Iranian J Biotech. 2020 January;18(1): e2547 DOI: 10.30498/IJB.2020.195413.2547

Research Article

# Parallelizing Assignment Problem with DNA Strands

**Babak Khorsand , Abdorreza Savadi, Mahmoud Naghibzadeh**

Computer Engineering Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

*Corresponding author: Babak Khorsand, Computer Engineering Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran. Tel: 05138805071, Fax: 05138805000, E-mail: khorsand@mail.um.ac.ir

**Background:** Many problems of combinatorial optimization, which are solvable only in exponential time, are known to be Non-Deterministic Polynomial hard (NP-hard). With the advent of parallel machines, new opportunities have been emerged to develop the effective solutions for NP-hard problems. However, solving these problems in polynomial time needs massive parallel machines and is not applicable up to now.
**Objectives:** DNA (Deoxyribonucleic acid) computing provides a fantastic method to solve NP-hard problems in polynomial time. Accordingly, one of the famous NP-hard problems is assignment problem, which is designed to find the best assignment of n jobs to n persons in a way that it could maximize the profit or minimize the cost.
**Material and Methods:** Applying bio molecular operations of Adelman Lipton model, a novel parallel DNA algorithm have been proposed for solving the assignment problem.
**Results:** The proposed algorithm can solve the problem in time complexity, and just $O(n^2)$ initial DNA strand in comparison with initial sequence, which is used by the other methods.
**Conclusions:** In this article, using DNA computing, we proposed a parallel DNA algorithm to solve the assignment problem in linear time.
*Keywords:* Adelman Lipton model, Assignment, DNA computing, DNA algorithm, Molecular computation.

## 1. Background

Deoxyribonucleic acid (DNA) is a molecule that carries most of the genetic instructions used in the development, functioning, and reproduction of all known living organisms and many viruses. Just one gram of DNA can hold about 680 peta bytes ($6.8 * 10^{14}$) equal to one billion (1) Compact Disc (CD), which take 163,000 centuries to be listened.

Considering parallelism aspect, it also is capable of parallelizing molecules at a time(2), and this enormous parallelism persuades the researchers to build a model by the use of DNA strands to solve NP-hard problems (3). Considering the similarities in biological and mathematical operations, and the facts that DNA is stable and also predictable in its reactions, this macromolecule can be used to encode information for mathematical systems.

In 1994, *Adelman* (4) designed the first model of molecular computation by solving a seven-node instances of Hamiltonian path problem, and then showed the enormous parallel power of DNA computation. In 1995, *Lipton* (5) solved satisfiability problem by Adelman's method and designed a new method called Adelman-Lipton model, which was consisted of three levels as follows:

a) Producing specific sequences and putting them in a test tube.

b) Performing a number of operations on the DNA sequences inside the tube and removing unwanted sequences, which certainly cannot be considered as the final answer of the problem.

c) Reading and reporting the DNA sequences, if any exists, in the final tube as final answer.

Ouyang (6) solved Maximal clique problem in 1997 by designing *Restriction enzyme model.* Roweis designed *sticker model* (7) in 1998. The *Self-assembly model* was designed by Winfree in 1998 (8). The *Surface based model* (9) was designed by Smith in 1998. And finally, Sakomoto proposed the *Hairpin model* in 2000 (10).

As the other applications of DNA computing we can point to Zhang's image encryption algorithm (11) or Patel multiple image encryption (12) which is based on DNA computing or novel data encryption scheme which is proposed by Patro (13) by using DNA computing.

Many efforts have been done by researchers to solve NP-hard problems via these models such as knapsack (14), n-queen (15), binpacking (16), maximal clique problem (17), graph vertex coloring problem (18), maximal matching problem (19), maximal connected subgraph problem (20), maximum complete bipartite

subgraph (21), maximum independent set problem (22), maximum matching problem (23), maximum k-colourable subgraph (24), hamiltonian path (25, 26), minimum k-suppliers (27).

## 2. Objectives

*Assignment* is another famous NP-hard problem (28), which has been solved in different versions by researchers as *Wang* (29) who solved the unbalanced version of assignment, or *Yang* (30) who solved the quadratic version of assignment.

Based on Adleman-Lipton model, we proposed a parallel DNA algorithm with time complexity for assignment problem, and initial DNA strand in comparison with initial DNA strands of similar solutions such as that was proposed by *Wang et al* (31).

## 3. Material and Methods

In Assignment problem, we have n jobs, which can be performed by n people. Each of the people will do each one of the jobs with a specific wage. The objective of this study was to assign each job to one person in such a way that the total wages become minimized.

As an example, consider four teachers ($T_1$-$T_4$) from four different cities ($C_{T1}$-$C_{T4}$) and also four universities ($U_1$-$U_4$) in four different other cities ($C_{U1}$-$C_{U4}$).

**Table 1** shows the ticket price for each flight from cities $C_{T1}$-$C_{T4}$ to cities $C_{U1}$-$C_{U4}$. The goal is to send each teacher to one of the universities in a way that the total price of purchased ticket becomes minimized.

**Table 1.** Ticket prices for the example described in the text.

| City | $C_{T1}$ | $C_{T1}$ | $C_{T1}$ | $C_{T1}$ |
|---|---|---|---|---|
| $C_{U1}$ | 40 | 140 | 90 | 180 |
| $C_{U2}$ | 120 | 20 | 70 | 110 |
| $C_{U3}$ | 80 | 25 | 25 | 115 |
| $C_{U4}$ | 80 | 180 | 130 | 140 |

We can represent **Table 1** as a cost matrix:

$$\begin{bmatrix} 40 & 140 & 90 & 180 \\ 120 & 20 & 70 & 110 \\ 80 & 25 & 25 & 115 \\ 80 & 180 & 130 & 140 \end{bmatrix}$$

One possible assignment is as follow:

$$\begin{bmatrix} 40 & 140 & \mathbf{90} & 180 \\ 120 & 20 & 70 & \mathbf{110} \\ 80 & \mathbf{25} & 25 & 115 \\ \mathbf{80} & 180 & 130 & 140 \end{bmatrix}$$

Where selected flights highlighted as yellow. The total cost of this assignment is: 90$ + 110$ + 25$ + 80$= 305$. But the best possible assignment is:

$$\begin{bmatrix} \mathbf{40} & 140 & 90 & 180 \\ 120 & \mathbf{20} & 70 & 110 \\ 80 & 25 & \mathbf{25} & 115 \\ 80 & 180 & 130 & \mathbf{140} \end{bmatrix}$$

Which will cost 40$ + 20$ + 25$ + 140$= 225$ and means to send $T_1$ to $U_1$, $T_2$ to $U_2$, $T_3$ to $U_3$, and $T_4$ to $U_4$.

### *3.1. Adelman-Lipton Model:*

*Different operations of Adelman-Lipton model:*

- Merge ($T_1$, $T_2$): The content of tube $T_1$ will be merged with $T_2$ and will also be moved into $T_1$.
- Copy ($T_1$, $T_2$): The content of $T_1$ will be copied into the empty tube $T_2$.
- Detect (T): Determining whether any sequence exists in tube T or not.
- Extract ($T_1$, X, $T_2$): The subset of $T_1$, which contains the X sequence will be transferred into $T_2$.
- Select ($T_1$, X, $T_2$): The subset of $T_1$ containing sequences longer than X will be transferred into $T_2$.
- Selecting ($T_1$, L, $T_2$): It will select sequences of size L from tube $T_1$ and put them in tube $T_2$.
- SelectMax ($T_1$, $T_2$): The longest sequence in tube $T_1$ will be chosen and moved into tube $T_2$.
- SelectMin ($T_1$, $T_2$): The shortest sequence in tube $T_1$ will be chosen and moved into tube $T_2$.
- Annealing (T): ssDNA will be converted to dsDNA by freezing the tube.
- Denaturing (T): dsDNA will be converted to ssDNA by freezing the tube.
- Discard (T): The content of tube T will be eliminated.
- PCR (T, $T_1$): A lot of copies of DNA strands will be made and placed in tube $T_1$.
- Append Tail (T, Z): Sequence Z will be appended to the end of all sequences in tube T.
- Read (T): A sequence of tube T will be read.

### *3.2. DNA Algorithm for Assignment Problem*

First of all, all $n^n$ combinations of people and jobs will be generated as different DNA strands. Then, the strands having all people will be separated and the others will be discarded. After that, among remained stands, those that have all jobs will be separated and the others will be discarded. The shortest sequences will be regarded as the final solutions.

*The proposed algorithm consists of six steps:*

- Building *2n* unique10-mer sequences (sequence with length 10) for each of n people and n jobs.
- Building all $n^2$ possible combinations of jobs with people

and then adding $C_i$-mer sequences of G nucleotides ($C_i$ is the cost or profit of performing $J_i$ by $P_i$).

- Construct all $n^n$ possible combinations of choosing n jobs by n people.
- Eliminate all sequences which do not consist of all people.
- Among remaining sequences of previous step, keep only that were consisted of all jobs.
- Consider the shortest sequence as the optimal solution.

*3.3. Strand Design:*

First of all, for each person ($P_i$) and each job ($J_i$) of assignment problem produce a distinct 10mer sequence. We can produce 59049 different 10mer sequences using 3 nucleotides a, t, c. Among them, we should select the sequences, which have the most difference with each other and then put these *2n* sequences in *2n* tube and perform the Polymerase Chain Reaction (PCR) on them. PCR, developed in 1983 by Kary Mullis (32), is a molecular biology technology, which generates a lot of copies from a single piece of DNA sequence. We have to generate sequences by dot product of *n* jobs to *n* person's sequences. Finally, equal to the cost of assigning a person to a job, we should add $C_i$-mer sequences of G nucleotides to the related sequences. For making the length of sequences shorter according to the costs, it is easily possible to make a rule. Such as subtracting the minimum cost can form the cost belonged to all people.

As an example, in our problem, the minimum ticket price was 20$ and the costs were differed at least 5$ with each other. So, we subtracted 20 from all ticket prices and then added a G for each 5$. As you can see in **Table 2** for the four teachers, we used four different 10mer, and for the four universities, we used four others different 10mer.

**Table 2.** Sequences chosen to represent the elements of the assignment problem, which has been described in the **Table 1**.

| | DNA Sequence | | DNA Sequence |
|---|---|---|---|
| $T_1$ | 5' TCTATAACTA 3' | $U_1$ | 5' AACTACATTT 3' |
| $T_2$ | 5' TAACACTATT 3' | $U_2$ | 5' CATCATTTAC 3' |
| $T_3$ | 5' ACTAATCTCT 3' | $U_3$ | 5' ATTACTCCTA 3' |
| $T_4$ | 5' CACACATCTA 3' | $U_4$ | 5' TACCTCAACT 3' |

Then, as it is shown in **Table 3**, we should combine all possible ways and then add one G for each *5$* to all of the sequences and also remove 4G from all of them because of subtracting minimum.

**Table 3.** Sequences chosen to represent the elements of assignment problem.

| | **Length** | **DNA Sequence** |
|---|---|---|
| $T_1 U_1$ | 40 | 5' TCTATAACTAAACTACATTTGGGGGGGGGGGGGGGGGGGG 3' |
| $T_1 U_2$ | 32 | 5' TCTATAACTACATCATTTACGGGGGGGGGGGG 3' |
| $T_1 U_3$ | 24 | 5' TCTATAACTAATTACTCCTAGGGG 3' |
| $T_1 U_4$ | 32 | 5' TCTATAACTATACCTCAACTGGGGGGGGGGGG 3' |
| $T_2 U_1$ | 20 | 5' TAACACTATTAACTACATTT 3' |
| $T_2 U_2$ | 52 | 5' TAACACTATTCATCATTTACGGGGGGGGGGGGGGGGGGGGGGGGGGGG 3' |
| $T_2 U_3$ | 44 | 5' TAACACTATTATTACTCCTAGGGGGGGGGGGGGGGGGGGGGGGG 3' |
| $T_2 U_4$ | 21 | 5' TAACACTATTTACCTCAACTG 3' |
| $T_1 U_3$ | 30 | 5' ACTAATCTCTAACTACATTTGGGGGGGGGG 3' |
| $T_2 U_3$ | 42 | 5' ACTAATCTCTCATCATTTACGGGGGGGGGGGGGGGGGGGGGG 3' |
| $T_3 U_3$ | 34 | 5' ACTAATCTCTATTACTCCTAGGGGGGGGGGGGGG 3' |
| $T_4 U_3$ | 21 | 5' ACTAATCTCTTACCTCAACTG 3' |
| $T_1 U_4$ | 38 | 5' CACACATCTAAACTACATTTGGGGGGGGGGGGGGGGGG 3' |
| $T_2 U_4$ | 44 | 5' CACACATCTACATCATTTACGGGGGGGGGGGGGGGGGGGGGGGG 3' |
| $T_3 U_4$ | 52 | 5' CACACATCTAATTACTCCTAGGGGGGGGGGGGGGGGGGGGGGGGGGGG 3' |
| $T_4 U_4$ | 39 | 5' CACACATCTATACCTCAACTGGGGGGGGGGGGGGGGGG 3' |

*3.4. Sample Space*

Put each of the related strands to the first person in a separated tube.

- For (i in 1 to n)
  - Merge (T, $T_i$)
  - Discard ($T_i$)

- For (i in 2 to n)
  - For (j in 1 to n)
    - PCR (T, $T_j$)
    - AppendTail ($T_j$, $P_i$ $J_j$)
    - Merge (Temp, $T_j$)
    - Discard ($T_j$)
  - Discard (T)
  - Copy (Temp, T)
  - Discard (Temp)

The trace of the top algorithm for three people and three job is shown as an example in **Table 4**.

**Table 4**. Trace for three people and three jobs

| Round | Tube $T_1$ | Tube $T_2$ | Tube $T_3$ | Tube T |
|---|---|---|---|---|
| i=1 | $P_1J_1$ | $P_1J_2$ | $P_1J_3$ | $P_1J_1$,$P_1J_2$,$P_1J_3$ |
| i=2, *j*=1 | $P_1J_1P_2J_1$,$P_1J_2P_2J_1$,$P_1J_3P_2J_1$ | | | $P_1J_1$,$P_1J_2$,$P_1J_3$ |
| i=2, *j*=2 | | $P_1J_1P_2J_2$,$P_1J_2P_2J_2$, $P_1J_3P_2J_2$ | | $P_1J_1$,$P_1J_2$,$P_1J_3$ |
| i=2, *j*=3 | | | $P_1J_1P_2J_3$,$P_1J_2P_2J_3$,$P_1J_3P_2J_3$ | $P_1J_1$,$P_1J_2$,$P_1J_3$ |
| i=3, *j*=1 | $P_1J_1P_2J_1P_3J_1$,$P_1J_2P_2J_1P_3J_1$, $P_1J_3P_2J_1P_3J_1$,$P_1J_1P_2J_2P_3J_1$, $P_1J_2P_2J_2P_3J_1$,$P_1J_3P_2J_2P_3J_1$, $P_1J_1P_2J_3P_3J_1$,$P_1J_2P_2J_3P_3J_1$, $P_1J_3P_2J_3P_3J_1$ | | | $P_1J_1P_2J_1$,$P_1J_2P_2J_1$, $P_1J_3P_2J_1$,$P_1J_1P_2J_2$, $P_1J_2P_2J_2$,$P_1J_3P_2J_2$, $P_1J_1P_2J_3$,$P_1J_2P_2J_3$, $P_1J_3P_2J_3$ |
| i=3, *j*=3 | | | $P_1J_1P_2J_1P_3J_3$,$P_1J_2P_2J_1P_3J_3$, $P_1J_3P_2J_1P_3J_3$,$P_1J_1P_2J_2P_3J_3$, $P_1J_2P_2J_2P_3J_3$,$P_1J_3P_2J_2P_3J_3$, $P_1J_1P_2J_3P_3J_3$,$P_1J_2P_2J_3P_3J_3$, $P_1J_3P_2J_3P_3J_3$ | $P_1J_1P_2J_1$,$P_1J_2P_2J_1$, $P_1J_3P_2J_1$,$P_1J_1P_2J_2$, $P_1J_2P_2J_2$,$P_1J_3P_2J_2$, $P_1J_1P_2J_3$,$P_1J_2P_2J_3$, $P_1J_3P_2J_3$ |
| Final | $P_1J_1P_2J_1P_3J_1$,$P_1J_2P_2J_1P_3J_1$,$P_1J_3P_2J_1P_3J_1$,$P_1J_1P_2J_2P_3J_1$,$P_1J_2P_2J_2P_3J_1$,$P_1J_3P_2J_2P_3J_1$,$P_1J_1P_2J_3P_3J_1$, $P_1J_2P_2J_3P_3J_1$,$P_1J_3P_2J_3P_3J_1$,$P_1J_1P_2J_1P_3J_2$,$P_1J_2P_2J_1P_3J_2$,$P_1J_3P_2J_1P_3J_2$,$P_1J_1P_2J_2P_3J_2$,$P_1J_2P_2J_2P_3J_2$, $P_1J_3P_2J_2P_3J_2$,$P_1J_1P_2J_3P_3J_2$,$P_1J_2P_2J_3P_3J_2$,$P_1J_3P_2J_3P_3J_2$,$P_1J_1P_2J_1P_3J_3$,$P_1J_2P_2J_1P_3J_3$,$P_1J_3P_2J_1P_3J_3$, $P_1J_1P_2J_2P_3J_3$,$P_1J_2P_2J_2P_3J_3$,$P_1J_3P_2J_2P_3J_3$,$P_1J_1P_2J_3P_3J_3$,$P_1J_2P_2J_3P_3J_3$,$P_1J_3P_2J_3P_3J_3$ | | | |

Eliminate all sequences, which do not consist of all people.

- For (i in 1 to n)
  - Extract (T, $P_i$, Temp)
  - Discard(T)
  - Copy (Temp, T)
  - Discard (Temp)

Eliminate all sequences, which do not consist of all jobs.

- For (i in 1 to n)
  - Extract (T, $Job_i$, Temp)
  - Discard(T)
  - Copy (Temp, T)
  - Discard (Temp)

Final Answer

- SelectMin (T, Temp)
- Read (Temp)

## 4. Results

*4.1. The solutions of assignment problem with n jobs and individuals can be obtained by above mentioned DNA molecules operations.*

**Proof.** At first, all combinations of the n job assignments were generated in the sample space. Then, the sequences containing all people were kept and this guaranteed that none of the people are jobless. And finally, among the remaining sequences, the sequences containing all the jobs were kept and this guaranteed that all of the jobs will be done at the end. Therefore, for sure, all possible assignments are in final tube, and as the cost of each assignment was attached to each of them in the strain design phase, so the strand with minimum length

among the strands of final tube would be the minimum cost assigning solution.

*4.2. The time complexity of the proposed algorithm for solving assignment problem with n jobs and individuals is $n^{2.}$*

**Proof.** As the complexity of every biological operation is *O(1)* (33), the time complexity of algorithm can be calculated by adding the time complexity of all steps as follows:

T (Sample space): *O(n)* for the first loop and $O(n^2)$ for the nested loop.

T (Persons): Four biological operations in each step, which become *O(1)*, and in n step it would be *O(n)*.

T (Jobs): Four biological operations in each step, which become *O(1)*, and in n step it would be *O(n)*.

T (Last step): Two biological operations, which become *O(1)*.

So, the time complexity of the algorithm will be arrived by equation 1 as follows:

*Equation 1 : Time Complexity*

$$T(n)= O(n) + O(n^2) + O(n) + O(n) + O(1) = O(n^2)$$

## 5. Discussion

In the proposed method, by using adenine, thymine, and cytosine; we produced distinct 10mer sequences equal to the number of people and jobs of assignment problem. Then, by using polymerase chain reaction on them, we produced $n^2$ sequences equal to the dot product of n persons to n jobs. Finally, by adding guanine equal to the cost of assigning a job to a person to the related sequence, we got the desired sequence.

*Wang et al.*(31), *Kang et al.*(34), *Shu et al.*(35), and *Ebrahim et al.*(36), *Tsaftaris et al.*(37), *Rashid et al.* (38) all used the same model to solve the assignment problem using $n^n$ distinct sequence with the time complexity of $O(n^2)$. The proposed algorithm reduced the initial cost to a great extent by using just 2n initial sequences in comparison with $n^n$ initial sequences, which is used by the other methods and solve the problem with the same time complexity.

## 6. Conclusion

Ultra-efficient parallelism of the methods of DNA computing in contrast with the obvious limitations in storage, speed, intelligence, and miniaturization of electronic computers is considerable. Although there are still some problems that need a further study in biologic technology, it is still possible to solve a lot of NP-hard problems in linear time via DNA computing. In this article, we highlighted a DNA computing model with biological operations based on Adelman-Lipton model to solve Assignment problem with the time complexity of $O(n^2)$ with just buying *2n* 10mer sequences and making the other needed sequences in our lab in comparison with the previous methods which need a lot of different sequences.

We hope that, in near future, molecular computer become usable instead of electronic computers, which can cause DNA computing solutions become applicable.

## References


1. Erlich Y, Zielinski D. DNA Fountain enables a robust and efficient storage architecture. *Science*. 2017;355(6328):950-954. https://doi.org/10.1126/science.aaj2038
2. Amos M, Păun G, Rozenberg G, Salomaa A. Topics in the theory of DNA computing. *TCS*. 2002;287(1):3-38. https://doi.org/10.1016/s0304-3975(02)00134-2
3. Knuth DE. Postscript about NP-hard problems. ACM SIGACT News. 1974;6(2):15-16. https://doi.org/10.1145/1008304.1008305
4. Adleman LM. Molecular computation of solutions to combinatorial problems. *Science*. 1994:1021-1024.
5. Lipton RJ. DNA solution of hard computational problems. *Science*. 1995;**268**(5210):542-545. https://doi.org/10.1126/science.7725098
6. Ouyang Q, Kaplan PD, Liu S, Libchaber A. DNA solution of the maximal clique problem. *Science*. 1997;**278**(5337):446-449. https://doi.org/10.1126/science.278.5337.446
7. Roweis S, Winfree E, Burgoyne R, Chelyapov NV, Goodman MF, Rothemund PW, *et al.* A sticker-based model for DNA computation. *JCB*. 1998;**5**(4):615-629. https://doi.org/10.1089/cmb.1998.5.615
8. Winfree E, Liu F, Wenzler LA, Seeman NC. Design and self-assembly of two-dimensional DNA crystals. *Nature*. 1998;**394**(6693):539. https://doi.org/10.1038/28998
9. Smith LM, Corn RM, Condon AE, Lagally MG, Frutos AG, Liu Q, *et al.* A surface-based approach to DNA computation. *JCB*. 1998;**5**(2):255-267.
10. Sakamoto K, Gouzu H, Komiya K, Kiga D, Yokoyama S, Yokomori T, *et al*. Molecular computation by DNA hairpin formation. *Science*. 2000;**288**(5469):1223-1226. https://doi.org/10.1126/science.288.5469.1223
11. Zhang Y. The image encryption algorithm based on chaos and DNA computing. *MTA*. 2018;**77**(16):21589-21615. https://doi.org/10.1007/s11042-017-5585-x
12. Patel A, Parikh M. Multiple Image Encryption Using Chaotic Map And DNA Computing. 2018.
13. Patro KAK, Acharya B. Novel data encryption scheme using DNA computing. Advances of DNA Computing in Cryptography. *CRC*. p.69-110. https://doi.org/10.1201/9781351011419-4
14. Darehmiraki M, Nehi HM. Molecular solution to the 0–1 knapsack problem based on DNA computing. *AMC*. 2007;**187**(2):1033-1037. https://doi.org/10.1016/j.amc.2006.09.020
15. Wang Z, Huang D, Tan J, Liu T, Zhao K, Li L. A parallel algorithm for solving the n-queens problem based on inspired computational model. *BS*. 2015;**131**:22-29. https://doi.org/10.1016/j.biosystems.2015.03.004
16. Sanches CAA, Soma NY. A polynomial-time DNA computing solution for the Bin-Packing Problem. *AMC*. 2009;**215**(6):2055-

2062. https://doi.org/10.1016/j.amc.2009.07.051
17. Lingjing S, Zhichao S, Liuqing W, Yafei D. A Molecular Computing Model for Maximum Clique Problem Based on DNA Nanoparticles. *JCTN*. 2014;**11**(10):2120-2124. https://doi.org/10.1166/jctn.2014.3615
18. Xu J, Qiang X, Zhang K, Zhang C, Yang J. A DNA computing model for the graph vertex coloring problem based on a probe graph. *Engineering*. 2018;**4**(1):61-77. https://doi.org/10.1016/j.eng.2018.02.011
19. Wang Z, Ji Z, Su Z, Wang X, Zhao K. Solving the maximal matching problem with DNA molecules in Adleman–Lipton model. *IJB*. 2016;**9**(02):1650019. https://doi.org/10.1142/s1793524516500194
20. Guo N, Pu J, Wang Z, Huang D, Li L, Zhao K. A New Algorithm to Solve the Maximal Connected Subgraph Problem Based on Parallel Molecular Computing. *JCTN*. 2016;**13**(10):7692-7695. https://doi.org/10.1166/jctn.2016.4276
21. Yin C, Pu J, Wang Z, Wang X. Solving the Maximum Complete Bipartite Subgraph Problem Based on Molecule Parallel Supercomputing. *JCTN*. 2016;**13**(1):314-318.
22. CHEN J, NING A, ZHI Z, HU L, ZHANG H. Exact algorithm for maximum independent set problem in graph theory. *CEA*. 2016;(1):5.
23. Yang J, Yin Z, Huang K, Cui J. The Maximum Matching Problem Based on Self-Assembly Model of Molecular Beacon. *NNL*. 2018;**10**(2):213-218. https://doi.org/10.1166/nnl.2018.2616
24. Moazzam A, Dalvand B. Molecular solutions for the Maximum K-colourable Sub graph Problem in Adleman-Lipton model. arXiv preprint arXiv:161007294. 2016.
25. Yang R, Zhang C, Gao R, editors. A new bionic method inspired by DNA computation to solve the hamiltonian path problem. *ICIA*. 2017: IEEE. https://doi.org/10.1109/icinfa.2017.8078909
26. Mahasinghe A, Hua R, Dinneen MJ, Goyal R. Solving the Hamiltonian cycle problem using a quantum computer. *Computing*. 2019;**29**:40. https://doi.org/10.1145/3290688.3290703
27. Safaei S, Safaei V, Trippe ED, Aguar K, Arabnia HR, editors. Solving Minimum k-supplier in Adleman-Lipton model. *CSCI*. 2017: IEEE. https://doi.org/10.1109/csci.2017.33
28. König D. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *MA*. 1916;**77**(4):453-465. https://doi.org/10.1007/bf01456961
29. Wang Z, Pu J, Cao L, Tan J. A parallel biological optimization algorithm to solve the unbalanced assignment problem based on DNA molecular computing. *IJMS*. 2015;**16**(10):25338-25352.
30. Yang X, Lu Q, Li C, Liao X. Biological computation of the solution to the quadratic assignment problem. *AMC*. 2008;200(1):369-377. https://doi.org/10.1016/j.amc.2007.11.016
31. Wang Z, Tan J, Huang D, Ren Y, Ji Z. A biological algorithm to solve the assignment problem based on DNA molecules computation. *AMC*. 2014;**244**:183-190. https://doi.org/10.1016/j.amc.2014.06.098
32. Mullis K, Erlich H, Arnheim N, Horn G, Saiki R, Scharf S. One of the first Polymerase Chain Reaction (PCR) patents. Google Patents. 1987.
33. Chang W-L, Ren T-T, Luo J, Feng M, Guo M, Lin KW. Quantum algorithms for biomolecular solutions of the satisfiability problem on a quantum machine. *TNB*. 2008;**7**(3):215-222. https://doi.org/10.1109/tnb.2008.2002286
34. Kang Z, Tong X, Jin X. Algorithm of DNA computing on optimal assignment problems. *JSEE*. 2007;**29**:1183-1187.
35. Shu J-J, Wang Q-W, Yong K-Y. DNA-based computing of strategic assignment problems. *PRL*. 2011;**106**(18):188702.
36. Ibrahim Z, Tsuboi Y, Ono O, editors. Solving unconstraint assignment problem by a molecular-based computing algorithm. *ISIE*. 2004: IEEE. https://doi.org/10.1109/isie.2004.1572031
37. Ekka AA, Sahoo B, editors. A DNA computing approach to solve Task Assignment problem in Real Time Distributed computing System. *MMC*. 2007: Citeseer.
38. Ibrahim GJ, Rashid TA, Sadiq AT. Evolutionary DNA Computing Algorithm for Job Scheduling Problem. *IETEJS*. 2018;**64**(4):514-527. https://doi.org/10.1080/03772063.2017.1362964