DTMNP: Developing A Turing Machine Against for NP issues

Muhammad Aamir Panwar[1], Sijjad Ali khuhro[2], Tehseen Mazhar[3], Salahuddin Saddar[4], Zulfiqar Ali[5]

[1]School of Electronic Engineering Beijing University of Posts and Telecommunications, Beijing, China
[2]School of Computer Science and Technology University of Science and Technology of China, Hefei, Anhui, China.
[3]Department of Computer Science Virtual University of Pakistan.
[4]Department of Software Engineering Mehran University of Engineering and Technology, Jamshoro, Pakistan
[5]Dawood University of Engineering and Technology, Karachi, Pakistan

Abstract
A basic question that is unsolved in theoretical issues of computer science is NP problems. These problems are interlinked with P issues. In our paper we search for developing a Turing machine that can execute an algorithm for these deep complex issues. Although it is a bit hardware related solutions .our paper we focus on developing an analogy that can describe non-deterministic solutions. An aim to discuss possibilities of analogical transformation for a complex, powerful machine with solid state properties.

Keywords:
NP; Graph Representation; Nodes; a Box model.

1. Introduction

A problem in literature is TSP to understand NP issues better [1] [3]. It is a good initiative to describe applications of the automaton. Now we will discuss a variant of TSP[2]. Actually, when a sale man goes to various cities in the different sequence, we find that the cost of computation changes from city to city with the passage of time and adopted travel route. The ease of transposition we need to pay various amounts. It is possible that the higher authority of the transportation facility may vary their service plan. Let a TSP be expressed by a graph. Moreover, the prices are changing with time. If a sale man wants to move from the city a to the city b within t time span than after t period the varying cost of the edge would be constant. Its mean decision is on how to move from a to b. And the sales man would take decision resulting the routing path would be ultimate an solution at last.

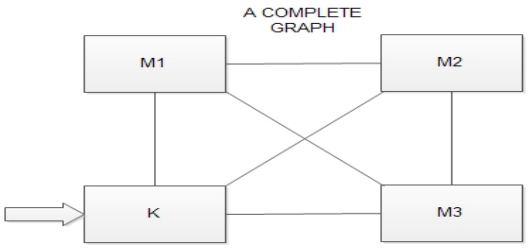

Fig.1 An invariant of TSP graph representation.

2. Graph-based representation of tsp

TSP is an example of NP issue that is taken as a problem to describe its corresponding Turing machine. If we represent a TSP using graph. Nodes are represented in tree format and leaves increase in exponential order. If every path is checked than complexity is also in exponential order. An interesting case about tree base representation is that if we merge leaf node and reverse the graph, it shows the same resulting tree.

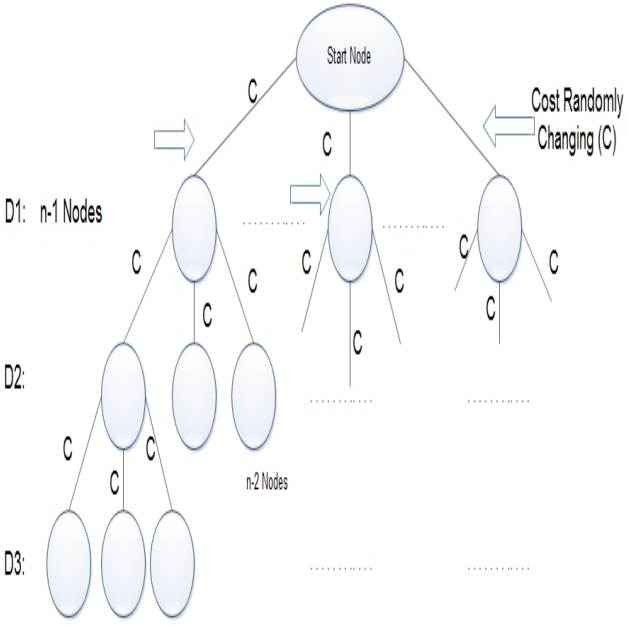

Fig. 2 Graph-Based Representation of TSP decision levels.

Manuscript received November 5, 2018
Manuscript revised November 20, 2018

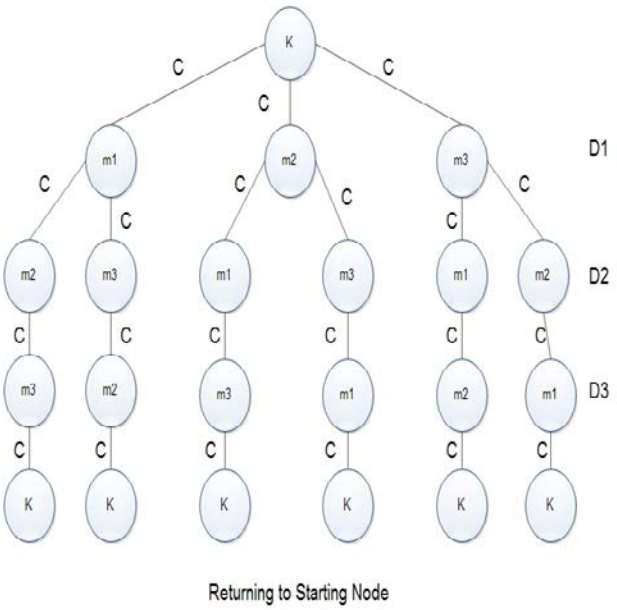

Fig. 3 Complexity increases as the leaves increases in exponential order Graph return to the starting point after reversing at *k*

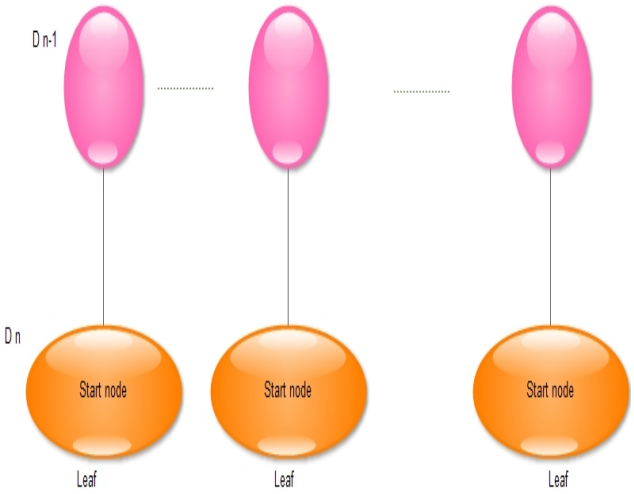

Fig. 4 every level decision depends upon its previous level decision

3. proposed scheme

Let d_k describes the decision at level *k* than optimum decision should cost minimum. Every level depends upon its next level decision. As decisions are taken at every level to decide rote to take. Therefore every decision is dependent on xt decisions. For example d_1 depends upon $d_2, d_2 \ldots d_{n-1}$.

$$D_1 \leftarrow D_2$$

$$D_{n-2}, D_{n-1}$$

D Depend on $D_2, D_3 \ldots D_{n-1}$

$$D_{n-3} \leftarrow \{D_{n-2}, D_{n-3}\}$$

If for same decision and choice level remain sthe ame.

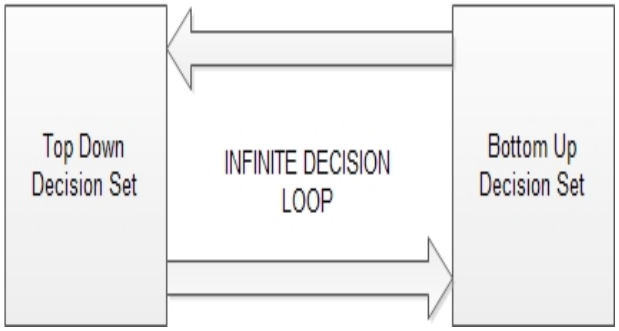

Fig. 5 Decision levels are infinite, interlinked with each other; flexibly they can be taken in bottom up or top down orders.

This shows that decision in reverse order can also be described, using same physically space and time complexity. It is not possible that this solution will be valid for every situation. The decision about every level in some specific algorithm can be ordered or in order.

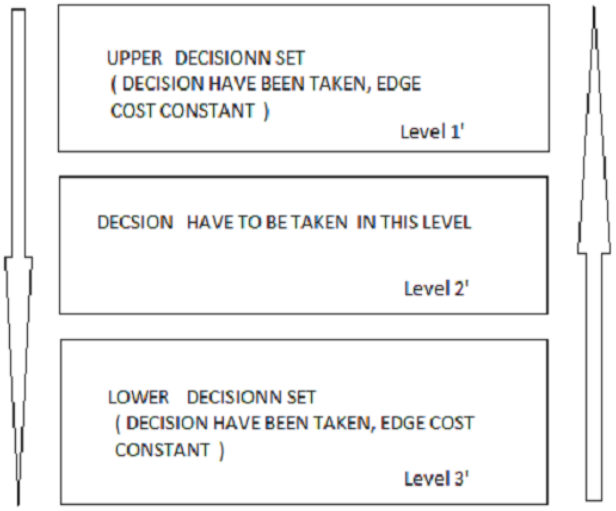

Fig. 6 Level of decision set, interlinked with each other, either in the top to bottom fashion, or bottom up. Every situation can be shown in either top down or bottom up. Both these two need a bit change it until level.

A. Box model representation

Another representation for TSP can be box model representation. The box model is a simple box structure placed in a container. Container represents the level and box represent node at that level. For example level one is represented at outermost box and box at level two contains a box of level three. This represents that k level boxes have random cost to open them concerning them. If $k = n-1$ we cannot get any box. The smallest box represents the minimal cost. The level of this model start from level 1 up to level $n-2$ .every inner level box shows the total cost of outer level, for example box at level $n-2$ shows the cost of level n and level $n-1$ .the decision at level is equal to decision at any level in time t. This show that we cannot take any decision in time t-1.therefore optimal solution cannot be taken before time $t-1$.

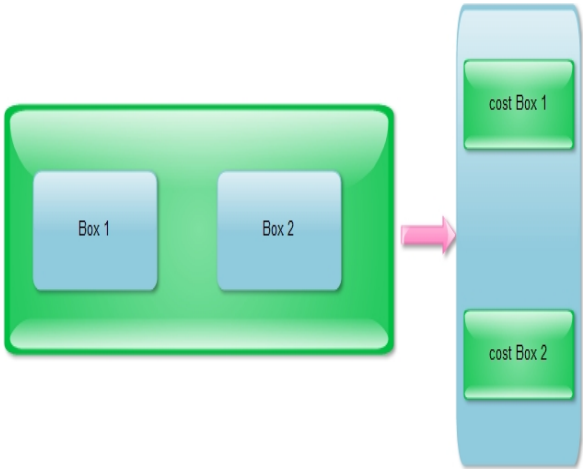

Fig. 7 showing cost through boxes, equivalent computer memory representation using arrays, * box showing decisions array, cost array.

4. issues and challenges

There can be many algorithms to solve that problem. Our issue is to find an algorithm that takes optimal discussion .Algorithm decision is optimal if its cost is similar at every level. Cost is one if algorithm ignores are does not check decision at that level. Box model represent a leaf node as small container with two empty boxes. Opening this boxes cost vary randomly along with time. Algorithm should choice minimum cost and should be able to compare costs of opening boxes.

This kind of Turing machine is not possible. Therefore a Turing machine should be checked all costs. This is same as checking all paths in tree. Mathematically it is caller permutation .If we check all possible path it shows algorithm is growing expentionaly. This problem shows output of expression if M_1 and M_2 are unknown.

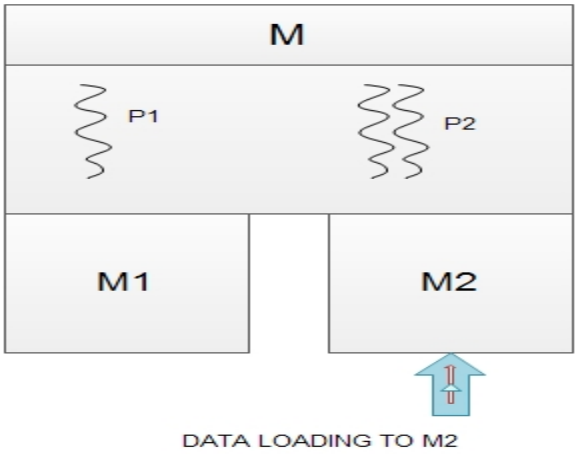

Fig. 8 Equivalent memory architecture to handle non-determinism.

5. proposed architecture of tm

Consider a memory with $m_1, m_2 .... m_k$ together physical memory addresses $p_1, p_2 ... p_k$ as its property. This property can be state machine. This property generates single stimuli depending on data. Along with change in magnitude is current position of a memory cell stimuli is strength or weakened. Magnitude of stimuli causes' memory to find that memory location that is generated it.

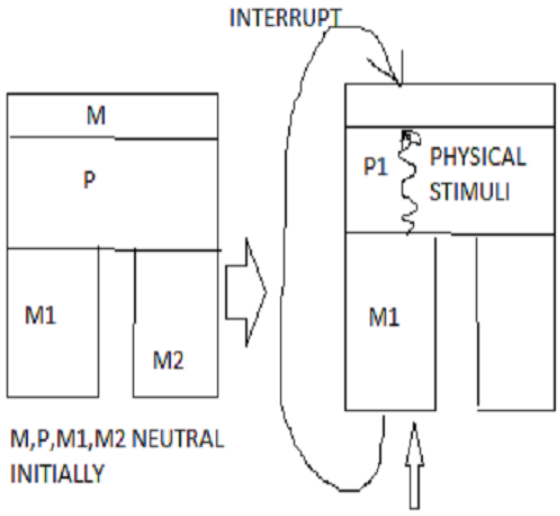

Fig.9 data uploading to M_1 , memory machine, during this phase, M hold address of M_1 .

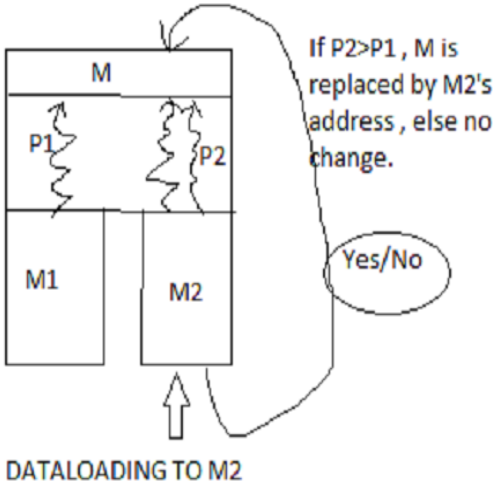

Fig. 10 M restores memory address of address that generates larger signal, containing larger data value.

A. Sorting in non-determinism

A non-deterministic data array can be sorted automatically following liquid analogy for computing machines [4]. We can take an example. Consider number of cups having water placed at a burner. Assume each cup is receiving equal heat. Boils Law Says: $pv = nrt$. This shows that p is proportional to $\frac{i}{v}$ if other are constant. Therefore non-deterministic cups order can be sorting by getting a cup number and then removing that cup from list and adding to new list. We can sort these cups using their volume, because volume is relationally proportional to pressure. Similarly non deterministic data array can be sorted using one property at least given if that property is relationally proportional to the given data array. [5]

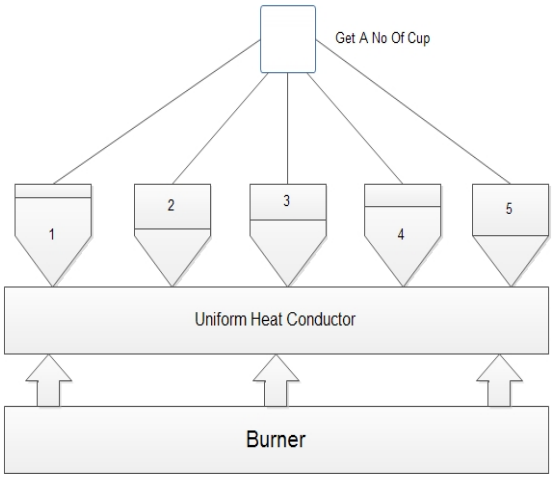

Fig.11 An example of non-uniform non deterministic ordering.

B. A Congitive machine

A powerful cognitive machine can be visioned using solid state memory and performing following algorithm listed below.

1: Load data in memory randomly
2: place each value in any cell…
3: Activate physical layer
4: Send a sequential response to cup.
5: Let the CPU ACK it and listed it in cache.
6: Physical layer generated next sequence after receiving ACK
7: Physical layer send null at the end
8: CPU performed sorting and placing data in cache until it receive null.
9: Deactivate physical layer at null.

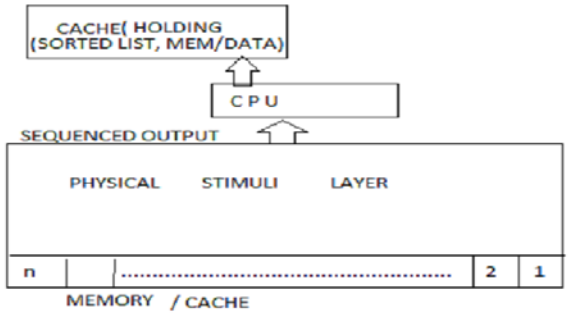

Fig. 12 machine architecture to support proposed algorithm.

6. conclusion

This paper try's to find a base for a powerful memory model reducing computational overhead to solve non deterministic issues. A high level abstraction to hardware level implemented is discussed to cope-up with non-determinism. Quantum and cognitive computing is based on new technology finding on new ways of developing computing devices. Solid state can be made powerful if they can transform parallelism naturally. This can we can solve non determinisitng issue as an inherent property of solid state devices for upcoming quantum computing and neon technology.

Acknowledgment

This research work has been carried out in state key laboratory intelligent communication, Navigation and micro-Nano system, Beijing University of posts and communications. The research work is financial supported by national high technology 863 program of china (No.2015AA124103) and national key R&D program no 2016YFB05502001.The authors are thankful to the

financial support and acknowledgment guidance and support provided by state key laboratory intelligent communication, Navigation and Micro-Nano System, BUPT.

References

[1] Applegate, D. L., Bixby, R. M., Chvátal, V., & Cook, W. J. The Traveling Salesman Problem (2006). ISBN 0-691-12993-2.

[2] Gutin, G., & Punnen, A. P. (Eds.). (2006). The traveling salesman problem and its variations (Vol. 12). Springer Science & Business Media.

[3] Fortnow, L. (2009). The status of the P versus NP problem. Communications of the ACM, 52(9), 78-86.

[4] Natschläger, T., Maass, W., & Markram, H. (2002). The" liquid computer": A novel strategy for real-time computing on time series. Special issue on Foundations of Information Processing of TELEMATIK, 8(LNMC-ARTICLE-2002-005),39-43.

[5] van Leeuwen, Jan, and Jiri Wiedermann. "Question-answering and cognitive automata with background intelligence." Technical Report Series UU-CS-2016-007 (2016).