The Biomolecular Computation Paradigm: A Survey In Massive Biological Computation

Georgios Drakopoulos[1], Dimitrios Tsolis[2], Antonia Stefani[3], and Phivos Mylonas[1]

[1] Department of Informatics, Ionian University
Plateia Tsirigoti 7, Kerkyra 49100, Hellas
{c16drak, fmylonas}@ionio.gr
[2] Department of Cultural Heritage Management and New Technologies
University of Patras
G. Seferi 2, Agrinio 30100, Hellas
dtsolis@upatras.gr
[3] Hellenic Open University
Tsamadou 13-15, Achaia 26222, Hellas
stefani@eap.gr

Abstract. Biomolecular computation is the scientific field focusing on the theory and practice of encoding combinatorial problems in ordinary DNA strands and applying standard biology lab operations such as cleansing and complementary sequence generation to them in order to compute an exact solution. The primary advantage offered by this computational paradigm is massive parallelism as the solution space is simultaneously searched. On the other hand, factors that need to addressed under this model are the DNA volume growth and computational errors attributed to inexact DNA matching. Biomolecular computation additionally paves the way for two- and three-dimensional self assemblying biological tiles which are closely linked at a theoretical level to a Turing machine, establishing thus its computational power. Applications include medium sized instances of TSP and the evaluation of the output of bounded fan-out Boolean circuits.

Keywords: biomolecular computation, DNA computation, computing paradigm, computational media, parallel computing, CREW PRAM, TSP, Boolean circuits, nondeterministic Turing machine

1 Introduction

The seminal paper [19] is widely known for essentially establishing the field of quantum computing. However, a lesser known offshoot is *biomolecular computing*, alternatively known as *DNA computing*. The founding notions of this computational paradigm were presented in [2], where it was proposed that regular DNA strands can represent combinatorial inputs instead of the functions of a living organism. Then standard lab operations can be applied to these strands in order to extract strands containing a solution. As a concrete application, the

TSP was solved exactly through a brute force methodology in a medium sized graph $G = (V, E)$ in $\mathrm{O}(|V| + |E|)$ elementary operations and $\mathrm{O}(\log |V|)$ DNA strands, indicating the potential of massive parallelism. This was repeated in [32] with a different algorithmic approach though in the sequence of lab operations, establishing the fact that novel and efficient algorithms are also necessary in this computational paradigm.

The primary objective of this survey is to concisely summarize the principles and notions of the paradigm of biomolecular computing with an emphasis on the potential for massively parallel computations. The latter may well serve in the dawn of the big data and 5V era as an unconventional inspiration for the designers of parallel algorithms or distributed systems.

The remaining of this survery is structured as follows. Section 2 summarizes the principal concepts of biomolecular computing, the connections to known computational models, and describes computational applications not examined elsewhere in this survey. The elementary operations, advantages, and disadvantages of the biomolecular paradigm are explained in section 3. The most important application, namely TSP, is described in 4. The parallelism potential is explored in section 5, while factors working against the computation scale up are investigated in sections 6 and 7. The main points are summarized in 8 and certain conclusions are drawn. Finally, table 1 summarizes the survey notation.

Table 1. Survey notation.

Symbol	Meaning
$\triangleq$	Definition or equality by definition
$\lceil s \rceil$, $\lfloor s \rfloor$	DNA strand s in 5-3 and 3-5 direction respectively
$\bar{s}$	Complementary DNA strand of s in 5-3 direction
$\lvert s \rvert$	Length (number of bases) of DNA strand s
$s_1 \parallel s_2$	Strands s_1 and s_2 match completely
$s_1 \sqsupset s_2$	Strand s_2 matches to the right side of s_1
$s_1 \sqsubset s_2$	Strand s_2 matches to the left side of s_1
$s_1 \sqcup s_2$	Strand s_2 matches to a middle segment of s_1
$s_1 \nparallel s_2$	Strands s_1 and s_2 do not match
$s_1 \cap s_2 = k$	Strands s_1 and s_2 overlap in k bases
$\lvert S \rvert$	Cardinality of set S
$\mathrm{E}[X]$	Mean value of random variable X
$\mathrm{Var}[X]$	Variance of random variable X

2 Previous Work

As stated earlier, the groundwork for the paradigm of biomolecular computing was laid in [2] followed by [32]. Soon, notions and applications emerged as

noted in the early surveys [45], [40], and [47]. The computational power of the paradigm is explored in [9] and [6] and its limits in [18]. The enormous potential for parallelism is highlighted in [21], [43], [23], [24], and [22]. The notion of self assembly was applied to this paradigm in [1], [48], and [31]. According to this principle, which is reminiscent of non-supervised learning, DNA tiles in a test tube given proper mobility conditions and time can attach themselves to tiles or strands containing fully or partial complementary sequences without human intervention in two [51][50] or three dimensions [28]. Issues pertaining to complexity of the biomolecular paradigm are examined from various viewpoints in [4], [46], and [29].

Another way to examine the potential and the complexity of biomolecular computation is through the simulation of the operation of sequential, bounded fan-in Boolean circuits with DNA strands as first shown in [3] and [36] and described in detail in the follow up work [39]. Questions regarding the complexity of constructing and evaluating the output of such circuits are addressed in [35], [37], and [34], whereas self assembling circuits are investigated in [38].

Among the algorithmic applications of biomolecular computation are the brute force parallel solution of k-SAT in [13] and in [5], dynamic programming on the Cell Matrix architecture [49], and splicing systems [11]. Shortest path algorithms implemented in biomolecular elementary operations are presented in [33] and in [42]. Length bounded computing with DNA strands and its connections to space complexity are explored in [20]. An evolutionary algorithm also expressed these operations is described in [12].

Finally, steps regarding the implementation of a DNA computer are given in [26], [41], [44], and [25], although these proposals vary. DNA operations can be also simulated over Neo4j with properties in edges corresponding to physical or chemical DNA properties in an approach similar to the one presented in [30] for implementing persistent data structures. Concerning software, a fully functional graphical computing environment for biomolecular computation is described in [10]. It also includes DNA C, a C variant which is a combination of the C constructs pertaining to integers with an extension implementing the fundamental operations of biomolecular computing. The exclusion of floating point arithmetic should not come as a surprise, since biomolecular computation is discrete in nature and has been so far applied only in combinatorial problems. Nonetheless, should the need arise, floating point numbers can be approximated fairly well by rationals or by continued fractions. For instance, the golden ratio φ is represented by the infinite fraction[4]

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}} \tag{1}$$

[4] OEIS sequence A000012.

3 Paradigm Notions

3.1 Definition

Knowledge transfer and inspiration between computer science and biology has been fruitful. This relationship has already resulted in bioinformatics, connectomics, and computational biology. One of them, with direct reference to the everyday laboratory handling of DNA strands, is the paradigm of biomolecular computation which can be formally defined as

Definition 1. *Biomolecular computing is the art and science of using DNA strands as computational medium to appropriately encode candidate solutions to (possibly intractable for a Turing machine) combinatorial problems and using standard biological laboratory techniques in order to select an exact solution.*

From definition 1 follows that any DNA encoding corresponds strictly to candidate solutions of a combinatorial problem and not to the design and functions of a living being. Also, the difference from the fields of bioinformatics and computational biology should be clear. Although the inspiration, terminology, and implementation are biological, the paradigm is definitiely computational as the objective is to codify and efficiently solve instances of intractable, at least under the conventional Turing machine model, combinatorial problems.

3.2 Abstract DNA Operations

Under the biomolecular computation paradigm the elementary operations applied to the DNA strands in the test tube are the follwing.

- **Initialize**(T_0): Crate a test tube T_0 containing each admissible candidate solution for a combinatorial problem according to a probability distribution, usually the uniform one.
- **Copy**(T_0, T_1): Copy the contents of T_0 to the tube of T_1.
- **Merge**(T_0, T_1): Mix the contents of T_0 and T_1 to T_0.
- **Detect**(T_0): Examine whether T_0 is empty.
- **Select**(T_0, s_0): Extract s_0, if present, from T_0.
- **Extract**(ℓ): Extract strands of length ℓ.
- **Cleavage**(s_0, σ_0): Slice s_0 according to the shorter template strand σ_0.
- **Anneal**(s_0): Create double DNRA strands from a single one.
- **Denature**(s_0): Create single DNA strands from a double one.
- **Ligate**(T_0): Create bonds between double strands inside tube T_0.

The basic storage unit in this paradigm is the test tube which may contain a fixed volume of DNA strands. The latter are not necessarily of the same type. In fact, the opposite is quite common as the tube contains the results of a sequence of lab operations applied to a set of candidate solutions. It is only after the end of these operations where the solution, if any, to the instance at hand is extracted and cultivated that a test tube may contain only copies of a single strand.

4 TSP

Perhaps the most well known biomolecular algorithm is the one shown in algorithm 1 for solving TSP for a graph $G = (V, E)$. Notice this is a brute force method which can be used among others to discover community structure in graphs as in [15] and [27]. However, this approach could not be a basis for heuristic techniques such as those in [16] and [17]. The primary characteristic of algorithm 1 is that all paths of length j are created in j steps. Assume that each vertex is encoded with b_v and each edge with b_e bases. Each vertex v_k has a unique coding s_k in DNA bases, while the edge (v_i, v_j) is encoded as the concatenation of $\bar{s_i}$ and $\bar{s_j}$.

Algorithm 1 TSP expressed in biomolecular operations

Require: tubes T_0, T_v with encoded vertices; tube T_e with encoded edges
Ensure: a Hamilton circle is found
1: **initialize**(T_0)
2: **for** $j \leftarrow 1$ **to** $|V|$ **do**
3: **merge**(T_0, T_e)
4: **merge**(T_0, T_v)
5: **ligate**(T_0)
6: **end for**
7: **extract**($|V|\,(b_v + b_e)$)
8: **denature**(T_0)
9: **return detect**(T_0)

5 Parallelism

Theoretically, the biomolecular computation paradigm can be reduced to simulating a CREW PRAM, a version of RAM with $P > 1$ processors and M memory locations where multiple processors can read the same memory address but only one can write any given time at a given memory address. The CREW PRAM operation set LOAD, READ, WRITE as well as the memory configuration of this machine can be simulated by a sequence of biomolecular operations as shown in [46] through a sequence of successive configurations of length $\mathrm{O}\,(\log\,(PM))$ each. Starting from a random configuration, mixing a sequence of admissible configurations, at the expense of a multiplicatively growing volume, finally yields one superconfiguration of concatenated configurations representing a desired sequence of computation, assuming that one can be found. By repeatedly applying cleavage operations, the individual configurations are extracted.

Another way to quantify the actual parallelism offered by the biomolecular paradigm is to use Amdahl's law as a benchmark

$$\Gamma \triangleq \frac{1}{(1-\gamma_0)+\frac{\gamma_0}{s}}, \qquad 0 \leq \gamma_0 < 1 \tag{2}$$

where Γ is the actual speedup, γ_0 is the parallelizable part of the task, and s is the speedup of that part. Thus, the size of the parallelizable part plays a crucial role and essentially defines how much can be actually sped up as for infinite s

$$\Gamma \to \frac{1}{1-\gamma_0} \tag{3}$$

6 Error Free Computations

6.1 Overview

Errors in the biomolecular computation can be classified into two broad categories. The first source of errors lies in the encoding of candidate solutions as there can be partial DNA matches which may eventually create false final solutions, in other words creating false positives. Moreover, the elementary operations described above may not be executed with absolute sucess due to a number of factors. This might create depending on the nature of the operations involved either false positives or false negatives.

6.2 Encoding

Concerning the solution encoding, an obvious approach might be to choose an encoding with controllable redundancy β_0. Namely, each bit of information is expressed with $\alpha_0 = 1 + \beta_0$ bits in total and on average. Thus, the ratio ρ_0 of original to redundant information is

$$\rho_0 \triangleq \frac{1}{\beta_0} = \frac{1}{\alpha_0 - 1} < 1 \tag{4}$$

For instance, in a graph problem with $|V|$ if each vertex normally requires $\lceil \log |V| \rceil$ bits to encode, then it would require α_0 times as much.

Regarding the number of false negatives or false positives, it can be argued that it is proportional to the partial matches of DNA strings. While there is a case where an intended match of two DNA strands s_1 and s_2 fails, strand mismatches denoted by $s_1 \nparallel s_2$ can be considered accurate for the most part. On the contrary, partial matches either from left or from right are with high probability indicators of failed operations, especially if the two strands overlap in only a few bases. For a single biomolecular step in the same test tube T_0, let

$$u = |s_1 \parallel s_2|$$

$$\begin{aligned} u_r(k) &= |s_1 \sqsupset s_2, s_1 \cap s_2 = k| \\ u_l(k) &= |s_1 \sqsubset s_2, s_1 \cap s_2 = k| \\ u_m &= |s_1 \sqcup s_2| \\ u_o &= |s_1 \nparallel s_2| \end{aligned} \tag{5}$$

denote respectively the number of DNA perfect matches, right and left matches in k bases, middle matches and no matches. Thus, in a single biomolecular step the ratios of true matches Q^+ and true mismatches Q^- are

$$\begin{aligned} Q^+ &\triangleq \frac{u + u_m}{u + u_m + \sum_{k=1}^{|s_2|} u_r(k) + \sum_{k=1}^{|s_2|} u_l(k)} \\ Q^- &\triangleq \frac{u_o}{u_o + \sum_{k=1}^{|s_2|} u_r(k) + \sum_{k=1}^{|s_2|} u_l(k)} \end{aligned} \tag{6}$$

Another way to assess the reliability of a sequence of n_0 biomolecular operations is the following. Let π_k be the success probability of each operation. The success probability π^+ by the product rule, assuming operation independence, is

$$\pi^+ \triangleq \prod_{k=1}^{n_0} \pi_k = \pi_1 \dots \pi_{n_0} \tag{7}$$

Thus, if $\mu_1 \leq \pi_k \leq \mu_2$, then the following bounds can be derived

$$e^{-n_0 \mu_2} \leq \pi^+ \leq e^{-n_0 \mu_1} \tag{8}$$

Perhaps a more accurate way to assess the average value of π^+ is to consider the geometric mean of the sequence

$$\bar{\pi} \triangleq \left(\prod_{k=1}^{n_0} \pi_k \right)^{\frac{1}{n_0}} = \mu_1 \left(\prod_{k=1}^{n_0} \left(1 + \frac{\eta_k}{\mu_1} \right) \right)^{\frac{1}{n_0}}, \quad \pi_k = \mu_1 + \eta_k \tag{9}$$

6.3 Trials

Due to the nature of the actual biological operations, they are not always executed with abolute correctness. This can be attributed to a number of reasons including the age, technology, and condition of lab equipment, the chemistry and quantity of DNA strands themselves, the nature of bonds between strands, and lab conditions such as radiation and electomagnetic pulses of various frequences. As a result, a strand encoding an invalid solution can emerge from the test tubes or a strand containing the solution can be missed in them [8][7].

The geometric distribution models sequences of test outcomes where the probability of success is p_0. Its probability mass function is defined as

$$\text{prob}\left\{X \;=\; k\right\} \triangleq p_0 \left(1 - p_0\right)^k, \quad k \in \mathbb{Z}^+ \tag{10}$$

The interpretation of X is that it models the number of failed attempts of an experiment before the single successful outcome of that experiment. The mean value of this distribution is readily calculated in equation (11).

$$\text{E}\left[X\right] \triangleq \sum_{k=0}^{+\infty} k \,\text{prob}\left\{X \;=\; k\right\} \;=\; \frac{1 - p_0}{p_0} \;=\; e^{\text{logit}(1-p_0)} \tag{11}$$

The $\text{logit}\left(\cdot\right)$ function is the inverse of the sigmoid function extensively used to train deep recurrent and tensor stack networks [14] expresses the logarithm of the odds of a single Bernoulli trial. Also is a special case of a link function to the generalized linear model and leads to the logistic regression, commonly found in deep learning applications.

The variance of X is its mean value scaled by the success probabilty p_0. This is coherent with intuition, since the larger p_0, the fewer attempts are required to achieve success and the number of failed attempts will be close to zero.

$$\text{Var}\left[X\right] \triangleq \sum_{k=0}^{+\infty} \left(k - \text{E}\left[X\right]\right)^2 \text{prob}\left\{X \;=\; k\right\} \;=\; \frac{1 - p_0}{p_0^2} \;=\; \frac{\text{E}\left[X\right]}{p_0} \tag{12}$$

7 Volume Considerations

The major practical limitation of biomolecular computation is the volume necessary to represent each candidate solution. This is limited not only by the volume of the test tube, but also by the encoding which, in turn, implies a redundancy rate to ensure higher operation success probabilites. If V_0 is the hard limit for the test tube volume, α_0 the encoding redundancy factor, n is the number of candidate solutions, and ℓ_0 the solution length, then

$$V_0 \;\geq\; \left(a_0 n \ell_0\right)^{1+\epsilon} \;\Leftrightarrow\; n \;\leq\; \frac{1}{a_0 \ell_0} V_0^{\frac{1}{1+\epsilon_0}} \tag{13}$$

where ϵ_0 is a constant which depends on a number of diverse factors such as lab temperature and tube technology. If a biomolecular computation requires J_0 steps to complete, then the required volume grows exponentially. Then, the above limit should be modified as

$$n \;\leq\; \frac{1}{a_0 \ell_0} V_0^{\frac{1}{J_0(1+\epsilon_0)}} \tag{14}$$

8 Conclusions

This survey explores the foundations, applications, and limits of biomolecular computation which represents an alternative computational paradigm. The latter is based on the handling of potentially very long strands of ordinary DNA using standard biology lab operations such as annealing, generating a complementary strand, selecting a strand, or concatenating two strands. These strands do not codify the inner workings of any living organism. Instead, they contain a suitably selected representation of a computational problem. By selecting an appropriate representation of an input instance, a plethora of various output instances are created by a series of biological operations. Although a fraction of the abovementioned output instances may contain incomplete operations and must be removed by cleansing operations, their majority will contain with high probability a solution.

Acknowledgments

The financial support by the European Union and Greece (Partnership Agreement for the Development Framework 2014-2020) under the Regional Operational Programme Ionian Islands 2014 - 2020 for the project "Smart vine variety selection and management using ICT - EYOINOS" is gratefully acknowledged.

References

1. Adleman, L., Cheng, Q., Goel, A., Huang, M.D., Kempe, D., De Espanes, P.M., Rothemund, P.W.K.: Combinatorial optimization problems in self-assembly. In: STOC. pp. 23–32 (2002)
2. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Nature 369 (1994)
3. Amos, M., Dunne, P.E.: DNA simulation of Boolean circuits. In: Proceedings of 3rd Annual Genetic Programming Conference. pp. 679–683 (1997)
4. Amos, M., Gibbons, A., Dunne, P.E.: The complexity and viability of DNA. Biocomputing and emergent computation: Proceedings of BCEC97 (1997)
5. Baum, E.B., Boneh, D.: Running dynamic programming algorithms on a DNA computer. DNA Based Computers II 44, 77–80 (1999)
6. Beigel, R., Fu, B.: Solving intractable problems with DNA computing. In: IEEE Conference on Computational Complexity. p. 154 (1998)
7. Bijlani, R., Cheng, Y., Pearce, D.A., Brooks, A.I., Ogihara, M.: Prediction of biologically significant components from microarray data: Independently consistent expression discriminator (ICED). Bioinformatics 19(1), 62–70 (2003)
8. Boneh, D., Dunworth, C., Lipton, R.J., Sgall, J.: Making DNA computers error resistant. DNA Based Computers II 44, 163–170 (1996)
9. Boneh, D., Dunworth, C., Lipton, R.J., Sgall, J.: On the computational power of DNA. Discrete Applied Mathematics 71(1-3), 79–94 (1996)
10. Carroll, S.: A complete programming environment for DNA computation. In: Workshop Non-Silicon Comp. NSC-1. pp. 46–53 (2002)

11. Dassen, J.: Molecular computation and splicing systems. Master's thesis, Leiden University (1996)
12. Deaton, R., Murphy, R.C., Rose, J.A., Garzon, M., Franceschetti, D.R., Stevens, S.: A DNA based implementation of an evolutionary search for good encodings for DNA computation. In: International Conference on Evolutionary Computation. pp. 267–271. IEEE (1997)
13. Díaz, S., Esteban, J.L., Ogihara, M.: A DNA-based random walk method for solving k-SAT. In: International Workshop on DNA-Based Computers. pp. 209–220. Springer (2000)
14. Drakopoulos, G.: Knowledge mining with tensor algebra. Tech. rep., Ionian University (October 2017), doi: 10.13140/RG.2.2.25548.92803
15. Drakopoulos, G., Kanavos, A., Karydis, I., Sioutas, S., Vrahatis, A.G.: Tensor-based semantically-aware topic clustering of biomedical documents. Computation 5(3) (May 2017)
16. Drakopoulos, G., Kanavos, A., Tsakalidis, A.: A Neo4j implementation of fuzzy random walkers. In: SETN (May 2016)
17. Drakopoulos, G., Kanavos, A., Tsakalidis, K.: Fuzzy random walkers with second order bounds: An asymmetric analysis. Algorithms 10(2) (2017)
18. Dunne, P.E., Amos, M., Gibbons, A.: Boolean transitive closure in DNA (1998)
19. Feynman, R.P.: There is plenty of room at the bottom. Engineering and science 23(5), 22–36 (1960)
20. Fu, B., Beigel, R.: Length bounded molecular computing. BioSystems 52(1), 155–163 (1999)
21. Gehani, A., Reif, J.: Micro flow bio-molecular computation. Biosystems 52(1), 197–216 (1999)
22. Gorban, A.N., Gorbunova, K.O., Wunsch, D.C.: Liquid Brain: The proof of algorithmic universality of quasichemical model of fine-grained parallelism. Neural Network World 11(4), 391–412 (2001)
23. Gusfield, D.: Algorithms on strings, trees and sequences: Computer science and computational biology. Cambridge University Press (1997)
24. Henaut, A., Contamine, D.: Computation with DNA. Tech. rep., Rapport de recherche - Institut national de recherche en informatique et en automatique (1996)
25. Hinze, T., Sturm, M.: Towards an in-vitro implementation of a universal distributed splicing model for DNA computation. Proc. Theorietag pp. 185–189 (2000)
26. Hoheisel, J.D., Vingron, M.: DNA chip technology. Biospektrum 4, 17–20 (1998)
27. Kanavos, A., Drakopoulos, G., Tsakalidis, A.: Graph community discovery algorithms in neo4j with a regularization-based evaluation metric. In: WEBIST (April 2017)
28. Kao, M.Y., Ramachandran, V.: DNA self-assembly for constructing 3D boxes. In: ISAAC. vol. 2223, pp. 429–440. Springer (2001)
29. Karp, R.M., Kenyon, C., Waarts, O.: Error resilient DNA computation. In: SODA. pp. 458–467 (1996)
30. Kontopoulos, S., Drakopoulos, G.: A space efficient scheme for graph representation. In: ICTAI. IEEE (November 2014)
31. LaBean, T.H., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J.H., Seeman, N.C.: Construction, analysis, ligation, and self-assembly of DNA triple crossover complexes. Journal of the American Chemical Society 122(9), 1848–1860 (2000)
32. Lipton, R.J.: Speeding up computations via molecular biology. DNA Based Computers 27, 67–74 (1995)
33. Narayanan, A., Zorbalas, S.: DNA algorithms for computing shortest paths. Proceedings of genetic programming 718, 723 (1998)

34. Ogihara, M.: Relating the minimum model for DNA computation and Boolean circuits. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2. pp. 1817–1821. Morgan-Kaufmann Publishers Inc. (1999)
35. Ogihara, M., Ray, A.: Circuit evaluation: Thoughts on a killer application in DNA computing. Computing with Bio-Molecules. Theory and Experiments pp. 111–126 (1998)
36. Ogihara, M., Ray, A.: DNA-based self-propagating algorithm for solving bounded-fan-in Boolean circuits. Genetic Programming 98, 725–730 (1998)
37. Ogihara, M., Ray, A.: The minimum DNA computation model and its computational power. Unconventional Models of Computation pp. 309–322 (1998)
38. Ogihara, M., Ray, A.: Executing parallel logical operations with DNA. In: Proceedings of the 1999 Congress on Evolutionary Computation. vol. 2, pp. 972–979. IEEE (1999)
39. Ogihara, M., Ray, A.: Simulating Boolean circuits on a DNA computer. Algorithmica 25(2-3), 239–250 (1999)
40. Pisanti, N.: DNA computing: A survey. Bulletin of the EATCS 64, 188–216 (1998)
41. Qiu, Z.F., Lu, M.: Take advantage of the computing power of DNA computers. In: International Parallel and Distributed Processing Symposium. pp. 570–577. Springer (2000)
42. Reif, J.H.: Paradigms for biomolecular computation. In: First International Conference on Unconventional Models of Computation. pp. 72–93 (1998)
43. Reif, J.H.: Parallel biomolecular computation: Models and simulations. Algorithmica 25(2), 142–175 (1999)
44. Reif, J.H.: Successes and failures. Science 296, 478–479 (2002)
45. Reif, J.H.: The emergence of the discipline of biomolecular computation in the US. "http://citeseer.nj.nec.com/reif02emergence.html" (2002)
46. Reif, J.H.: The design of autonomous DNA nanomechanical devices: Walking and rolling DNA. Lecture notes in computer science pp. 22–37 (2003)
47. Rooß, D.: Recent developments in DNA computing. In: Proceedings of the 27th International Symposium on Multiple Valued Logic. pp. 3–9 (1997)
48. Rothemund, P.W., Winfree, E.: The program-size complexity of self-assembled squares. In: STOC. pp. 459–468. ACM (2000)
49. Wang, B.: Implementation of a dynamic programming algorithm for DNA Sequence alignment on the Cell Matrix architecture. Master's thesis, Utah State University, Department of Computer Science (2002)
50. Wasiewicz, P., Borsuk, P., Mulawka, J.J., Węgleński, P.: Implementation of data flow logical operations via self-assembly of DNA. In: International Parallel Processing Symposium. pp. 174–182. Springer (1999)
51. Yan, H., LaBean, T.H., Feng, L., Reif, J.H.: Directed nucleation assembly of DNA tile complexes for barcode-patterned lattices. Proceedings of the National Academy of Sciences 100(14), 8103–8108 (2003)