Incremental Building in Peptide Computing to Solve Hamiltonian Path Problem

Muthiah Sakthi Balan and Parameswaran Seshan

E-Comm Research Lab
Infosys Technologies Limited
Bangalore - 560100, India
{sakthi_muthiah,parameswaran_seshan}@infosys.com

Abstract. To solve intractable problems using biomolecules the model requires exponential number of the same. As a proof of concept this model is acceptable, but when it comes to realization of the model the number of biomolecules needed should be drastically reduced by some techniques. In this work we address this issue for peptide computing – we propose a method called incremental building to reduce the number of peptides needed to work with for solving large combinatorial problems. We explain this model for solving the Hamiltonian path problem, analyze the space and time complexity for the same, and also discuss this method from the perspective of molecular computing as a whole and study its implications.

1 Introduction

One of the primary objectives to look for new computational models is to solve difficult combinatorial problems, called intractable problems, at an exponentially faster rate by using the massive parallelism inherent in bio-computing models. Many models such as DNA computing [1], P-systems [11] and peptide computing [9] have been proposed to solve some of the difficult problems.

In most of the bio-computing models the system works by exploring all possible ways by performing exhaustive search in parallel and by finding out the correct solution from the pool of possible solutions after performing a constant number of bio-steps. Since these models can have several copies of bio-molecules or cells, it has the flexibility to explore all possible combinations simultaneously. This helps the model to arrive at the solution, if any, at a rate exponentially faster than silicon computer models.

Peptide computing is a computing model that considers the interaction between peptides and antibodies as a computational operation and builds on the massive parallelism present in it to solve various NP-complete problems in an exponentially faster way. For solving hard combinatorial problems this computing model works as follows:

1. Formation of peptide sequences where each one represents a possible solution for the problem. For example, to solve the satisfiability problem in [9],

peptides are so formed, that each sequence represents one possible truth value for any Boolean formula over n variables. Hence, we form 2^n different peptide sequences. Note that the formation of peptide sequences does not depend on the specific input.

2. Antibodies are chosen to eliminate those sequences that do not represent the solution for the given problem. After all the elimination methods are done, if we are left with some peptide sequences then the answer will be *yes* to the decision problem, otherwise the answer will be *no*.

It is to be noted that in DNA computing, peptide computing, or P-system, we explore all possible solutions of the problem, which is already exponential in terms of its input size, and, for that to be carried out the system needs exponential number of biomolecules or cells. Even if we assume that the exponential number of molecules can be synthesized artificially, the sum of the molecular weights of the biomolecules involved in the process, would be too big to be handled in a wetlab [7].

When we examine the literature, we see that it is not only that exponential number of different bio-molecules or cells is required, but also each bio-molecule or cell should have multiplicities. When we mention about sequences, we are always talking about multiple number of the same bio-molecules or cells. Thus, a multi-set is involved in these models. We need non-determinism and parallelism in the model to compute hard problems quite efficiently.

Recently, it has been shown that a bacterial computer can be successfully used to solve the Hamiltonian Path Problem for a graph [3], by encoding gene segments to represent the graph and allowing a large number of E-coli bacteria to take those segments inside of themselves to automatically form permutations of genes. Since a large number of such bacteria is involved and given that they can self-reproduce, this gene combination activity happens in each bacterium in parallel with other bacteria, giving us benefits of parallelized processing. Some such random permutations of the DNA segments result in gene sequences that represent Hamiltonian paths in the graph - the bacteria that eventually gets such sequences inside them appear as emitting a distinct detectable fluorescence. This computing model also uses exhaustive search.

Another important point to mention in bio-computing in general is, the formation of specific bio-molecules is a pre-processing step. So the model requires that all bio-molecules, that are exponential in number, are available in order to start the processing. Therefore, when we look from the top level this computing model consists of three primary steps (see Figure 1): (i) pre-processing (can also be called as encoding) (ii) processing, and (iii) finding output, if any, and decoding it.

In this paper we first address the above issues with respect to peptide computing. The main issue here is the need for exponential number of peptide sequences even before the processing takes place. We address this problem by proposing a method called incremental building. In this method, the peptide sequences are not formed a priori unlike the older models in peptide computing, but only when it is required. We start with some part of the peptide sequence and build

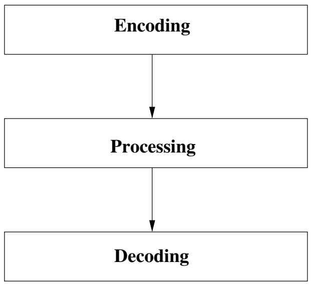


Fig. 1. Bio-computing Model

it incrementally as and when it is required. For this to happen unlike making pre-processing and processing as two separate steps, we do them in turns, i.e., alternatively. We first do a part of encoding in peptide sequences and then process them to see if they will lead to a solution, if yes then we will continue with them, or else we reject them. In the next step, with the selected peptide sequences from the first step, we add/append more peptide sequences and do the processing again to select only those peptide sequences that might lead to a solution. This sequence of encoding and processing is continued for a finite number of steps usually in linear number of steps with respect to the input size of the problem. This we call as the incremental building model. We also feel that this method can be extended to other bio-computing models and this will greatly reduce the number of bio-molecules needed for processing – this amounts to doing pre-processing and processing in turn for a finite number of steps and then decoding (see Figure 2).

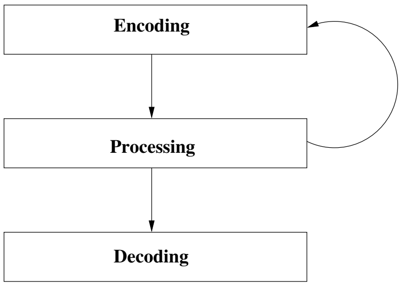


Fig. 2. Incremental Building Model

In this incremental approach the system builds the solution incrementally in an automated fashion unlike the method presented in [2] wherein, we need to form specific peptides to represent a possible solution. Even though the building of sequences that represent possible solution (Hamiltonian paths) in the

Adleman's experiment [1] and bacterial computing [3] occurs in an automated fashion, it is not done in an incremental way. Hence they suffer from explosion in number of molecules needed for processing. Incremental way provides an option of continuing with some molecules or to reject some molecules from further processing. This option is not present in the older models.

In the next section, we briefly look into the peptide computing model and the way it solves the Hamiltonian path problem which is presented in [2]. Since our paper is mainly based on the HPP we present briefly the model presented in [2]. In Section 3, we propose our new method called *incremental building* to reduce the number of molecules needed in the pre-processing and processing steps in order to solve Hamiltonian path problem. In the penultimate section we discuss the feasibility of our model with respect to present day techniques in bio-chemistry. Our paper concludes with the Section 5.

2 Background and Preliminaries

In this paper, a sequence will always mean a peptide sequence unless otherwise stated. Sequences are denoted in small-case letters, like for example p_1, p_2 and so on and antibodies are denoted by upper-case letters like A, B and so on. If A is an antibody we denote the affinity of antibody as $aff(A)$. If p_1 and p_2 are two sequences then p_1p_2 denotes the concatenation of two sequences p_1 and p_2.

2.1 A Brief Look into the Existing Models

In this section, we examine the pre-processing and processing steps in the existing bio-computing models closely.

As mentioned earlier in bio-computing models that are used for solving combinatorial problems involves three steps: first step is a pre-processing step or encoding step, then secondly, the processing step and lastly, the decoding step.

The pre-processing step consists of the preparation of bio-molecules of specific sequences or structures which altogether constitute the solution space of the given problem. The processing step explores all possible molecules constructed in the pre-processing step in search of specific molecules that represent the solution for the problem, if there is any solution for it. The main part of the processing step is the elimination of bio-molecules that does not represent the solution for the given problem. The output step is the detection part to see if there are any molecules left in the pool after all the elimination steps. Since we are concerned in this paper in reducing the number of biomolecules needed, we concentrate only on the first two steps.

When we solve an instance of a hard combinatorial problem using bio-molecular computing, all possible solutions of the problem have to be represented. And for this to happen there is a need for a pre-processing step where specific biomolecules are prepared and put in a pool to start the processing step. Since the number of possible solutions is exponential, the number of biomolecules will also be exponential. And all the more, we also need many copies of the biomolecules. This all adds to a huge number of biomolecules and makes the existing methods almost impossible, even when the input size is slightly larger, to implement in a wetlab [7].

Once we have this pool of exponential number of biomolecules then using molecular biology techniques the processing starts by eliminating those biomolecules which do not represent exact solutions for the problem. Hence, we note here that once the processing starts it is assumed that all the biomolecules representing all possible solutions are present. This means that we have all the biomolecules a priori before starting the processing. Another issue to be noted here is in most of the cases almost all, except very few, biomolecules are going to be eliminated when processing.

In the next two subsections we briefly describe two things – first, the peptide computing model and second, the method for solving Hamiltonian path problem that is presented in [2].

2.2 Peptide Computing

Peptide computing was first proposed by H. Hug and R. Schuler in [9] where they presented a method to solve Satisfiability problem [6] using the natural interactions between peptides and antibodies.

Peptide is a short sequences of proteins consisting of sequence of amino acids attached by peptide bonds. A peptide consists of recognition sites called *epitopes* for the antibodies to bind on it. A peptide can contain more than one epitopes for the same or different antibodies and for each antibody which attach to a specific epitope there is a binding power associated with it called as *affinity*. If the sites for two or more antibodies overlap in the given peptide, then the antibody with more affinity always gets the higher priority to bind to its epitope.

The process of binding of antibodies to specific sites and removal of antibodies by another antibody that binds to a site with a higher affinity[1] resembles the action a Turing machine where at some particular state the head reads/(re)writes a symbol according to some specified rules. This exemplifies the fact that there is some computation taking place when peptides and antibodies interacts. Another fact is that since the process works with several copies of peptides and antibodies it has massive parallelism and non-determinism in it. These facts provides the flexibility to solve some intractable problems efficiently.

2.3 Solving Hamiltonian Path Problem – Previous Method

Let $G = (V, E)$ be a directed graph. Let $V(G) = \{v_1, v_2, \cdots, v_n\}$ be the vertex set of the graph and $E(G) = \{e_{ij} \mid v_j \text{ is adjacent to } v_i\}$ be the edge set. Without loss of generality we take v_1 as the source vertex and v_n as the end vertex. Our problem is to test whether there exists a Hamiltonian path between v_1 and v_n. We assume that m is the number of edges in the graph G.

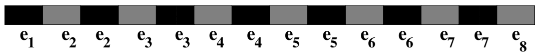
[1] This is basically an immune reaction.

First step is the pre-processing step that involves formation of peptides and antibodies.

For each vertices $v_1, v_2, \ldots, v_n$ we choose an epitope $e_1, e_2, \ldots, e_n$ and these epitopes are arranged in the peptide sequence in such a way that they represent a possible Hamiltonian path on a graph over n vertices. For example, one such specific peptide sequence P will be

$$P = e_1 e_2 e_2 e_3 e_3 \ldots e_{n-1} e_{n-1} e_n$$

Also see Fig. 3, for one such peptide sequence over 8 vertices.



e_1 e_2 e_2 e_3 e_3 e_4 e_4 e_5 e_5 e_6 e_6 e_7 e_7 e_8

Fig. 3. Peptide sequence over 8 vertices

Likewise if we permute the set of epitopes $\{e_2, e_3, \ldots, e_{n-1}\}$ with e_1 and e_n as two extremes then we get $(n-2)!$ peptide sequences. Note that this pre-processing step does not depend on the graph G.

In the next step of the pre-processing step we form three set of antibodies. The first set of antibodies that we call as A antibodies are formed for each edge present in the graph G. These antibodies binds to the subsequence $ep_i ep_j$ in the peptide sequence if there exists an edge $v_i v_j$. The second set of antibodies called B antibodies binds to all subsequences $ep_i ep_j$ where $v_i v_j$ is not an edge in the graph G. The third set of antibodies C consists of only one labelled or colored antibody that binds to the whole of peptide sequences. If they bind to some peptide sequence then this can be seen by the emitted fluorescence as a result of this binding. The affinity of the antibodies are defined as follows:

$$aff(A) < aff(C) < aff(B).$$

The algorithm is:

Algorithm 1

1. *Take all the peptide sequences formed in an aqueous solution.*
2. *Add A set of antibodies to the collection.*
3. *Add B set of antibodies to the collection.*
4. *Add C set of antibodies to the collection.*
5. *If fluorescence is detected then there exists a Hamilton path in the graph G, otherwise there exists no such path. If the peptides are bound to an addressed chip the solution can be immediately read.*

It should be easy to note that even a single presence of a B antibody binding to a subsequence in a peptide sequence denotes that the path it corresponds is not a valid path in the given graph G. Hence the valid paths are the ones where the corresponding peptide sequences that contain only A antibodies binding to it. When we add the C set of antibodies since C has more affinity than A, C will remove the antibodies A and binds to that whole peptide sequence. With respect to antibodies B, C has lesser affinity, so C can not bind to those peptide sequences where there are one or more antibodies B binding to the sequences. Hence C selects all the Hamiltonian paths of the graph G, if at all it is present there.

A quick analysis will show the following resource complexity – (i) number of peptides is $(n-2)!$, (ii) length of peptides is $\mathcal{O}(n)$, and (iii) number of antibodies is $\mathcal{O}(n^2)$.

Hence we note that this model requires $(n-2)!$ peptide sequences before the start of the processing, because we assume that any of the $(n-2)!$ sequences might lead to a solution without even considering the input of the graph G.

It should also be clear to see that there are three separate steps – one encoding step, next the processing step and the last one the decoding step. For more details on this model please refer [2].

In the next section we describe how we build the peptide sequences only when it is required and reduce much of the elimination methods which in turn reduces the number of peptide sequences needed on the whole.

3 Incremental Building of Peptide Sequences

In this section we present our proposed model called incremental building of peptide sequences for solving Hamiltonian path problem.

As mentioned earlier, our main motivation here is to reduce the number of peptide sequences. Unlike the previous methods where we first assumed every possible solution might be a solution for the given problem and start the elimination process, in this method we build the required solution in an incremental way.

Let us suppose the problem statement for Hamiltonian path problem is given as in the previous section (see Section 2.3).

First we explain the pre-processing step in the following.

Pre-Processing Step

For each vertices $v_1, v_2, \ldots, v_n$ we take an unique peptide sequence. Let us denote these peptide sequences as $p_1, p_2, \ldots, p_n$. Each peptide sequence p_i for $2 \leq i \leq n-1$ is of the form (see Figure 4):

$$p_i = pre_i x_i suf_i$$

where suf_i and pre_i denote the suffix and prefix part of the sequence p_i and x_i is a random sequence. p_1 and p_n are represented as $x_1 suf_1$ and $pre_n x_n$ respectively.

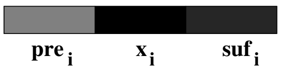


Fig. 4. Peptide sequence p_i

Hence the sequence p_1 is devoid of the prefix part and the sequence p_n the suffix part.

In the next step, we form antibodies. For each edge $e = v_i v_j, 1 \leq i, j \leq n$ and $i \neq j$, in the graph G we form antibodies A_{ij}. The epitope of A_{ij} is defined as $suf_i pre_j$. Let us denote this set of antibodies as $\mathcal{A}$.

We also form another set of labelled antibodies denoted as $\mathcal{L}$. The set $\mathcal{L}$ consists of antibodies L_i for each $v_i \in V(G)$. The antibody L_i binds to the subsequence x_i of the peptide sequence p_i.

Incremental Building

In this step we always have two sets of peptide sequences one called as *source*, denoted as S, and the other one called as *target*, denoted as T. First, the set S consists of p_1 and the set T will always be the set of all peptide sequences $\{p_2, p_3, \ldots, p_n\}$.

In the sequel we describe the steps. Please refer Algorithm 2 for step-wise description. A note on the implementation of these steps is presented in the Section 4.

First we take the source set $\{p_1\}$ in an aqueous solution. We add the target set T to it followed by the set of antibodies A. The epitope is split across two peptide sequences. This enables that, when antibodies attach to their epitopes it actually creates a link between the two peptide sequences p_1 and p_i. To be more clear, an antibody A_{1i} binds to its epitopes $suf_1 pre_i$ and creates a link between two separate peptide sequences p_1 and p_i thus forming a bigger peptide sequence consisting $p_1 p_i$. It is noted that the link between two sequences happens only when there is an edge e_{1i} in the given graph G (see Figure 5).

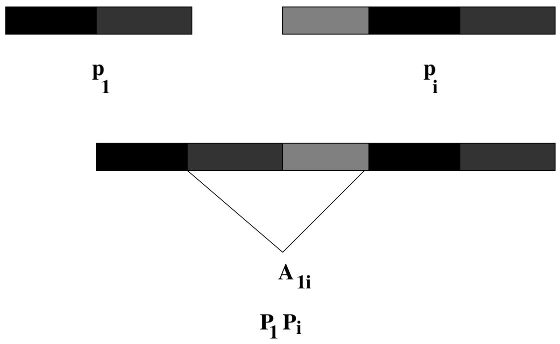


Fig. 5. Two peptide sequences p_1 and p_i linked together by the antibody A_{1i}

After this linking happens, all the antibodies A_{ij} and peptide sequences p_k that are simply floating around are filtered out. Therefore we are left with only those sequences of the form $p_1 p_j$ $(2 \leq j \leq n)$ where $e_{ij} \in E(G)$ – this set of sequences is denoted by $S^{(1)}$. After the end of this step, which we call as the first iteration, the set $S^{(1)}$ becomes the source set.

In the first step of the next iteration we add the target set T and the set of antibodies A to the aqueous solution containing $S^{(1)}$. This facilitates further elongation of the peptide sequences since antibodies A_{ij} will again link one peptide sequence from $S^{(1)}$, say $p^1 = p_1 p_i$, with a peptide sequence from T, say p_j, provided $e_{ij} \in E(G)$. At the end of this step we filter out all the floating antibodies A_{ij} and peptides p_k.

After repeating these steps $n-1$ times, if we end up with a sequence of length $n-1$ then it shows that there exists a Hamiltonian path for the graph G. If not, then there is no such path in the graph G.

When we do these iterations we might end with a sequence like $p_1 p_{i_1} p_{i_2} p_{i_3} p_{i_1}$ where $2 \leq i_1, i_2, i_3 \leq n$. We should not allow this sequence to be elongated at the next step because this has repetition of a vertex, namely v_{i_1}. To take care of these situations, we use the set of antibodies $\mathcal{L}$. The set $\mathcal{L}$ consists of antibodies L_i for each $v_i \in V(G)$. The antibody L_i binds to the subsequence x_i of the peptide sequence p_i. At the end of each iteration we add the set of antibodies $\mathcal{L}$ to the source set. If any sequence has two labelled antibodies, that can be inferred with a detection of two fluorescence emitting antibodies, then it is filtered out. One way of doing this is, each labelled antibody can be given a color (this is normally done in experiments in bio-chemistry) and through the emitted fluorescence we can find out the duplication of the vertices.

Moreover, again, at the end of i^{th} $(1 \leq i \leq n-1)$ iteration we check if the length of the peptide sequences are of length i or not. If they are not of length i then they are also filtered out. Since at the end of i^{th} iteration if the length of the peptide sequence is not of length i then it shows that it can not be further elongated and it will not lead to a Hamiltonian path for the graph G.

The complete algorithm is given below:

Algorithm 2

1. *Take the source set of peptide sequences S as the set $\{p_1\}$ in an aqueous solution;*
2. *Set the counter $i = 1$;*
3. *Add the set of peptide sequences T to the solution;*
4. *Add the set of antibodies $\mathcal{A}$ to the solution;*
5. *Filter-out all the peptide sequences p_j and antibodies A_{ij} that are simply floating around the solution;*
6. *Add the set of antibodies $\mathcal{L}$ to the solution;*
7. *Filter-out all the sequences having a labelled antibody L_i binding to two or more epitopes;*
8. *Filter-out all the sequences of length less than i;*
9. *If $i < n-1$ then $i = i+1$ and go to step 3 with S as the set of remaining peptide sequences in the aqueous solution;*
10. *If there exists a peptide sequence in the solution then there exists a Hamiltonian path or else there is no such path in the graph.*

Discussion on the Incremental Building Model

We analyze the amount of peptides and antibodies needed in the proposed incremental model in the sequel.

If we assume the number of vertices in the given graph G as n and the number of edges in G as m then the amount of peptides needed is $O(n)$. The amount of antibodies needed in the set A is $O(m)$ which is $O(n^2)$. And the amount of antibodies needed in the set L is $O(n)$. Hence the total amount of antibodies needed is of the order of n^2.

In the following we discuss the main differences between the previous model [2], that we call as the old model, and the incremental model.

1. In the old model we needed $(n-2)!$ peptide sequences but here we need only n peptide sequences that can be linked using antibodies to form larger sequences.
2. In the old model we had only constant number of bio-steps but here the number of bio-steps needed is of the order of n, the number of vertices.
3. In the old model we need to form specific peptide sequences to denote all possible paths of any graph over n number of vertices. In this incremental model the algorithm itself builds the paths automatically using antibody as a link.

4 Remarks

This paper is mostly based on the assumption that two peptide sequences can be combined together using an antibody that acts a linker. In this section we ask the following question and survey some work that has been done in this context.

Is it possible for two amino acid sequences to form a single linear sequence upon binding to a target protein?

Any two peptides can bind to a target protein in a vectorial fashion. In order to combine two separate amino acid sequences in to a single one upon binding to a target protein, those two peptides should follow certain conditions:

1. Both peptides should come together in close proximity to each other upon binding to a target protein.
2. Binding sites for these peptides should be distinct with no overlap.
3. Binding can be head-tail, tail-head or tail-tail fashion (N and C-terminal of a peptide denotes, head and tail respectively).
4. The binding of one peptide may or may not affect the binding of the other (often refereed as allosteric vs independent binding).
5. The conformation of the peptide may or may not change upon binding.
6. The binding constant between the peptides and the target protein should be in nano-molar range, so that both the peptides would be in bound-form for an extended period of time.
7. Finally, the interface of these bound peptides may create a unique key that might fit perfectly into the lock of the target protein, implying the symbiotic nature of stability of individual peptides.

The antigen binding region of the antibody is composed of hetero-dimer of Heavy chain (VH) and Light chain (VL) regions. Specificity of the antibody is attributed to the complementarity determining region (CDRs) in Heavy and light chains. It has been shown that the in vitro recombination of heavy chain and light chain binds to the antigen, albeit with low-affinity [8]. Close proximity of these two different polypeptide chains have been confirmed by the addition of linker between the two [10]. The affinity was drastically improved upon stabilizing both VH and VL.

First breakthrough in antibody design also featured linking the two different polypeptide chains with a short linker of 6 amino acids [4]. Also, the length of the linker between the polypeptide affects the stability of the complex [5]. Finally, a universal Linker theory [12] has also been proposed in order to create an antibody that can bind to two different epitopes of an antigen. The theory suggests that linking the polypeptide fragment A and B via a flexible linker drastically improves the binding to the target protein. The length of the linker depends upon the distance between the non-overlapping epitopes. It suggests that the any two polypeptides can arrange into a single polypeptide sequence with certain reservation.

5 Conclusion

We proposed a new method called incremental building for peptide computing wherein instead of building all the peptide sequences in the pre-processing step we build it incrementally only after checking if it will lead to a possible solution or not. We saw that this method reduces the number of peptide sequences required in the pre-processing step. This method, we feel, can be applied to other computing models using bio-molecules. This will have a good impact on reducing the number of bio-molecules needed for the processing since the building of further bio-molecules, like extending peptide sequences, is done only after checking the feasibility of getting to a solution. Another interesting question will be – what set of problems this method can solve.

Acknowledgments

The authors wish to thank Dr. Saravanakumar Narayanan from the Wagner Lab, Harvard Medical School for his inputs on peptide-antibody interactions.

References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024 (1994)
2. Balan, M.S., Krithivasan, K., Sivasubramanyam, Y.: Peptide computing: Universality and computing. In: Jonoska, N., Seeman, N.C. (eds.) DNA 2001. LNCS, vol. 2340, pp. 290–299. Springer, Heidelberg (2002)
3. Baumgardner, J., Acker, K., Adefuye, O., Crowley, S., DeLoache, W., Dickson, J., Heard, L., Martens, A., Morton, N., Ritter, M., Shoecraft, A., Treece, J., Unzicker, M., Valencia, A., Waters, M., Malcolm, A., Heyer, L., Poet, J., Eckdahl, T.: Solving a hamiltonian path problem with a bacterial computer. Journal of Biological Engineering 3(1), 11 (2009)

4. Bird, R.E., Hardman, K.D., Jacobson, J.W., Johnson, S., Kaufman, B.M., Lee, S.M., Lee, T., Pope, S.H., Riordan, G.S., Whitlow, M.: Single-chain antigen-binding proteins. Science 242(4877), 423–426 (1988)
5. Blenner, M., Banta, S.: Characterization of the 4D5Flu single-chain antibody with a stimulus-responsive elastin-like peptide linker: A potential reporter of peptide linker conformation. Protein Science 17, 527–536 (2008)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability A Guide to the Theory of NP-Completeness. W.H.Freeman and Company, New York (1979)
7. Hartmanis, J.: On the Weight of Computation. Bulletin of the EATCS 55, 136–138 (1995)
8. Hudson, N., Mudgett-Hunter, M., Panka, D., Margolies, M.: Immunoglobulin chain recombination among antidigoxin antibodies by hybridoma-hybridoma fusion. J. Immunol. 139, 2715–2723 (1987)
9. Hug, H., Schuler, R.: Strategies for the developement of a peptide computer. Bioinformatics 17, 364–368 (2001)
10. Huston, J., Levinson, D., Mudgett-Hunter, M., Tai, M.S., Novotny, J., Margolies, M., Ridge, R., Bruccoleri, R., Haber, E., Crea, R., Oppermann, H.: Protein engineering of antibody binding sites: Recovery of specific activity in an anti-digoxin single-chain Fv analogue produced in Escherichia coli. Proc. Natl. Acad. Sci. 85, 5879–5883 (1988)
11. Păun, G.: Computing with membranes–A variant: P systems with polarized membranes. Intern. J. of Foundations of Computer Science 11(1), 167–182 (2000)
12. Zhou, H.X.: Quantitative account of the enhanced affinity of two linked scFvs specific for different epitopes on the same antigen. J. Mol. Biol. 329, 1–8 (2003)