The First Conference and Workshop on Mathematical Chemistry
January 29-31, 2008, Tarbiat Modares University, Tehran, Iran

Solving bin-packing problem in the Adleman–Lipton model

Ardeshir Dolati[a], Mehdi S. Haghighat[b], and **Saeed Safaee[b]**

[a]Department of Mathematics, Shahed University, Tehran, Iran dolati@shahed.ac.ir
[b]Department of Mathematics, Arak University, Arak, Iran

Abstract

Adleman wrote the first paper in which it is shown that deoxyribonucleic acid (DNA) strands could be employed towards calculating solutions to an instance of the *NP-complete* Hamiltonian path problem (HPP). Lipton also demonstrated that Adleman's techniques could be used to solve the *NP-complete* satisfiability problem. In this paper, we use DNA operations presented by Adleman and Lipton, for developing a DNA algorithms to solve the bin-packing problem (BPP). BPP is one of the most important combinatorial optimization problems. In spite of the *Np-hardness* of BPP our procedures are done in a polynomial time.

1. Adleman-Lipton model

Bio-molecular computers work at the molecular level. Since biological and mathematical operations have some similarities, DNA, the genetic material that encodes the living organisms, is stable and predictable in its reactions and can be used to encode information for mathematical systems.

A DNA (deoxyribonucleic acid) is a polymer, which is strung together from monomers called deoxyribonucleotides [7,8]. Distinct nucleotides are detected only with their bases. Those bases are adenine *(A),* guanine *(G),* cytosine *(C)* and thymine *(T)*. Two strands of DNA can form (under appropriate conditions) a double strand, if the respective bases are the Watson–Crick complements of each other—*A* matches *T* and *C* matches *G*; also *3*_ end matches *5*_ end. e.g. the singled strands *5ACCGGATGTCA3* and *3TGGCCTACAGT5* can form a double strand. We also call the strand *3TGGCCTACAGT5* as the complementary strand of *5ACCGGATGTCA3*.

The length of a single strand DNA is the number of nucleotides comprising the single strand. Thus, if a single strand DNA includes 20 nucleotides, it is called a 20 mer. The length of a double stranded DNA (where each nucleotide is base paired) is counted in the number of base pairs. Thus, if we make a double strand DNA from a single stranded 20 mer, then the length of the double strand DNA is 20 base pairs, also written as 20 bp (for more discussion of the relevant biological background, refer to [2,7,8]. The DNA operations proposed by Adleman and Lipton [1,2,4]. are described below. Adleman showed how DNA strands could be applied to manipulate solutions for an instance of the *NP-complete* Hamiltonian path problem (HPP). After that Lipton [4] demonstrated that the Adleman techniques could be employed to solve the NP-complete satisfiability problem (SAT).

These operations will be used to solve the bin-packing problem (BPP) in this paper. The Adleman–Lipton model: A (test) tube is a set of molecules of DNA (i.e. a multi-set of finite strings over the alphabet *{A, C, G, T}*). Given a tube, one can perform the following operations:

1. *Merge(T_1, T_2):* for two given test tubes T_1, T_2 it stores the union $T_1 \cup T_2$ in T_1 and leaves T_2 empty.
2. *Copy(T_1, T_2)* : for a given test tube T_1 it produces a test tube T_2 with the same contents as T_1.
3. *Detect(T) :* given a test tube *T* it outputs ”yes” if *T* contains at least one strand, otherwise, outputs ”no”
4. *Separation(T_1,X, T_2)* : for a given test tube T_1 and a given set of strings *X* it removes all single strands containing a string in *X* from T_1, and produces a test tube T_2 with the removed strands;
5. *Selection(T_1, L, T_2)* : for a given test tube T_1 and a given integer *L* it removes all strands with length *L* from T_1, and produces a test tube T_2 with the removed strands;
6. *Cleavage(T, $\sigma_0\sigma_1$)* : for a given test tube *T* and a string of two (specified) symbols $\sigma_0\sigma_1$ it cuts each double strand containing $\frac{\sigma_0\sigma_1}{\sigma_0\sigma_1}$ in *T* into two double strands as follows: $\frac{\alpha_0\sigma_0\sigma_1\beta_0}{\alpha_0\sigma_0\sigma_1\beta_0} \Rightarrow \frac{\alpha_0\sigma_0}{\alpha_0\sigma_0}$ and $\frac{\sigma_1\beta_0}{\sigma_1\beta_0}$
7. *Annealing(T)* : for a given test tube *T* it produces all feasible double strands in *T*. The produced double strands are still stored in *T* after annealing;
8. *Denaturation(T)* : for a given test tube *T* it dissociates each double strand in *T* into two single strands;
9. *Discard(T)* : for a given test tube *T* it discards the tube *T*;
10. *Append(T,Z)* : for a given test tube *T* and a given short DNA singled strand *Z* it appends *Z* onto the end of every strand in the tube *T*.

2. Solving BPP by Adleman Lipton Model

In the bin-packing problem (BPP) one is given a set of *n* items, each item *i* having a fixed weight w_i, $0<w_i<W$, and a number of bins of fixed capacity *W*. The goal in the BPP is to pack the items in as few bins as possible. The BPP is an *Np-hard* problem. In this paper we solve the problem by a polynomial time algorithm.

We denote each bin as a binary *n*-tuple. Its *i*-th place is 1 or 0 depending on it contains *i*-th item or not. Each solution of the BPP is a concatenation of many binary *n*-tuples. The number of the binary *n*-tuples is clearly at most *n*. Without loss of generality we can assume that for each $1\le i\le n$, w_i =10t for some integer *t* and $W=10^{m+1}$ some integer *m*. We model the solution of the problem as follows: Let *X* and # be two DNA single strands of length *20* mer. We use *X* as delimiter. Suppose that *0,1,* A_k, and B_k for $k=1,2,\ldots,n$ are DNA single strands of length *10* mer. Let W_k be a single

strand of length w_k for $k=1,\ldots,n$. If the corresponding substrand of each bin contains substrand $B_k w_k 1 A_k$ it means the bin contains the k-th item in the solution. We use the substrand $B_k 0 A_k$ in the corresponding substrand of a bin to show that the bin does not contain the k-th item, e.g. suppose that the number of items is *3*. The strand $\# B_3 w_3 1 A_3 B_2 0 A_2 B_1 w_1 1 A_1 X B_3 0 A_3 B_2 w_2 1 A_2 B_1 0 A_1 \#$ is correspond the following solution: The items 1 and 3 are packed in a bin and the item 2 is packed in another bin. We solve the problem by the following DNA operations:

2.1. Generation all packs

Let P and Q be two tubes as $P=\{1,0,A_1\#,A_1XB_n,\#B_n,A_kB_{k-1},W_k,W_1|\quad k=2,3,\ldots,n\}$, $Q=\left\{\overline{X},\overline{\#},\overline{B_kW_k1A_k},\overline{B_k0A_k}\mid k=1,2,\ldots,k\right\}$ by the following operations we generate all packs.

2.1.1b *Merge(P.Q)*
2.1.2b *Annealing(P)*
2.1.3b *Discard(Q)*
2.1.4b *Copy(P,Q)*
2.1.5b *Cleavage(Q,#B_n)*
2.1.6b *Cleavage(Q,A_1#)*
2.1.7b *Denaturation(Q)*
2.1.8b *Separation(Q,X,T)*
2.1.9b *Separation(Q,A_1,T_1)*
2.1.10b *Discard(Q)*
2.1.11b *Discard(T)*
2.1.12b *Separation(T_1,B_n,Q)*
2.1.13b *Denaturation(P)*
2.1.14b *Separation(P,#B_n,T)*
2.1.15b *Discard(P)*
2.1.16b *Separation(T,A_1#,P)*
2.1.17b *Discard(T)*

Now, P contains solutions of the BPP which are not necessarily feasible. Each strand in Q is correspond to a packing of a bin. These operations are done in $O(1)$ time.

2.2. Removing the infeasible solutions

Here we delete the infeasible solutions. Note that, there are two reasons for infeasibility. (1) The total weight of items in a bin exceeds the capacity of the bin. (2) Existence of an item in several bins.

2.1.18b *Copy(Q,T_1)*
For d=1 to 10^m
2.1.19b *Selection(T_1,d*10+30*n,T_2)*
2.1.20b *Copy(T_2,T_3)*
2.1.21b *Discard(T_2)*
End for
2.1.22b *Separation(P,T_1,T_2)*
2.1.23b *Discard(T_2)*

2.1.24b *Discard(T_1)*
2.1.25b *Discard(T_3)*

These operations are done in *O(1)* time.

After that, we delete the solutions in which an item exists in the several bins

For d=1 to n

2.2.1c *Separation($P,B_dW_d1A_d,T_2$)*
2.2.2c *Discard(P)*
2.2.3c *Copy(T_2,P)*
2.2.4c *Discard(T_2)*
End for

These operations are done in *O(n)* time.

For d=1 to n
For r=1 to n
If d≠r then
2.2.1d Copy (Q,T_1)
2.2.2d Separation (T_1,W_d,T_2)
2.2.3d *Separation(T_2,W_r,T_3)*
2.2.4d Discard(T_1)
2.2.5d Copy(Q,T_1)
2.2.6d Separation($T_1,0A_d,T_2$)
2.2.7d Separation(T_2,W_r,T_4)
2.2.8d Separation(P,T_3,P_1)
2.2.9d Separation(P_1,T_4,T_5)
2.2.10d Merge(P,P_1)
2.2.11d Discard(T_2)
2.2.12d Discard(T_3)
2.2.13d Discard(T_4)
2.2.14d Discard(T_5)
2.2.15d Discard(P_1)
2.2.16d Discard(T_1)
End if
End for
End for

The above operations are done in $O(n^2)$ time.

2.3 Obtaining the optimum solution

Now, there are feasible solutions in *P.* Let $F=\sum_i w_i$. Here we obtain the optimum solution as follow:

For d=1 to n
*2.3.1 Selection (P,d*30*n+(d-1)*20+40 +F,T_2)*
If Detect (T_2) then break the loop
End for

These operations are done in *O(n)* time.

3. Conclusion

As the first work for DNA computing, Adleman [1] presented an idea to demonstrate that deoxyribonucleic acid (DNA) strands can be applied to solve the Hamiltonian path problem of size n (in spite of its NP-completeness)in O(n) steps using DNA molecules. Adleman's work shows that one can solve an NP-complete problem, which usually needs exponential time on a silicon based computer, in a polynomial number of steps with DNA molecules. After that, Lipton [4] demonstrated that Adleman's experiment could be used to determine the satisfiability (SAT) problem (the first NP-complete problem). Recently, the method has been used to solve the optimization problems [9,6]. Ouyang et al. [6] showed that restriction enzymes could be used to solve the NP-complete clique problem. In recent years, lots of papers have occurred for designing DNA procedures and algorithms to solve various NP-complete problems. As Guo et al. [3] pointed out, it is still important to design DNA procedures and algorithms for solving various NPcomplete problems since it is very difficult to use biological operations for replacing mathematical operations. In this paper, we propose some procedures to solve the bin-packing problem in the Adleman-Lipton model. The procedures work in $O(n^2)$ time.

References

[1] L.M. Adleman, Molecular computation of solution to combinatorial problems, Science 266 (1994) 1021-1024.

[2] Boneh, D., Dunworth, C., Lipton, R.J., Sgall, J., 1996. On the computational power of DNA. In: Discrete Applied Mathematics. Special Issue on Computational Molecular Biology, vol. 71. pp. 79–94.

[3] 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.

[4] R.J. Lipton, DNA solution of HARD computational problems, Science 268 (1995) 542-545.

[5] J.-Y. Lee, S.-Y. Shin, T.-H. Park, B.-T. Zhang, Solving traveling salesman problems with DNA molecules encoding numerical values, BioSystems 78 (2004) 39-47.

[6] Q. Ouyang, Peter D. Kaplan, S. Liu, A. Libchaber, DNA solution of the maximal clique problem, Science 278 (1997) 446-449.

[7] Paun, G., Rozenberg, G., Salomaa, A., 1998. DNA Computing: New Computing Paradigms. Springer, New York. ISBN: 3-540-64196-3.

[8] Sinden, R.R., 1994. DNA Structure and Function. Academic Press, New York.

[9] Z. Wang , D. Xiao, W. Li,1, L. He, A DNA procedure for solving the shortest path problem, Applied Mathematics and Computation 183 (2006) 79-84.