i)$ to $\mathcal{R}^{3D}(\vec{q}_{i+1})$ is the same as in the 2D mode and it may be also solved by applying translation before the rotation.

Pseudocode 4 The 3D mode traversing.

Input: $\mathcal{C}_i$ – current configuration $\mathcal{R}^{3D}(\vec{q}_i)$ from stack S
Output: a set of successor configurations $\mathcal{C}_{i+1}$

```
Atoms ← C_i.getAllAtoms()
for all A in Atoms do
    {the remaining atoms in C_i are denoted B and C}
    N ← A.getAllNeighbours()
    for all D in N do
        if the smallest circle containing △BCD is smaller
        then radius of R then
            C_{i+1} ← the new 3D configuration represented
            by the atoms B, C and D
        end if
        if C_{i+1} was not used and a collision-free path
        from C_i to C_{i+1} exists then
            C_{i+1}.previous ← C_i
            put C_{i+1} onto the stack S
        end if
    end for
end for
```

5 Results

The algorithm was tested nearly on the 250 real ligands downloaded from the protein database[1]. The number of atoms in ligands used for the testing varied from 5 to 168. The starting point for each ligand was chosen by a user to satisfy conditions described in the section 4.3. Using binomial search we determined the smallest circle for which the algorithm would still return positive answer on question whether the ligand would pass through it or not. The radius of the smallest circle was then compared with the radius of the bounding sphere. The final results are shown in Figure 12. According to these results we are able to find a path through a 40% (in average) narrower channel in comparison with the bounding sphere approach.

The number of steps needed to compute the path of a ligand from one side of the circle to the other one is shown in the next graph (Figure 13). Our experiments show that

[1] http://www.rcsb.org/pdb/home/home.do

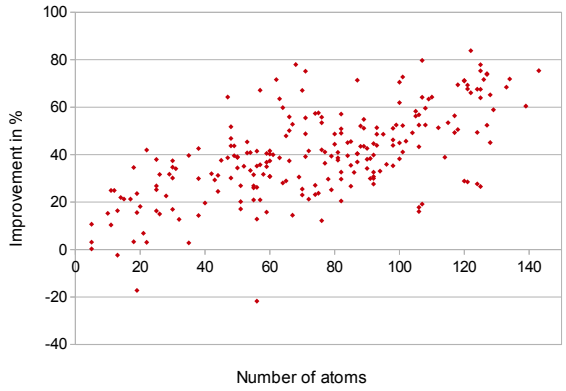

Figure 12: The improvement of the channel radius in comparison to the bounding sphere method.

the average time complexity for the ligands used for testing tends to be between $O(n\ ln\ n)$ and $O(n\ ln^2\ n)$ where n denotes the number of atoms in ligand.

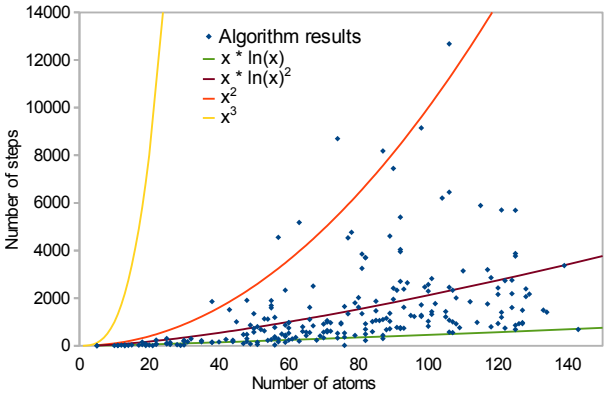

Figure 13: The number of steps needed to compute the path of the ligand through the smallest possible gap represented by a circle.

As it can be seen in Figure 12, our algorithm has some limitations. In some cases the method provides the worse solution than the bounding sphere method (see the points with negative values). These cases occur due to the fact that in our implementation the circle is using only a simple hard-coded sequence of a translation followed by a rotation. The collision detection on the molecule structure can then, in particular cases, prevent the circle from switching from one configuration to another even though the different sequence of movements would allow it.

We believe that this problem can be removed entirely by implementing a more complex system of movements. The proposed algorithm, however, would still not be optimal in the sense of minimizing the circle radius. The radius of the minimal circle found by our algorithm is limited by the distance of the connected atoms (see Figure 14).

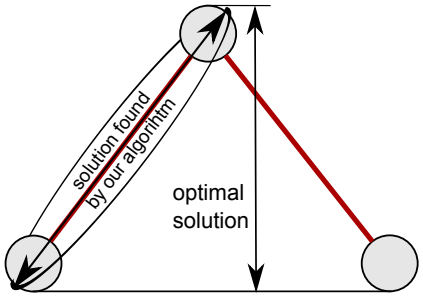

Figure 14: The limitations of the presented algorithm.

6 Conclusion

We have described a simple and fast algorithm based on the motion path planning that computes a path of a set of spheres through a circular-profile shaped gap. Three modes of the algorithm was purposed. Each of them solving part of the given problem according to the complexity of the substrate in the specific area. It was shown that the purposed algorithm has the average time complexity somewhere between $O(n\ ln\ n)$ and $O(n\ ln^2\ n)$ where n denotes the number of atoms in ligand and can find a path through a 40% (in average) narrower channel in comparison with the bounding sphere approach.

Acknowledgements

I would like to thank to my colleague Mgr. Petr Tobola, Ph.D for his original idea which is the base of my diploma thesis and this paper.

References

[1] Maciej Haranczyk and James A. Sethian. Navigating molecular worms inside chemical labyrinths. *Proceedings of National Academy of Science (PNAS)*, pages 21472–21477, 2009.

[2] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[3] Peter Medek, Petr Beneš, and Jiří Sochor. Computation of tunnels in protein molecules using delaunay triangulation. *Journal of WSCG*, pages 107–114.

[4] Amit P. Singh, Jean-Claude Latombe, and Douglas L. Brutlag. A motion planning approach to flexible ligand binding. *ISMB-99 Proceedings*, pages 252–261.

[5] Guang Song and Nancy M. Amato. A motion-planning approach to folding: from paper craft to protein folding. *Robotics and Automation, IEEE Transactions on*, 20(1):60 – 71, feb. 2004.