Desalination and Water Treatment
www.deswater.com
doi:10.5004/dwt.2018.22175

111 (2018) 88–93
April

Research on water resources optimal scheduling problem based on parallel biological computing

Zuwen Ji[a], Zhaocai Wang[b,a,*], Xiaoguang Bao[b], Xiaoming Wang[b], Tunhua Wu[c,*]

[a]*State Key Laboratory of Simulation and Regulation of River Basin Water Cycle, China, Institute of Water Resources and Hydro power Research, Beijing 100048, China, Tel. +86 10 68786631, Fax +86 10 68786631, email: jzw@iwhr.com, zwji66@163.com (Z. Ji)*
[b]*College of Information, Shanghai Ocean University, Shanghai 201306, China, Tel. +86 2161900624, Fax +86 2161900624, email: zcwang1028@163.com (Z. Wang), xgbao@shou.edu.cn (X. Bao), xmwang@shou.edu.cn (X. Wang)*
[c]*School of Information & Engineering, Wenzhou Medical University, Wenzhou 325035, China, Tel. +86 57786689913, Fax +86 57786689913, email: appll188@163.com (T. Wu)*

Received 17 November 2017; Accepted 7 March 2018

ABSTRACT

Water resource shortage has become one of the main factors restricting the rapid economic and social development of mankind, how limited water resources reasonable allocation to various users, protection of life and production, ecological water safety is facing an important problem in various countries. Water resources optimal scheduling problem also has a wide range of application space, which involves cost control and optimization of design expertise. The shortest flow path problem, as a well-known NP-complete problem and one of famous water resources optimal scheduling problem, has attracted the attention of many scholars. In this paper, we use a new parallel algorithm to solve the problem by basic DNA molecular operations. We reasonably design DNA chains that characterize cities and paths, take appropriate biological operations and get solutions of the task scheduling problem in proper length range with $O(n^2)$ time complexity. The ability of biological manipulation in the algorithm helps us better understand DNA computing, and can be widely used to solve more complex problems.

Keywords: DNA computing; Water resources optimal scheduling problem; The shortest flow path problem; NP-complete problem

1. Introduction

The optimal scheduling of water resources is a multi-objective stochastic sequential decision making problem in theory. The scheduling objectives are usually related to flood control, power generation, urban and industrial water supply, irrigation and reservoir deposition prevention and ecological environment protection. In order to reduce the complexity of the problem, the weight of each objective can be assigned according to the actual situation, or the secondary objective is turned into the constraint condition, so that the problem becomes a single objective stochastic sequential decision-making problem.

The main steps of water resources optimal scheduling are as follows: (1) clear scheduling objectives and various constraints; (2) build appropriate models and select optimization methods; (3) analyze the results and form the scheduling scheme; (4) use administrative and economic measures to promote the implementation of scheduling scheme; (5) by using the simulation of the actual investigation or other scheduling schemes, it is necessary to determine whether it is necessary to improve the target, model, solution method, scheduling rules and water rate system, etc.

There are mathematical programming method, network flow method, large scale system decomposition coordination method and simulation technology for water

*Corresponding author.

resources optimization scheduling. In mathematical programming, linear programming and dynamic programming are used more frequently. The nonlinear model is usually used to avoid the loss of precision caused by linearization when there has a hydraulic power generation target. When the scale of the problem is large, the decomposition coordination technique can be applied to deal with the problem. Simulation technology is a powerful tool to evaluate whether the system operation can produce expected benefits. Because the simulation model can describe the water resources allocation system characteristics and the expected benefits in various water conditions, water requirement and operation mode of operation, and easy to solve. Therefore, it has been widely used at home and abroad in recent years.

In this paper, we use biological parallel computing algorithm to solve the classical problem of water resources scheduling—the shortest flow path problem. The problem can be described as: given *n* cities and *m* flow paths with cost weight between different cities, the solution is to find minimum cost weight flow path which it starts designated city, passes the rest of the cities one time and only once, and finally comes back to the original starting point. The shortest flow path problem is a typical NP-complete problem, and it is also considered as one of the most challenging issues in the field of water resources regulation. For instance, the edge-weighted graph *G* in Fig. 1 defines such a problem. We assume that the starting and ending city vertex is v_1. It is not difficult to find that the path $v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_4 \rightarrow v_3 \rightarrow v_1$ with total weight 10 is a solution to shortest flow path problem for graph *G* in Fig. 1. But with the scale of the problem increasing, the problem is getting more and more difficult to be solved.

In 1961, Feynman [1] gave a visionary talk to describe the construction of computer, the possibility of submicroscopic. Although remarkable progress has made in computer miniaturization, this goal has not been achieved yet. Computer scientists rank computational problems in three categories: easy, hard and incomputable [2]. One of the major opinions of computer science in the understanding is that many important computational problems are NP-complete and there are unlikely to have efficient algorithms to exactly solve the problems. To find more efficient algorithms is a hot spot of research. DNA computing, by virtue of its highly parallel computing ability, large storage space, low loss characteristics, can overcome the shortcomings of traditional electronic computer storage and computing speed. In 1994, Adleman [3] firstly proved that DNA computing can be used to solve computationally difficult problems, such as the Directed Hamiltonian Path Problem, and demonstrated the strong parallel computing power of DNA computing. The parallelism implies DNA computers may solve more complex problems such as NP-complete problems in linear scale growth time, comparing with the exponential growth time required by electronic computers. Lipton [4] confirmed that Adleman's method can be used to solve one kind of NP-complete problem (the satisfiability problem). DNA computing, as an interdisciplinary science using DNA biotechnologies to solve complex problems of computational mathematics and optimization, has a wide application prospect in handling scabrous problems. In recent years, DNA computation has received considerable interest from researchers. Many typical biological computing models, such as Adleman-Lipton model [3,4], the sticker model [5], the restriction enzyme model [2,6], the self-assembly model [7], the hairpin model [8] and the surface-based model [9] have already been proposed and built. Based on previous computing models, lots of DNA algorithms and procedures have been executed to solve intractable NP-complete problems [10–21]. In order to better reflect the advantages of biological computing, it is of great significance that we try to solve more complex problems by using DNA biological computing.

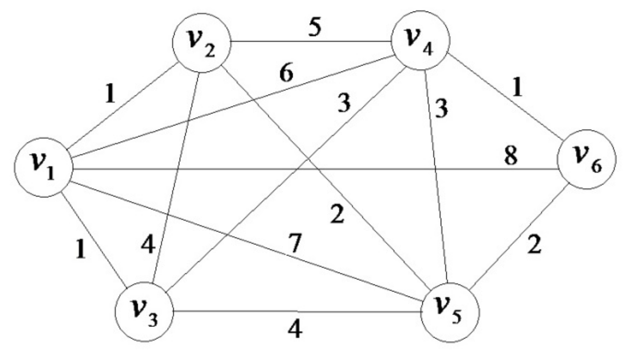

Fig. 1. An edge-weighted graph G with 6 vertices.

The rest of this paper is organized as follows. In Section 2, the preliminary knowledge, including the Adleman-Lipton model and the shortest flow path problem, is introduced. Section 3 proposes a DNA molecular parallel algorithm to solve the shortest flow path problem. The algorithm's theoretical proof and complexity are got in Section 4. We get conclusions in Section 5.

2. Preliminary knowledge

2.1. Adleman-Lipton model

The DNA nucleotides are adenine, guanine, cytosine and thymine, commonly abbreviated as *A*, *G*, *C* and *T*. Every strand, according to chemical structure, has a 5 and a 3 endpoint; hence, any single strand has a natural direction. The pairs (*A*, *T*)and (*G*, *C*) are known as complementary base pairs. For instance, The strand 3′*GACGTCATGTGG*5′ is as the complementary one of 5′*CTGCAGTACACC*3′ and 3′*GACGTCATGTGG*5′ can be denoted by $\overline{5'CTGCAGTACACC3'}$. Thus, the singled strand 5′*CTGCAGTACACC*3′and 3′*GACGTCATGTGG*5′ can form a double strand. The length of a single DNA stranded is determined by the number of nucleotides comprising the single strand. So, if a single stranded DNA having 10 nucleotides, we call it 15 *mer*.

The DNA biological manipulations introduced by Adleman [3] and Lipton [4] are firstly described. These manipulations can be used to solve the shortest flow path problem. Given test tubes, containing finite DNA molecule strings over the alphabet {A,C,G,T}, we can execute the following operations:

(1) *Merge* (T_1, T_2,…,T_n): given tubes T_1, T_2,…, T_n, the operation stores the union $T_1 \cup T_2 \cup \ldots \cup T_n$ in T_1 and leaves other tubes empty.

(2) *Copy* ($T_1, T_2, \ldots, T_n$): given a tube T_1, the operation gets other tubes $T_2, \ldots, T_n$ with same strands as T_1.

(3) *Separation* (T_1, X, T_2): given a tube T_1 and a kind of strings X, the operation moves away all strands with string X from T_1, and leaves surplus strands in tube T_2.

(4) *Selection* (T_1, L, T_2): given a tube T_1 and integer number L, the operation transfers all strands with length L from T_1 to T_2.

(5) *Discard (T)*: given a tube T, the operation clears the strands in T and leaves the T empty.

(6) *Read (T)*: given a tube T, the operation can be used to get the explicit DNA combinations of the strands in tube T.

(7) *Append-tail(T,Z)*: given a tube T and singled strands Z, the operation attaches Z at the end of every strand in tube T.

(8) *Cleavage* (T,): given a tube T and strings with, the operation divides every strand containing $[\gamma_0\gamma_1]$ in T to two kinds different strands:

$[\ldots\alpha\gamma_0\gamma_1\beta\gamma_0\gamma_1\delta\ldots] \Rightarrow [\ldots\alpha\gamma_0], [\gamma_1\,\beta\gamma_0], [\gamma_1\delta\ldots]$.

(9) *Annealing (T)*: given a tube T, the operation generates all feasible double strands and leaves them still in T.

(10) *Denaturation (T)*: given a tube T, the operation segments each double strand in T to two single strands as:

$$\begin{bmatrix} \alpha\beta \\ \overline{\alpha\beta} \end{bmatrix} \Rightarrow [\alpha\beta], [\overline{\alpha\beta}].$$

(11) *Sort* (T_1, T_2, T_3): for a given test tube T_1, it chooses the shortest length strands in the tube T_2, the longest strands in T_3 and the rest strands still in T_1.

Since these operations can be executed in a constant number steps with DNA biological strands [22], we confirm each operation with $O(1)$ time complexity.

2.2. The shortest flow path problem

The shortest flow path problem is a typical NP problem [23]. Given a graph with n cities and m edges, the solution path to the problem needs to be satisfied:

(1) The path must be sequentially joined together by the cities.

(2) The path starts and ends from the same designated city.

(3) The path goes through the rest of the $n - 1$ cities one time and only once.

(4) The goal solution is to find paths satisfying the above conditions and with minimized sum of weight cost.

While, the feasible combination paths is between $n!$ to $m!$ kinds, the work is very difficult to search optimal solution in such large quantities scope, this process needs a long time complexity and huge storage space, even for the fastest silicon based computers.

3. DNA algorithm for the shortest flow path problem

3.1. Initial idea

The initial idea to solve the shortest flow path problem is as followed: generate strands corresponding to all possible edges connected paths in data pool, next, check out the path strands for starting and terminating in a designated city and going through the rest cities one time and only once; then, append the cost weighted length strands in order to identify cost difference; finally, get the solutions of the shortest flow path problem by corresponding DNA operations. Concretely, the proposed algorithm has four steps.

Step 1: Automatic response generate all possible edge connected path strands starting and ending at the specified city vertex for the shortest flow path problem;

Step 2: Append corresponding cost weight strands on the end of all possible combinations for mutual comparison;

Step 3: Check out the eligible path strands containing the other cities one time and only once.

Step 4: Select the optimal strands as the solution to the shortest flow path problem;

3.2. Strand design

In the following, the symbols $A_i, B_i (i \in \{1,2,\ldots, n\})$ indicate different DNA singled strands having the same integer length (supposing t *mer*, certainly, the integer length t is mainly depended on the scale of corresponding problem as to distinguish all symbols [24–41]), and the vertex v_i can be represented by strands A_iB_i. We apply different DNA strands symbols $B_iA_j, B_iA_j (1 \le i < j \le n)$ to denote the edges $e_{i,j}$ in graph, the corresponding cost weight strands are $w_{i,j}$ and $||w_{i,j}|| = c_{i,j}$ *mer* in order to compare the cost sum of different combination paths. Meanwhile, we design singled weighted strands # with t-mer length to show the beginning and ending of the path strands.

3.3. An example of the problem

We firstly use Fig. 1 with 6 vertices and 13 edges as an example to illustrate the algorithm of the shortest flow path problem. We need to search the shortest continuous path which starts and ends vertex v_1, meanwhile, passes the rest 5 vertices one time and only once. Next, we explain the process of biological computing algorithm.

3.3.1. Generate data pool

For the shortest flow path problem, all combination paths are firstly continuous connected, in addition, the paths must start the setting vertex v_1 and stop it. We Let

$P = \{\#A_1B_1, B_1A_1\#, A_iB_k \mid k = 2,3,\ldots, n\}$

$Q = \{\overline{B_iA_j} \mid e_{i,j} E; 1 \le i < j \le n\}$

Algorithm 3.1: Generate all possible path strands in data pool.

```
(1-1) Merge (P,Q);
(1-2) Annealing (P);
(1-3) Denaturation (P);
(1-4) Separation (P, {#$A_1B_1$}, $T_1$);
(1-5) Discard (P);
(1-6) Separation ($T_1$, {$A_1B_1$#}, P);
```

In the above operations, we get the sequentially connected path strands that denote starting and ending specified vertex v_1. For example, for the graph in Fig. 1, we have singled strands:

$$\#A_1B_1A_6B_6A_5B_5A_4B_4A_2B_2A_3B_3A_5B_5A_1B_1\# \in P$$

which denote the path $v_1\rightarrow v_6\rightarrow v_5\rightarrow v_4\rightarrow v_2\rightarrow v_3\rightarrow v_5\rightarrow v_1$. Of course, it cannot be the optimum solution because the path passes the vertex v_5 twice. The operation can be finished in $O(1)$ steps since each single manipulation above works in $O(1)$ steps.

3.3.2. Check out proper paths

The solutions paths to shortest flow path problem need pass the other vertices one time and only once, so we should choose the path strands to meet this limitation. If one path miss the vertex v_i, it should be discarded. For example, for the graph in Fig. 1, we have singled strands:

$$\#A_1B_1A_2B_2A_4B_4A_5B_5A_6B_6A_1B_1\# \in P$$

which denote the path $v_1\rightarrow v_2\rightarrow v_4\rightarrow v_5\rightarrow v_6\rightarrow v_1$. Naturally, it cannot be the optimum solution because the path misses the vertex v_3. The length of strands A_iB_i denoting the vertex v_i is 2*t mer*. The length of the strands # denoting the starting and ending signal is *t mer*. Consequently the length of sequentially connected path strands which denote starting and ending vertex v_1 then passes the other n–1 vertices one time in tube P must be between $(2n + 4)t$ *mer*. We can get the strands by the following manipulations.

Algorithm 3.2: Check out the qualified path strands.

```
For $k = 2$ to $k = n$
(2-1) Separation (P,{$A_kB_k$}, $T_2$);
(2-2) Discard (P);
(2-3) Copy ($T_2$; P);
(2-4) Discard ($T_2$);
End for
(2-5) Selection (P,$2n + 4$, $T_3$).
```

In the above operation we use a "For" clause. Thus this operation can be finished in $O(n)$ steps since each single manipulation above works in $O(1)$ steps.

3.3.3. Append corresponding cost weight strands

In order to compare the sum of weights of different continuous paths, we append the weight chains to the corresponding paths strands.

Algorithm 3.3: Compare time cost of different individuals in same allocations.

```
For $i = 1$ to $i = n$–1
For $j = i + 1$ to $j = n$
(3-1) Separation($T_3$,{$B_iA_j$}, $T_4$);
(3-2) If(Detect($T_4$))
Then execute (3-3) to (3-5)
(3-3) Append -tail($T_4$, $w_{ij}$);
(3-4) Merge($T_3$, $T_4$);
(3-5) Discard($T_4$).
End for
End for
```

For example in Fig. 1, we have singled strands:

$$\#A_1B_1A_3B_3A_2B_2A_4B_4A_5B_5A_6B_6A_1B_1\# \in P$$

which denote the path $v_1\rightarrow v_3\rightarrow v_2\rightarrow v_4\rightarrow v_5\rightarrow v_6\rightarrow v_1$. After the step, the strands can be appended the corresponding weight chains to

$$\#A_1B_1A_3B_3A_2B_2A_4B_4A_5B_5A_6B_6A_1B_1\#w_{13}w_{32}w_{24}w_{45}w_{56}w_{61}.$$

The operation can be finished in $O(n^2)$ time complexity since each single manipulation above works in $O(1)$ steps.

3.3.4. Get the solutions to shortest flow path problem

The shortest length strands mean the solution to the shortest flow path problem. So we can choose it and get the final result. For example in Fig. 1, the singled strands

$$\#A_1B_1A_2B_2A_5B_5A_6B_6A_4B_4A_3B_3A_1B_1\#w_{12}w_{25}w_{56}w_{64}w_{43}w_{31}$$

standing for the path $v_1\rightarrow v_2\rightarrow v_5\rightarrow v_6\rightarrow v_4\rightarrow v_3\rightarrow v_1$ are the optimum solution strands. This is done by the following manipulations:

Algorithm 3.4: Search the best solution strands.

```
(4-1) Sort($T_3$,$T_5$,$T_6$);
(4-2) Read($T_5$);
```

Simply, the operation can be finished in $O(1)$ step.

4. The complexity and feasibility of the proposed DNA algorithm

The following theorems show that, our proposed DNA parallel algorithm can get solutions strands of the shortest flow path problem with proper length range in $O(n^2)$ time complexity.

Theorem 1. *The solutions strands of shortest flow path problem can be got by our designed DNA molecule operations.*

Proof. We firstly generate strands denoting all possible continuous paths which start and end definite vertex in data pool by Algorithm 3.1. Also, for the shortest flow path problem, each path should pass the other vertices one time and only once, we select the paths satisfying the restriction through the Algorithm 3.2. Next, we append edge cost weight strands at the end of previous ones in order to find best result in Algorithm 3.3. Surely, the solution to shortest flow path problem has minimum cost weight sum in all possible allocations. At last, we search and get the best answer to the shortest flow path problem in Algorithm 3.4.

Theorem 2. *The shortest flow path problem can be solved in $O(n^2)$ time complexity using DNA parallel operations.*

Proof. Owing to the time complexity of each operation is in O(1) [16], the operations of previous algorithm can be obviously executed in limited time range. The time complexity T of total algorithm is as follows:

$T(Algorithm\ 3.1) = O(1)$;

$T(Algorithm\ 3.2) = O(n)$;

$T(Algorithm\ 3.3) = O(n^2)$;

$T(Algorithm\ 3.4) = O(1)$;

$T = T(Algorithm\ 3.1) + T(Algorithm\ 3.2) + T(Algorithm\ 3.3) + T(Algorithm\ 3.4)$

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

$= O(n^2)$

In conclusion, we can find the solutions of shortest flow path problem in $O(n^2)$ time complexity.

Theorem 3. *The solution strands of shortest flow path problem can be searched in appropriate length range.*

Proof. After the operations of Algorithm 3.2, we generate strands standing for continuous path which start and end vertex v_1 and pass the other vertices one time and only once. The strands can be described:

$\#A_1B_1\ldots A_iB_i\ldots B_1A_1\#i$ {2,3,…, n}

In the Algorithm 3.3, we append the cost weight strands in order to find the optimal result strands, so the strands after Algorithm 3.3 can be showed:

$\#A_1B_1\ldots A_iB_i\ldots B_1A_1\#w_{1,m}w_{m,n}\ldots w_{s,1}i,\ m,\ n,\ s \in \{2,3,\ldots,n\}$.

Besides, we initially set the chain length as $||A_i|| = ||B_i|| = ||\#|| = t$ *mer*,($1 \le i \le n$) and $||w_{kl}|| = c_{kl}$ *mer*,($1 \le k < l \le n$, c_{kl} *is the edge cost weight*), and we suppose $t = max\{c_{kl}\}$,so the length of DNA strands S in T_4 is

$$||S|| = ||\#|| + ||A_1|| + ||B_1|| + \ldots + ||A_i|| + ||B_i|| + \ldots + ||B_1|| + ||A_1|| + ||\#|| + ||w_{1,m}|| + ||w_{m,n}|| + \ldots + ||w_{s,1}||$$

$$= t + \underbrace{t + t + \cdots + t + t}_{2(n+1)} + t + ||w_{1,m}|| + \left\|w_{m,n}\right\| + \ldots + ||w_{s,1}||$$

$$\therefore t = max\{c_{kl}\}, |E| = m$$

$$\therefore 0 \le ||w_{1,m}|| + ||w_{m,n}|| + \ldots + ||w_{s,1}|| \le mt$$

$$\therefore (2n+4)t£\ ||S||\ £(2n+4)t + mt$$

So the length of solutions strands in Algorithm 3.4 is in appropriate length range. We can be relatively easy to find and read out the strands.

5. Conclusions

In this paper, we utilize DNA computing algorithm to solve the shortest flow path problem using the Adleman-Lipton model based on biological operations. Owing to electronic computers have obvious limitations in speed, storage capacity, mentality and physical space, the methods of DNA computing is getting more and more attention. In particular, the method is highly efficient and parallel. Compared with previous algorithms, our DNA computing algorithm has the following advantages: firstly, the proposed algorithm has a lower hybrid error rate as we have developed computer program to generate better DNA sequences of the shortest flow path problem for the solution space. Secondly, the algorithm complexity is polynomial time growth with the size of the problem. For the shortest flow path problem with n vertices and m edges, it only needs $O(n^2)$ time complexity, faster than previous algorithms, also, the solution strands can be obtained in a certain length range. The ability based on complex biological operations in the algorithm may help us to have a better understanding of DNA computing and make a greater range of applications to complicated issues. We expect that, in future research, more nucleic acid operations will be utilized to derive computational models to solve NP-hard problems with high efficiency.

Acknowledgements

It was supported by National Key Research and Development Program of China (Grant No.2017YFC0405501) and the Open Research Fund of State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydro power Research(Grant No. SKL2018CG16, IWHR-SKL-201606). The project was also financially supported by National Natural Science Foundation of China (Grant No. 11701363, 11501359, 61702325, 11601324).

References

[1] R.P. Feynman (1961) In: Gilbert DH (ed) Minaturization. Reinhold, New York, pp 282–296.
[2] Q. Ouyang, P.D. Kaplan, S. Liu, A. Libchaber, DNA solution of the maximal clique problem, Science, 278(3) (1997) 446–449.
[3] L.M. Adleman, Molecular computation of solutions to combinatorial problems, Science, 266(5187) (1994) 1021–1024.
[4] R.J. Lipton, DNA Solution of hard computational problems, Science, 268 (5210) (1995) 542–545.
[5] S. Roweis, E. Winfree, R. Burgoyne, N.V Chelyapov, M.F. Goodman, P.W.K. Rothemund, L.M. Adleman, A sticker based model for DNA computation, J. Comput. Biol., 5(4) (1998) 615–629.
[6] J.F. Ren, Y.Z. Zhang, G. Sun, The np-hardness of minimizing the total late work on an unbounded batch machine, Asia Pac. J. Oper. Res., 26(03) (2009) 351–363.
[7] E. Winfree, F. Liu, L.A. Wenzler, N.C. Seeman, Design and self-assembly of two dimensional DNA crystals, Nature, 394 (1998) 539–544.
[8] K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, M. Hagiya, Molecular computation by DNA hairpin formation, Science, 288 (2000) 1223–1226.
[9] D.M. Xiao, W.X. Li, Z.Z. Zhang, L. He, Solving maximum cut problems in the Adleman-Lipton model, BioSystems, 82 (2005) 203–207.
[10] W.X. Li, D.M. Xiao, L. He, DNA ternary addition, Appl. Math. Comput., 182 (2006) 977–986.
[11] D.M. Xiao, W.X. Li, J. Yu, X.D. Zhang, Z.Z. Zhang, L. He, Procedures for a dynamical system on f0; 1gn with DNA molecules, BioSystems., 84 (2006) 207–216.

[12] W.L. Chang, Fast parallel DNA-based algorithms for molecular computation: the set-partition problem, IEEE Trans. Nano biosci., 6 (2007) 346–353.
[13] Z. Wang, D. Huang, H. Meng, C. Tang, A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation, Bio systems, 114(1) (2013) 1–7.
[14] M.Y. Guo, W.L. Chang, M. Ho, J. Lu, J.N. Cao, Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-based computing, BioSystems, 80 (2005) 71–82.
[15] W.L. Chang, K.W. Lin, J.C. Chen, C.C. Wang, L.C. Liu, M. Guo, Molecular solutions of the RSA public-key cryptosystem on a DNA-based computer, J. Super Comput., 61 (2012) 642–672.
[16] Z. Wang, J. Pu, L. Cao, J. Tan, A parallel biological optimization algorithm to solve the unbalanced assignment problem based on DNA molecular computing, Int. J. Mol Sci., 16(10) (2015) 25338–25352.
[17] Z.C. Wang, J. Tan, D.M. Huang, Y.C. Ren, Z.W. Ji, A biological algorithm to solve the assignment problem based on DNA molecules computation, Appl. Math Comput., 244 (2014) 183–190.
[18] Z. Wang, D. Huang, J. Tan, T. Liu, K. Zhao, L. Li, A parallel algorithm for solving the n-queens problem based on inspired computational model, Biosystems, 131(5) (2015) 22–29.
[19] X.C. Liu, X.F. Yang, S.L. Li, Y. Ding, Solving the minimum bisection problem using a biologically inspired computational model, Theor. Comput. Sci., 411 (2010) 888–896.
[20] Z.C. Wang, Y.M. Zhang, W.H. Zhou, H.F. Liu, Solving traveling salesman problem in the Adleman-Lipton model, Appl. Math Comput., 219 (2012) 2267–2270.
[21] C. Wang, J. Zhou, X. Xu, Saddle points theory of two classes of augmented Lagrangians and its applications to generalized semi-infinite programming, Appl. Math Opt., 59(3) (2009) 413–434.
[22] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness., W. H. Freeman and Company, 1979.
[23] M. Yamamura, Y. Hiroto, T. Matoba, Solutions of shortest path problems by concentration control, Lecture Notes Computer Science, 2340 (2002) 231–240.
[24] H. Zhang, Y. Wang, A new CQ method for solving split feasibility problem, Front Math China, 5(1) (2010) 37–46.
[25] R.B.A. Bakar, J. Watada, W. Pedrycz, DNA approach to solve clustering problem based on a mutual order, Biosystems, 91 (2008) 1–12.
[26] Z. Wang, Z. Ji, Z. Su, X. Wang, K. Zhao, Solving the maximal matching problem with DNA molecules in Adleman-Lipton model, Int. J. Biomath., 9(02) (2016) 1650019.
[27] H.Y. Zhang, X.Y. Liu, A CLIQUE algorithm using DNA computing techniques based on closed-circle DNA sequences, Biosystems, 105 (2011) 73–82.
[28] L. Qi, X. Tong, Y. Wang, Computing power system parameters to maximize the small signal stability margin based on min-max models, Optim Eng., 10(4) (2009) 465–476.
[29] G. Wang, X.X. Huang, J. Zhang, Levitin-Polyak well-posedness in generalized equilibrium problems with functional constraints, Pac. J. Optim., 6(2) (2010) 441–453.
[30] R.S. Braich, C. Johnson, P.W.K. Rothemund, N. Chelyapov, L.M. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, Science, 296 (2002) 499–502.
[31] Z. Wang, Z. Ji, X. Wang, et al. A new parallel DNA algorithm to solve the task scheduling problem based on inspired computational model, Biosystems, 162 (2017) 59–65.
[32] H. Chen, Y. Wang, A Family of higher-order convergent iterative methods for computing the Moore–Penrose inverse, Appl. Math Comput., 218(8) (2011) 4012–4016.
[33] C.P. Wei, P. Wang, Y.Z. Zhang, Entropy similarity measure of interval-valued in tuition is tic fuzzy sets and their applications, Inform Sciences, 181(19) (2011) 4273–4286.
[34] C. Miao, Y. Zhang, Z. Cao, Bounded parallel-batch scheduling on single and multi machines for deteriorating jobs, Inform. Process Lett., 111(16) (2011) 798–803.
[35] G. Wang, Levitin–Polyak Well-Posedness for optimization problems with generalized equilibrium constraints, J. Optimiz Theory App., 153(1) (2012) 27–41.
[36] W. Liu, C. Wang, A smoothing Levenberg–Marquardt method for generalized semi-infinite programming, Comput. Appl. Math, 32(1) (2013) 89–105.
[37] N. Zhao, C. Wei, Z. Xu, Sensitivity analysis of multiple criteria decision making method based on the OWA operator, Int. J. Intell. Syst, 28(11) (2013) 1124–1139.
[38] Q. Liu, A. Liu, Block SOR methods for the solution of indefinite least squares problems, Calcolo., 51(3) (2014) 367–379.
[39] B. Liu, B. Qu, N. Zheng, A successive projection algorithm for solving the multiple-sets split feasibility problem, Numer. Func. Anal Opt., 35(11) (2014) 1459–1466.
[40] J. Ren, G. Sun, Y. Zhang, The supplying chain scheduling with outsourcing and transportation, Asia Pac. J. Oper. Res., 34(2) (2017) 1750009.
[41] B. Wang, A. Iserles, X. Wu, Arbitrary-order trigonometric fourier collocation methods for multi-frequency oscillatory systems, Found Comput Math., 16(1) (2016) 151–181.