SOLUTION TO THE 0/1 KNAPSACK PROBLEM BASED ON DNA COMPUTING

[1] SANCHITA PAUL, [2] ANIRBAN DE SARKAR
[1] Lecturer, Computer Science Dept., B.I.T., Mesra, Ranchi – 835215
[2] P.G. Student (M.E., Software Engg), B.I.T., Mesra, Ranchi – 835215
E-mail: [1] sanchita07@gmail.com , [2] anirban_123007@yahoo.co.in

ABSTRACT

A lot of current research in DNA computing has been directed towards solving hard combinatorial problems. Among them Knapsack problem is one of the most common problems which have been studied intensively in the last decade attracting both theorists and practicians. Fractional Knapsack Problem is easily solvable by greedy strategy, but 0/1 Knapsack Problem is not possible to solve in this method. In this paper we have described a DNA computing model to find the optimal solution of 0/1 Knapsack problem. Here we have used a unique strategy to encode data inside DNA. We have replicated the DNAs and in the later stage took the combination of each and every DNA to form double stranded DNAs in order to find out the optimal solution. This method is a clear evidence for the ability of DNA computing solving hard numerical optimization problems, which is not a very easy work to solve with the help of traditional electronic computers.

Keywords: *0/1 Knapsack Problem, DNA Operations, DNA Encoding Method.*

1. INTRODUCTION

In 1994, Adleman (from University of Southern California) suggested that DNA could be used to solve complex mathematical problems. He solved the Hamiltonian Path Problem (the Traveling salesman problem) whose solution required finding a path from start to end going through all cities only once. This experiment demonstrates the feasibility of carrying out computations at the molecular level. He also showed us the potential power of high intensity parallel computation of DNA molecules.
In this paper we have solved the 0/1 Knapsack Problem. The proposed approach involves a DNA encoding method and some basic biological operations, which are described in section 3. Section 2 introduces us with the 0/1 Knapsack Problem, in section 4 we have described the proposed algorithm and the last part section 5 (conclusion) introduces us with the result and summary of our work.

2. 0/1 KNAPSACK PROBLEM

Actually there is a story behind this problem. A thief was robbing a store and he finds n items, suppose those are I_1, I_2,..., I_n. The j th item, I_j is worth p_j dollars and weights w_j pounds, where b_j & w_j are integers. He wants to take as valuable a load is possible, but he can carry at most W pounds in his knapsack for some integers C.
Now, what items should he take? This is called 0/1 Knapsack Problem, because each item must either he has taken or left behind; the thief can't take a fractional amount of an item.
Just like that suppose that we want to fill up a knapsack by selecting some objects among various objects (generally called items). There are n different items available and each item j has a weight of w_j and a profit of p_j. The knapsack can hold a weight of at most W. The problem is to find an optimal subset of items so as to maximize the total profits subject to the knapsack's weight capacity. The profits, weights, and capacities are positive integers.
Let xj be binary variables given as follows:
$x_j = 1$ if item j is selected;
0 otherwise.
The knapsack problem can be mathematically formulated as follows:
Max $\sum p_j x_j$, here j =1 to n,
Such that $\sum w_j x_j \leq W$, here j =1 to n,
$x_j = 1$ or 0; j = 1, 2,...., n.
This is known as the 0/1 knapsack problem, which is pure integer programming with a single constraint and forms a very important class of integer programming.

3. DNA COMPUTING FOR SOLUTION OF 0/1 KNAPSACK PROBLEM

DNA computing is a form of computing which uses DNA (Deoxy Ribonucleic Acid) &

biochemistry & molecular biology, instead of traditional silicon based computer technologies.

DNA is extremely important biological macromolecules as it is used in transfer of genetic information & the synthesis of enzymes (proteins). The basic compositions of DNA include – one phosphate group, one deoxyribose sugar & one nitrogenous base. There are 4 kinds of nitrogenous bases – Adenine (A), Guanine (G), Cytosine (C) and Thymine (T). DNA is a double helix consisting of two single strand deoxynucleotide chains running in an antiparallel configuration.

Nitrogen base + Sugar = Nucleoside + Phosphate = Nucleotide.

After determining the precise structure of DNA, many experimental methods have been invented including annealing, amplifying, melting, separating, cutting, ligating, and so on, which can be used to help discover the mechanisms of information storage and output. The basic assumptions are that the data can be encoded in DNA strands and are error-free, and that molecular biologic technologies can perform all computational operations. The models of DNA computing are based on different combinations of the following biological operations on DNA strands:

1. Melting/annealing: break apart/bond together two single DNA strands with complementary sequences.
2. Synthesis of a desired DNA strand of polynomial length.
3. Separation of the strands by length.
4. Merging: pour two (or more) test tubes into one.
5. Extraction: extract the strands that contain a given pattern as a sub string.
6. Amplifying: make copies of DNA strands by using the polymerize chain reaction (PCR).
7. Polymerization: transform a single strand that has a portion of double-stranded subsequence into an entire double-stranded molecule.
8. Cutting: cut DNA strands by using restriction enzymes.
9. Ligation: paste DNA strands with complementary sticky ends by using ligases.
10. Marking single strands by hybridization.
11. Destroying the marked strands.
12. Detection: given a tube, check if it contains at least one DNA strand.
13. Number: given a tube, count many DNA strands in it.
14. Clear: a test tube is denoted by T. Consider a particular bit position ‘‘p’’, the operation logically turns bit position p ‘‘off ’’ (value 0) on every strand in tube T.

4. PROPOSED DNA ALGORITHM TO SOLVE 0/1 KNAPSACK PROBLEM

According to the 0/1 Knapsack Problem, we have Item set I = {I_1, I_2, I_3,…., I_n}, Profit P = {P1, P2, P3,…., Pn} and corresponding weights W = {W1, W2, W3, …, Wn}, now we have to choose the items in such a way so that we can gain maximum profit, but the weight of all items should not cross the Knapsack capacity. In this case, we cannot break any item.

To solve this problem we simply take an example that we have 3 items,

Item, I = {I_1, I_2, I_3}

Weight, W = {6, 8, 10}

Profit, P = {4, 6, 8}

Now we have to find out the maximum profit.

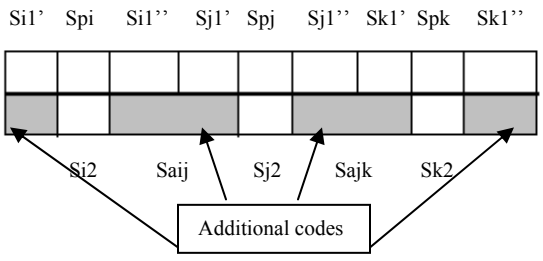
Step 1:

DNA ENCODING METHOD: For any item (suppose here we take, for first item I_1) i ($1 \le i \le n$) we use 2 DNA strands named Si1 and Si2 to encode it. Suppose Si1 is the longer strand whose length is Wi, which denotes the weight of that item. This Si1 has three parts: - first part (Si1’), center part (Spi) and last part (Si1’’). We can write it as, Si1 = Si1’ Spi Si1’’. The center part Spi denotes the profit of that item whose length is Pi. The shorter strand Si2 is the reverse complement of the center part of Si1, that is, Si2 = -h (Spi). Thus Si2 combine with the center part of Si1 to form a fragment of dsDNA.

So, Si1’ = $\lfloor$(Wi-Pi)/2$\rfloor$ and Si1’’ = Wi-Pi-$\lfloor$(Wi-Pi)/2$\rfloor$.

Just in the same way for the item 2 and item 3 we take the following DNA strands, Sj1 = Sj1’ Spj Sj1’’ and Sk1 = Sk1’ Spk Sk1’’ as weight of the other 2 strands; Sj2 = -h (Spj) and Sk2 = -h (Spk) as the reverse complement of the center part of those strands.

For any two items i, j ($1 \le i, j \le n$) we have to add an extra DNA strand known as additional code (Saij) and it will be the reverse complement of the last part of Si1 and last part of Sj1, so that Saij = -h (Si1’’ Sj1’). So we can easily say that additional code Sajk = -h (Sj1’’ Sk1’).



Take those DNA strands in different test tubes. Then,

READ (Si1, Si2, Sj1, Sj2, Sk1, Sk2, Saij, Sakj): Describes those DNA strands, which are taken in the following test tubes T1, T2, T3, T4, T5, T6, T7, T8.

Step 2:

MERGE (T1, T2; T3, T4; T5, T6): In this merge operation mix the DNA strand from test tubes T2 to T1, T4 to T3 and T6 to T5 to generate the double stranded DNAs with the help of center part's reverse complement part of every strand.

Step 3:

AMPLIFY (T1, T1', T1''),

AMPLIFY (T3, T3', T3''),

AMPLIFY (T5, T5', T5''): Amplify each DNA strands 2 times (because item is 3), then we will get 4 identical copies of each strand, this is because we have to take different possible combinations to generate dsDNAs to choose the highest weight part among all possible combinations.

Step 4:

MELT (T1, T1', T1'', T2, T2', T2'', T3, T3', T3''): By heating up all the test tubes (T1,…, T3'') dsDNAs will be converted into single stranded DNAs.

Step 5:

MERGE (T0, T7, T8, T1,..., T3''): Now mix all the copies of DNA strands and also the additional code strands in a test tube T0. As a result of mixing various double stranded DNAs will be generated randomly according to the Watson-Crick Complementary law.

For every item the shorter DNA strand that denotes the reverse complement of the center part can combine with the center part of the longer DNA strand to form dsDNAs.

For any two items, the DNA strands of encoding items i, j can combine with the additional code part to form dsDNAs.

If we see in some case there is lowest possibility to combine 2 single stranded DNAs to form a double stranded DNA due to the reason of lack of complementary pairs we ignore them by means of Pair wise Complementary Alignment technique (In this technique, we consider some value such as for mismatch of complementary pairs we assign it as 0, for matching of complementary pairs we assign it as +1 and for gap it will be 0).

Step 6:

SEPARATION BY LENGTH: After getting randomly generated all dsDNAs we separate the dsDNA strands whose length is above the Knapsack Capacity. This step will be done by means of Gel Electrophoresis.

Step 7:

DETECT (T0): After separating the dsDNAs whose length is greater than Knapsack Capacity from the test tube T0 we do the detect operation to see whether there are at least one dsDNA or not.

If detect operation returns (N) that means all the items are above the Knapsack Capacity, if it returns (Y) that means there are at least 1 item that is within the Knapsack Capacity.

Step 8:

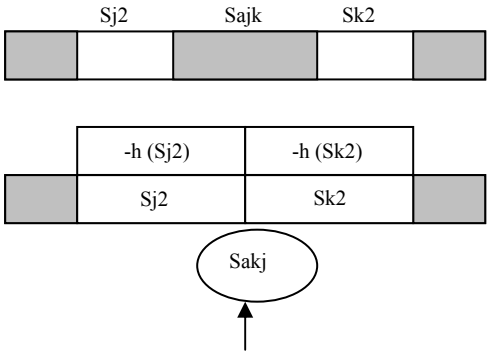
DENATURATION (T0): In this operation we heat up the test tube T0 to get single stranded DNA from the dsDNA**.**

Step 9:

SEPARATION: All the DNA strands without the additional code were in the upper portion and those strands are discarded by means of separation operation.

Step 10:

DELETION: The additional codes are deleted from the remaining DNA strands by means of Restriction Enzymes.



Deleted additional code

In this deletion operation the targeted DNA strand is suppose given in the figure below. Now we want to delete the additional code of the middle portion named as Sakj. So, at first we have to add the reverse complement of another 2 DNA strands in the test tube. Those are –h (Sj2) and -h (Sk2), which are reverse complement of strand Sj2 and Sk2. After mixing those strands Sj2 and Sk2 combine with their complementary strands and fold the additional code portion. Then we can easily cut that by means of restriction enzymes.

Step 11:

SEPARATION BY LENGTH: After deleting additional codes those DNA strands go through the Gel Electrophoresis technique (DNAs are separated by length) to find the longest DNA strands that stands for the maximum Profit.

Step 12:
SEQUENCING: Now at the last step we have to do the sequencing of the longest DNA strand to determine its nucleotides sequence, so that we can easily tell the maximum profit.

Example explained according to the DNA encoding method:
Item, I = {I_1, I_2, I_3}
Weight, W = {6, 8, 10}
Profit, P = {4, 6, 8}

Si1' Spi Si1'' = W1 = 6 = ATGCAA
Si2 = P1 = 4 = TGCA
Sj1' Spj Sj1'' = W2 = 8 = CGAATAGC
Sj2 = P2 = 6 = CTATTC
Sk1' Spk Sk1'' = W3 = 10 = ATGGCTACGA
Sk2 = P3 = 8 = CGTAGCCA
Saij = -h (Si1'' Sj1') = -h (AC) = GT
Sajk = -h (Sj1'' Sk1') = -h (CA) = TG
Saik = -h (Si1''Sk1') = -h (AA) = TT
During the time of deletion operation the additional codes will be deleted by the help of restriction enzymes. If we follow the above example, at last we get the following type of combinations such as:
>> TGCA GT CTATTC,
>> CTATTC TG CGTAGCCA.
In the above example the underlined portions are the additional codes, which will be deleted. After deletion the length of these 2 strands will be 10 and 14, which are the profit parts. Now by gel electrophoresis we choose the highest profit part that will be 14 in above example. If we mathematically crosscheck the result we can see that the profits can be the followings:– 4, 6, 8, 10, 12 and 14. So, the highest profit is obviously 14.

5. CONCLUSIONS

DNA performs millions of operations simultaneously, generates a complete set of potential solutions, conduct large parallel searches and efficiently handle massive amounts of working memory – those are the reasons why the DNA computation has arisen. But still now we have to face many problems during some biological operation, as an example temperature control is not an easy task, moreover there always comes a question of purity when millions of paths are generated and we have to choose the right one among them. So, more research and study is needed in this field. In this article we have highlighted a DNA computing model to solve the 0/1 Knapsack Problem whose time complexity is O (n^2) and space complexity is O (2^n).

REFERENCES

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

[2] W.Y. Chung, P.C. Chih, R.W. Kee, Molecular solutions to the binary integer programming problem based on DNA computing, Biosystem 83 (2005) 56–66.

[3] Z. Tiina, Formal models of DNA computing: a survey, Proc. Estonian Acad. Sci. Phys. Math. 49 (2) (2000) 90–99.

[4] L. Adleman, Computing with DNA, Sci. Am. 279 (1998) 34–40.

[5] Q.H. Liu et al., DNA computing on surfaces, Nature 403 (2000) 175–179.

[6] H.Y. Wu, An improved surface-based method for DNA computing, Biosystem 59 (2001) 1–5.

[7] L. Wang et al., Surface-based DNA computing operations: DESTROY and READOUT, Biosystem 52 (1999)

[8] S.P.A. Fodor et al., Light-directed, spatially addressable parallel chemical synthesis, Science 251 (1991) 767–773

[9] Faullhammer, D., Cukras, A.R., Lipton, R.J., Landweber, L.F., 2000. Molecular computation: RNA solutions to chess problems. Proc. Natl. Acad. Sci. U.S.A. 97, 1385–1389.

[10] Gillmor, S.D., Rugheimer, P.P., Lagally,M.G., 2002. Computing with DNA on surfaces. Surf. Sci. 500, 699–721.

[11] Lipton, R.J., 1995. DNA solution of hard computational problems. Science 268, 542–545.