Hardware Designation For Scaling P=NP Problem

Md. Yasin Ali Khan[1] and Md. Abu Sayed[2]

[1]Department of Computer Science and Engineering,
Chittagong University of Engineering and Technology, Chittagong-4349, Bangladesh

[2] Department of Computer Science and Engineering,
American International University-Bangladesh, Banani, Dhaka-1213, Bangladesh

***Abstract*—One of the fundamental question of computer science P=NP that is intuitively interlinked with many theoretical issues. In this paper we search for an algorithm or Turing machine equivalent that is reasonably sufficient to cope with this deep problem. Some hardware assistance may enable us to get better solution in this respect. In general we've shown some analogy that can deal with non-determinism effectively and hope to find some analogical transformation that will help making more powerful solid state computing device.**

Keywords—*travelling salesman problem ,P=NP , algorithm, turing machine, nondeterministic sorting.*

I. TRAVELLING SALESMAN AND ITS VARIANT

A well-studied problem in literature is TSP[1][2]. For easy understanding of P=NP[3] issues it is a good start. Here we'll consider a variant of TSP. In real life when a sales-man visit different cities in different order, we can see that transportation cost varies from city to city and from time to time. As for transportation facility of choice we need to pay different amount. Again authority of transportation facility may change their service plan. We simply simulate this situation. Let a TSP be represented in a complete graph. And costs of its edges are varying with time. If a salesman think to move from city K to city M within del(t) time duration, then after del(t) period, varying cost of that particular edge will be constant for ever. Meaning decision have taken on how to move from K to M. And think salesman will take decision in such way that resulting path will be optimal feasible solution at last.

II. SPECIAL REPRESENTATION

We can represent corresponding complete graph in a particular TSP problem such a way that results a tree like structure.

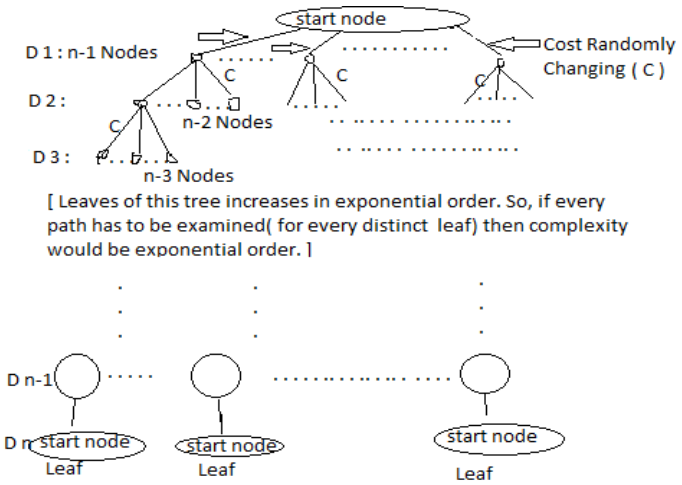
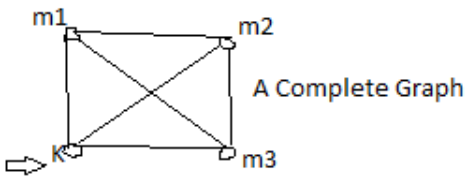
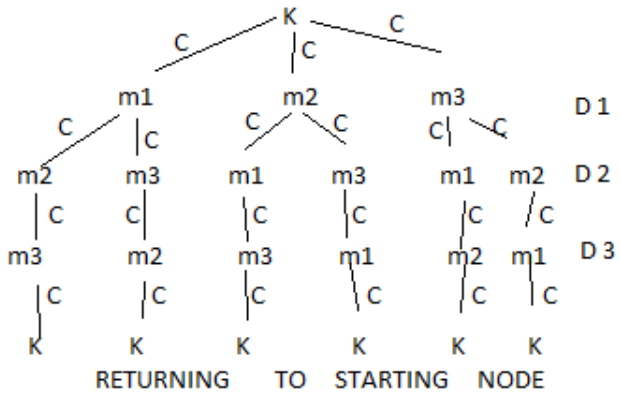


Fig : TSP Variant

An Example





If we merge k nodes and do upside down then it'll results same graph.

III. Decision Making

Let Dk represents Decision at level K. For optimal feasible solution we've to take decision in such a way that it results minimum cost. D1 depends on D2,D3,…..Dn-1. After taking decision at a level all varying values at that level become constant. So,

D1 ←D2,D3,……..,Dn-1 [D1 depends on D2,D3,…..,Dn-1]
D2 ←D3,……..,Dn-1
……..
……..
Dn-3←{Dn-2,Dn-1}
{} represents same decision, choice remains one.

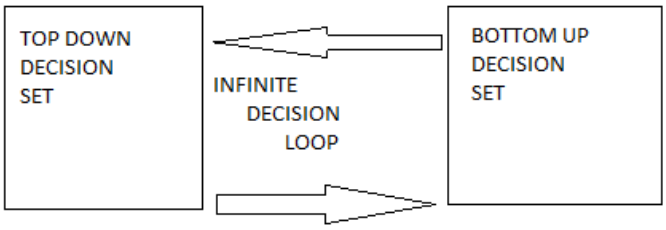
If we try to take decision in reverse order, though it is not feasible for our particular problem description as matching physical space and time restriction (time of decision taking and executing should be same) we get following situation,

Dn-1←Dn-2,……..,D1

……

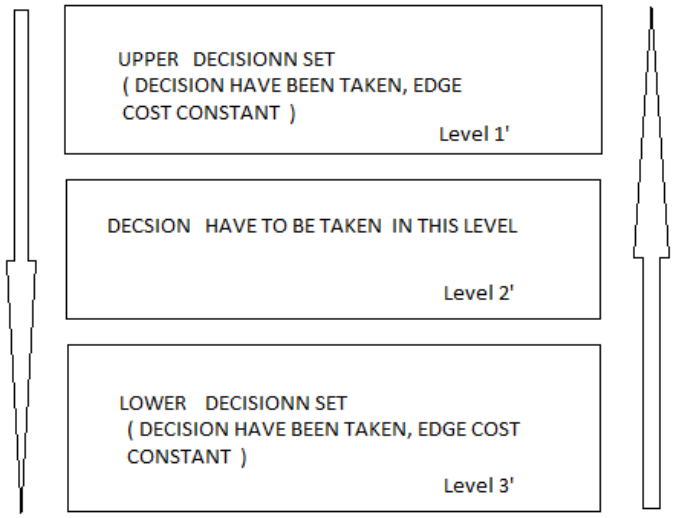
……

D2←D1



Starting in Top Down or Bottom Up we finally reach the opposite Situation. So, ono unique to take decision.

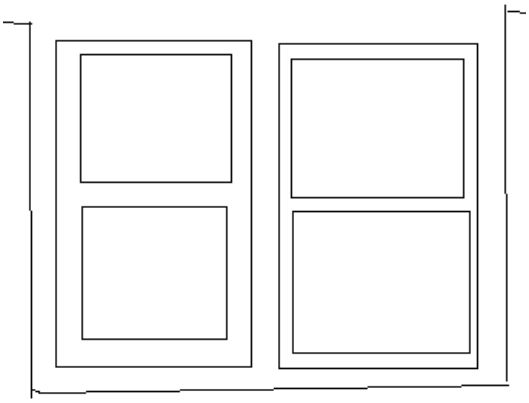
Not applicable for this particular variant : we consider another situation for experimental purposes.If using some algorithm we take decision that randomly take decison or otherwise ordered, then we get the situaiton below,



Now changing cost of a set with unit value we can simulate the above situation using Top-Down or Bottom-Up approach.

IV. Box Model Easier Representation

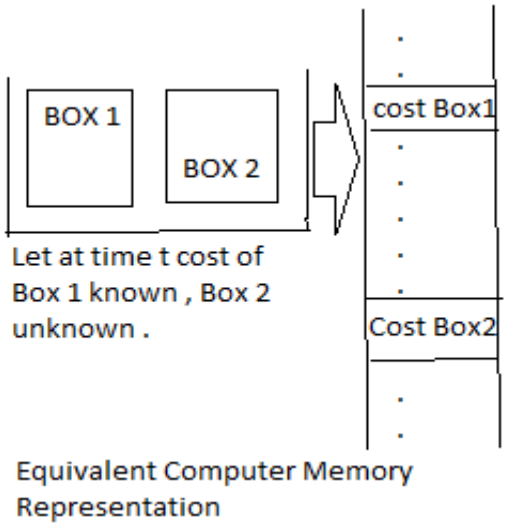
This permutation tree model can be modeled using simple box structure. A container box containing many small boxes and these small boxes again are containers. Outermost container initially open at level 0. This container containing boxes at level 1. Boxes at Level 1 containing boxes at Level 2. Boxes



at level K have different costs to open them that are also varying randomly with time parameter. If k = n-2 then we'll get no box. Here we are trying to reach smallest possible box with cost possible. We start at level 1 up to level n-2, as opening box at n-2 costs some that resembles with sum total cost of level n plus level n-1 in Tree representation. If we take decision 1 Level per time step, then while taking decision at time t in some level, no way we can take decision that will include in optimal feasible solution, before time (t-1).

V. Algorithmic Issue

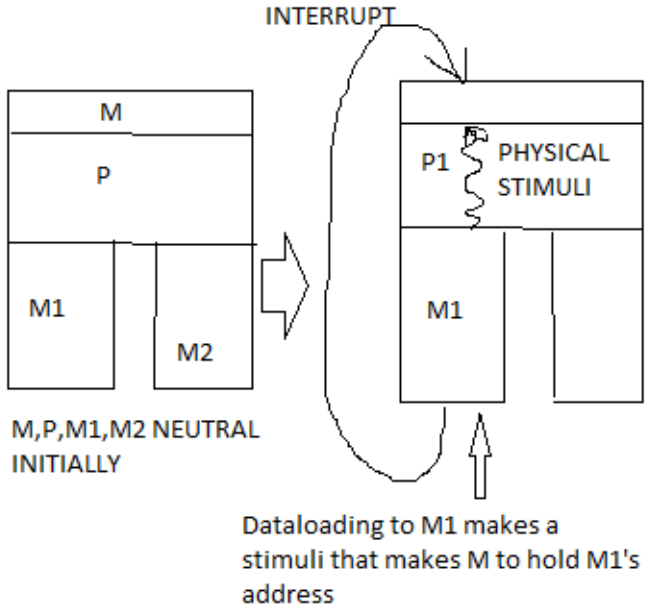
Let an algorithm B take decision in optimal way, meaning at least at one level , at least 1 cost that B will prune/ignore/do not check to take decision at that level. In our recursive box model basic building block are: a small container having two empty boxes. Cost of opening those empty boxes are

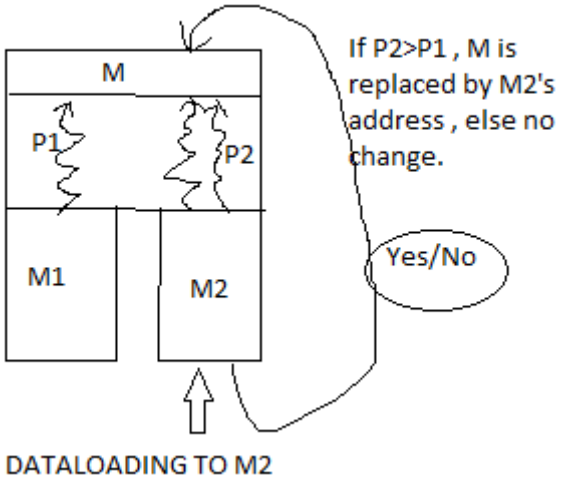


randomly changing with time. B should be able to choose one that results minimum cost knowing only cost of one. Such an algorithm (or Turing Machine) is not possible. So, we need to know/check all costs of opening empty boxes. Equivalently we've to check all possible paths in permutation tree. Checking all possible paths indicates exponential growth of algorithm. For another illustration consider 2-SAT problem (x1 and x2). Let we know x1 is true and x2 unknown, so what will be output of this expression.

VI. Memory Architecture

Consider a memory shown below,
Here M,M1,M2 Memory address, P = Physical (atomic/ nanoscale) property.

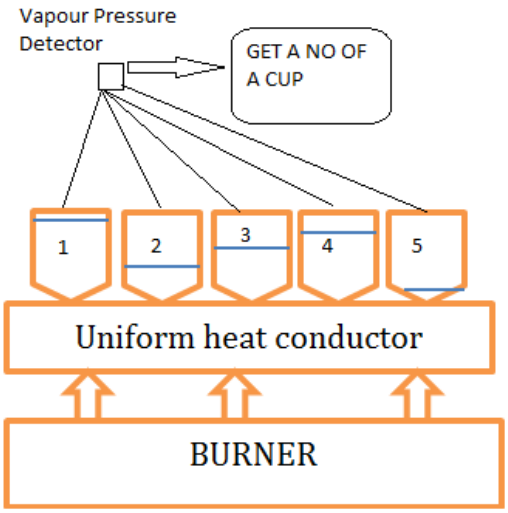




This property intuitively generate stimuli based on order/magnitude of recent data value. Greater stimuli cause M to collect address information about greater stimuli generator memory address.

VII. Nondeterministic Sorting In Nature

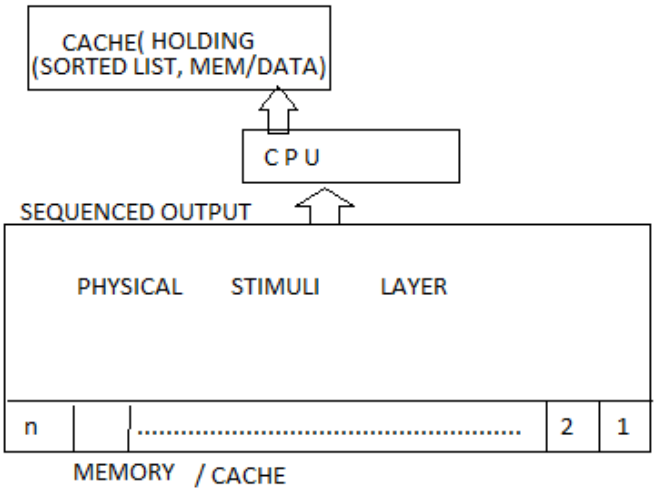
Consider a liquid computer analogy[4] ,



Cups containing water. Boyel's law says, PV=nRT.
So, p proportional to (1/V), others say constant. This indicate Non-deterministic auto sorting. After getting a cup number disconnect that cup from detector, list that cup number, in this way we can get sorted ordered cup numbers based on water volume.

VIII. Solid State Cognitive Computer

So, we can vision a powerful computer with solid state memory having properties described above with protocols listed below,



1) Load data in memory/cache in any order
2) Activate Physical Layer
3) Response (sequential) made to CPU.
4) CPU acknowledge it ,list (cache) it ,waiting for next
5) Physical layer acknowledge it, generate next sequence or NULL.
6) CPU acknowledge like step 4) or for NULL halts sorting process, returns list that is sorted.
7) Deactivate physical layer.

DISCUSSION

Tremendous advancement on different technologies like semiconductor and nanotechnology, quantum computing while new theories of basic physics, chemistry provides us new ways of computing devices. Proper analogical transformation of parallelism in natural process to solid state will enable us to cope with non-determinism inherently.

CONCLUSION

We hope this paper will help future researchers to find better memory model that will help reducing computational overhead and will provide high level abstraction and hardware software interface to cope with non-determinism effectively.

REFERENCES

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

[2] Gutin, G.; Punnen, A. P. (2006), *The Traveling Salesman Problem and Its Variations*, Springer, ISBN 0-387-44459-9.

[3] Lance Fortnow, *The status of the **P** versus **NP** problem*, Communications of the ACM 52 (2009), no. 9, pp. 78–86.

[4] Thomas Natsdiläger Wolfgang Maass Henry Markram,The "Liquid Computer": A Novel Strategy for Real-Time Computing on Time Series.