An optimized algorithm for abstraction based routing in connection oriented networks

Rouhollah Mostafaei
Computer Engineering Department
Islamic Azad University
Khoy,Iran
mostafaevn@iaukhoy.ac.ir

Ali Habiboghli
Computer Engineering Department
Islamic Azad University
Khoy,Iran
habiboghli@iaukhoy.ac.ir

Mohammad.R Maybodi
Computer Engineering Department
Amirkabir University of Technology
Tehran,Iran
meybodi@ce.aut.ac.ir

Abstract—**Communication networks are expected to offer a wide range of services to an increasingly large number of users, with a diverse range of quality of service. This calls for efficient control and management of these networks. We address the problem of quality-of-service routing, more specifically the planning of bandwidth allocation to communication demands. Shortest path routing is the traditional technique applied to this problem. However, this can lead to poor network utilization and even congestion. In this paper, we combine this technique with systematic search algorithm and heuristics which are derived from Artificial Intelligence (AI) to solve this problem more efficiently and empirical result shows that this algorithm works better than others in RAIN problem.**

Keywords- connection oriented networks, constraint-based routing, abstraction, constraint satisfaction

I. INTRODUCTION

The communication networks of the next millennium are expected to offer a wide range of services to an increasingly large number of users, with a diverse range of Quality of Service (QoS) requirements. This calls for efficient control and management of these high-speed networks. A central problem is the automatic routing of traffic through the network. Routing must be a very fast process, in order to guarantee customer satisfaction. Currently, shortest path routing is most often used to route traffic across a network. Although this ensures the best possible route for each particular demand, it can lead to ineffective use of the network as a whole and even congestion, especially in highly loaded networks.

From the routing point of view, the key resource to manage in networks is bandwidth. Therefore, in order to make better use of available network resources, there is a need for planning bandwidth allocation to communication demands, to set up routing tables (or any other route selection criterion) more purposefully. This can be achieved by the use of global information, including not only the available link capacities but also the expected traffic profile [2].

Formally, we define the problem of resource allocation in networks (RAIN) as follows: Given a network composed of nodes and bidirectional links, where each link has a given bandwidth capacity, and a set of communication demands to allocate, where each demand is defined by a triple: (source node, destination node, requested bandwidth).Wang and Crowcroft [1] have shown that the allocation of every single demand is NP-complete by itself. This creates a new situation for the networking community, as traditional routing algorithms such as shortest paths do not perform very well on this problem.

Constraint satisfaction [3][19] is a technique which has been shown to work well for solving certain NP-hard problems, and has been applied to a variety of domains [4]. A Constraint Satisfaction Problem (CSP) is defined by a triple (X, D, C), where $X=\{x_1,x_2,\ldots,x_m\}$ is a set of variables, $D=\{D_1,D_2,\ldots,D_m\}$ a set of finite domains associated with the variables and $C=\{c_1,c_2,\ldots,c_m\}$ a set of constraints. Solving a CSP amounts to finding a value of domain for each variable so that all constraints are satisfied. This may be done with a backtracking algorithm.

The RAIN problem is easily formulated as a CSP in the following way: variables are demands, the domain of each variable is the set of all routes between the endpoints of the demand, and constraints on each link must ensure that the resource capacity is not exceeded by the demands routed through it. A solution is a set of routes, one for each demand, respecting the capacities of the links. However, this formulation presents severe complexity problems. It is too expensive to compute, represent, and store the domain of a variable, i.e., all the routes that join the endpoints of a demand. Since methods such as forward checking or dynamic variable ordering require explicit representation of domains, they would be very inefficient on a problem of realistic size.

It has long been observed that the complexity of solving a problem can depend heavily on how it is formulated. Abstractions are naturally used by humans to solve problems[5]. Reducing problem complexity is a major reason for using abstraction techniques. A recent collection of papers addressing abstraction, reformulation, and approximation techniques in a variety of AI domains can be found in [6]. We can use an abstraction of the network called Blocking Islands; create a compact representation of the domains which allows the application of well-known CSP techniques such as forward checking, variable and value ordering to the RAIN problem with manageable complexity [2]. In this paper we not only use

CSP based methods, but the main idea is combining the two methods, shortest path (SP) and CSP based, respectively. The reason behind this idea is that, SP selects shortest route for each demand and consumes low network resources, however, when the number of demands increases in the same route, it may lead to congestion. In this case, we can use CSP base methods for routing. In the following section, we review some of the related work. In Section III, we illustrate how blocking islands help to route a single demand while attempting to preserve bandwidth connectivity inside the network. In Sections IV, we present proposed algorithm and some heuristics to solve the RAIN problem. Empirical results are summarized in Section V. Finally, we conclude by some future work directions.

II. RELATED WORKS

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

Surprisingly, there has been little published research on the RAIN problem. Currently, most network providers use some kind of best effort algorithm, without any backtracking due to the complexity of the problem: given an order of the demands, each demand is assigned the shortest possible route supporting it, or just skipped if there is no such route.

Operations Research (OR) techniques are also applied to the RAIN problem. Most often, a fixed number of shortest paths for each demand are pre-computed, and the problem is solved using linear programming with very large constraint systems of equations [7], [8], [9], [10]. However, because only a given number of routes are considered, these techniques are not guaranteed to find a solution if one exists. Moreover, OR techniques are not as flexible as CSP-based methods. Mann and Smith [11] search for routing strategies that attempt to ensure that no link is over-utilized (hard constraint) and, if possible, that all links are evenly loaded (below a fixed target utilization), for the predicted traffic profile. Finally, they attempt to minimize the communication costs. Genetic algorithms and simulated annealing approaches were used to develop such strategies. However, their methods do not apply well, if not at all, to highly loaded networks, mainly because the multi-criteria objective function they use cannot ensure that the hard constraint, i.e., no link is over-utilized, is respected in every case. Moreover, we think that load balancing should be viewed in terms of bandwidth connectivity and not the even distribution of the load among the links, especially in highly loaded networks, since high bandwidth connectivity allows to route additional demands without having to recompute a complete solution.

To our knowledge, the closest published work to ours is the CANPC framework [12]. It is based on the successive allocations of shortest routes to the demands, without any backtracking when an assignment fails. They propose several heuristics to order the demands (such as bandwidth ordering) to provide better solutions, i.e., to route more demands. However, results show that the methods we propose clearly outperform theirs. Abstraction and reformulation techniques have already been applied to permit more efficient solution of a CSP. Choueiry and Faltings [13] relate interchangeability to abstraction in the context of a decomposition heuristic for resource allocation.

Noteworthy, Gent et al. [15] observed that the relative behavior of algorithms on large and small problems is the same when plotted against order parameter. Comparison of different algorithms can therefore be performed on small problems, and results can be expected to scale to larger problems. Phase transition behavior has been reported in an increasing number of NP-complete problems [16].

III. AUTOMATICALLY SOLVING RAIN PROBLEM

Solving a RAIN problem amounts to solving the CSP introduced in Section I. This can be done using a backtracking algorithm with forward checking (FC). The formulation of the CSP presents severe complexity problems (see Section I). Frei and Faltings [2] introduce a clustering scheme based on Blocking Islands (BI), which can be used to represent bandwidth availability at different levels of abstraction, as a basis for distributed problem solving, given any resource requirement, blocking islands partition the network into equivalence classes of nodes. Moreover, BIs highlight the existence and location of routes at a given bandwidth level.

Nonetheless, blocking islands provide an abstraction of the domain of each demand, since any route satisfying a demand lies within the β-BI of its endpoints, where is the resource requirement of the demand. Blocking islands are used to build the β-blocking island graph (β-BIG), a simple graph representing an abstract view of the available resources: each β-BI is clustered into a single node and there is an abstract link between two of these nodes if there is a link in the network joining them. Therefore, if the endpoints of a demand are clustered in the same β-BI, there is at least one route satisfying the demand. We do not know what the domain of the variable is explicitly, i.e., we do not know the set of routes that can satisfy the demand; however, we know it is non-empty. Blocking Island Hierarchy (BIH) is recursive decomposition of BIGs in decreasing order to identify bottlenecks for different βs. A BIH can not only be viewed as a layered structure of β-BIGs, but also as an abstraction tree when considering the father-chiled relations. In the abstraction tree leaves are network elements (nodes and links), the intermediate vertices either abstract nodes or abstract links and the root vertex the 0-BI of the top level in the corresponding BIH. In fact, there is a mapping between each route that can be assigned to a demand and the BIH: a route can be seen as a path in the abstraction tree of the BIH. Thus, there is a route satisfying a demand if and only if there is a path in the abstraction tree that does not traverse BIs of a higher level than its resource requirement. This mapping of routes onto the BIH is used to formulate a

forward checking criterion, as well as dynamic value ordering and dynamic variable ordering heuristics.

Thanks to the route existence property [2], we know at any point in the search if it is still possible to allocate a demand, without having to compute a route.

Therefore, after allocating a demand, forward checking is performed first by updating the BIH, and then by checking that the route existence property holds for all uninstantiated demands. If the latter property does not hold at least once, another route must be tried for the current demand. Domain pruning (i.e., the update of the domain of the demands by propagation of the allocation decisions) is thus implicit while maintaining the BIH.

A backtracking algorithm involves two types of choices: the next variable to assign, and the value to assign to it. As illustrated above, the domains of the demands are too big to be computed beforehand. Instead, we compute the routes as they are required. In order to reduce the search effort, routes should be generated in "most interesting" order, so to increase the efficiency of the search, try to allocate the route that will less likely prevent the allocation of the remaining demands.

A natural heuristic is to generate the routes in shortest path order (SP), since the shorter the route, the fewer resources will be used to satisfy a demand. However, A kind of min-conflict heuristic, the Lowest Level (LL) that its principle is to route a demand along links clustered in the lowest BI clustering the endpoints of demand, i.e. , the BI for the highest bandwidth requirement containing the endpoints(less critical links), applied to the RAIN problem, it amounts to considers first, in shortest order, the routes in the lowest blocking island (in the BIH). Apart from attempting to preserve bandwidth connectivity, the LL heuristic allows to achieve a computational gain: the lower a BI is, the smaller it is in terms of nodes and links, thereby reducing the search space to explore. Generating one route with the LL heuristic can be done in linear time in the number of links (as long as QoS is limited to bandwidth constraints).

The selection of the next variable to assign may have a strong effect on search efficiency, as shown by Haralick [18] and others. There are some natural static variable ordering (SVO) techniques for the RAIN problem, such as first choose the demand that requires the most resources. Nonetheless, BIs allow dynamic (that is during search) approximation of the difficulty of allocating a demand in more subtle ways by using the abstraction tree of the BIH:

DVO-HL (Highest Level): first choose the demand whose lowest common father of its endpoints is the highest in the BIH (recall that high in the BIH means low in resources requirements).

The intuition behind DVO-HL is that the higher the lowest common father of the demand's endpoints is, the more constrained (in terms of number of routes) the demand is. Moreover, the higher the lowest common father, the more allocating the demand may restrict the routing of the remaining demands (fail first principle), since it will use resources on more critical links.

DVO-NL (Number of Levels): first choose the demand for which the difference in number of levels (in the BIH) between the lowest common father of its endpoints and its resources requirements is lowest.

IV. HYBRID ALGORITHM FOR RAIN PROBLEM

Now that the different parts have been examined, we can put everything together into a systematic search algorithm. Fig. 1 shows a pseudo-code for a recursive forward-checking backtracking algorithm FCRAIN as explained in [2] more carefully. In this algorithm if the life time of information flow (such as electronic post) be short, construct and maintenance of BIH is very expensive.

```
Function FCRAIN (D, Γ, H)
 if D = φ then
  (* no mor demands to allocate solution found *)
  return Γ
 else
  d = (x, y, β) ← pick a demand of D
  r ← NextBestRoute(d, H)
  repeat (* Try connection (d, r)*)
   γ ← (d, r)
   UpdateConnectionAddition(H, Γ, γ)
   (* Forward checking : verify that (d, r) does *)


    (* not prevent the remaining allocations *)
  if ExistsRouteForAll(D − {d}, H) then
       Res ← FCRAIN(D − {d}, Γ ∪ {γ}, H)
       if Res ≠ φ then
        return Res (* we have a solution *)
       end if
       UpdateConnection Removal(H, Γ, γ)
       r ← NextBestRoute(d, H)
      until r = 0
     return 0 (* Backtrack *)
    end if
   end FCRAIN
```

Fig. 1. The forward checking algorithm FCRAIN for the RAIN problem. Its input is threefold: D is the set of still unallocated demands, Γ the set of established connections, and H the current blocking island hierarchy. It returns either a set of connections Γ (a solution) or (in which case there is no solution).

Therefore we should use traditional routing algorithms in this case for routing and use BIH for connection management demands with long life time (such as video and telephone calls). For appending short life time demands, sum of resources on networks link be permanently preserved. From the BIH perspective, only one permanent circuit exists on every link. How many resources should be preserved for these connections? Now, with attempting to all problems that mentioned above, resource allocation problem differ from variety of demands and variety of service requirements. Here have been tried to introduce an algorithm that is useful for both

long life time demands and short life demands such as electronic mail, empirical results show that this algorithm work well in two cases. Initially, should be answer that, before allocating resources to demands, can we satisfy demands with current condition or no? (This can perform by information that derived from abstraction tree) and so begin resource allocating of accepted demand and updating of abstraction tree, we can check the probability of allocating of rejected demand (demands in rejection queue) again that we leave it for later works.

This algorithm performs as follow:

In the first step, we queuing all of incoming demands in two or more queues for reducing the number of backtracks when a demand is unallocated and the cost of searching is reduced. This can be performed by request queuing algorithm that is introduced in Fig 2.

In Hybrid allocation algorithm that its function is allocating resources to demand, initially checked that none of queues are empty because empty queue means that all of the demands are allocated and do not exist any demand that should be routed. There for a demand select from one of queues (first and second queue sequentially) and shortest path allocate for it. After this allocating, abstraction tree has been updated and tried next demand and this entire works repeat again for the next demand. Now, if we can't allocate a route for one of the demands in a queue, it means that the SP algorithm is not sufficient for that queue, so all of allocations ignored, then FCRAIN algorithm (Fig. 1) call for resource allocation and routing of demand of this queue. The routine of algorithm explained in Fig. 3.

Summarily we can categorize the advantage of the BIH in two fields: One is straightforward and quick answers about how exists a route that satisfy bandwidth requirement of a demand, and other, at least efficiently reducing the search space of the problem.

V. EXPERIMENTAL RESULT

Recall that in the typical scenario presented in Section I, a network/service provider must decide within a certain time decision threshold whether and how a set of demands could be accepted. A meaningful analysis of the performance of the heuristics we proposed would thus analyze the probability of finding a solution within the given time limit, and compare this with the performance that can be obtained using common methods of the networking world, in particular shortest-path algorithms.

For comparing the efficiency of different constraint solving heuristics, it is useful to plot their performance for problems of different tightness. In the RAIN problem, tightness is the ratio of resources required for the best possible allocation (in terms of used bandwidth) divided by the total amount of resources available in the network. This is an approximation of the "constraint tightness" in the CSP. Since it is very hard to compute the best possible allocation, we use an approximation, the best allocation found among the methods being compared.

We generated 22'000 instances of RAIN problems, each with at least one solution. Each problem has a randomly generated network topology of 21 nodes and 33 links, and a random set of 80 demands, each demand characterized by two endpoints and a bandwidth constraint. A solution must allocate all demands within the bandwidth capacities of the links. No other restriction was imposed on the routes. We especially supposed no hop-by-hop routing table constraints for instance.

A solution is thus applicable to a connection-oriented network such as ATM. The problems were solved and compared with four different strategies: BT-SP performs a search using the shortest path heuristic common in the world today, and incorporates backtracking to the previous in order to be able to undo bad allocations. The next search methods make use of the information derived from the BIH: BI-LL-HL uses the LL heuristic for route generation and DVO-HL for dynamic demand selection, whereas BI-LL-NL differs from the latter in using DVO-NL for choosing the next demand to allocate.

```
Function Re qurstQueui ng (x, y, β)
  if Q1 is not full then
   (* pushing each request in thefirst Queue *)
   Q1 ← du(x, y, β)
   return Q1
  else if Q2 is not full then
   (* pushing each request in thefirst Queue *)
   Q2 ← du(x, y, β)
   return Q2
  end if
 end Queuing
```

Fig. 2. Queuing algorithm arrival demands du

```
Function HybridAllo cation (Qi, Γ, H)
 if Q1 and Q2 = φ then
  (* no more demands to alloacte : solution is found *)
 return Γ
 else
  d = (x, y, β) ← pick a demand of Qi
  γ ← ShortestPa thRoute
  repeat (* Try connection (d, r) *)
   γ ← (d, r)
   UpdateConn ectionAdit ion(H, Γ, γ)
   (* not prevent the remaining allocation *)
   if ExistsRout ForAll (Qi − {d}, H) then
    Re s ← HybridAllo cation (Qi − {d}, Γ ∪ {γ}, H)
  if Re s ≠ φ then
  return Re s (* solution found *)
  else
   call Function FCRAIN
  until r ≠ φ
 end if
end HybridAllo cation
```

Fig.3.HybridAllocation algorithm initially allocates demands with SP and in the unsuccessful positions uses FCRAIN for backtracking

Hybrid Allocation that is our proposed algorithm uses LL heuristic for route generation and combinational method for choosing the next demand.

Fig.4 provides the probability of finding a solution to the problem in less than one second, given tightness of the problem (as defined above) all of three BI methods prove to perform much better than brute-force, even on these small problems, where heuristic computation (and BIH maintenance) may proportionally use up a lot of time.

Noteworthy, HL outperforms NL: NL is better than deciding which demand is more difficult to assign, and therefore achieves a greater pruning effect. However, HL outperforms Hybrid Allocation because proposed algorithm use SP in some cases. But it is better than SP, because it use BIH heuristics. The shape of the curves is similar for larger time scales. The quality of the solutions, in the terms of network resource utilization, was about the same for all methods. However, when the solutions were different, band with connectivity was generally better on those provided by BI methods.

If an operator wants to ensure high customer satisfaction, demands have to be accepted with high probability. This means that the network can be loaded up to the point where the allocation mechanism finds a solution with the probability close to 1.

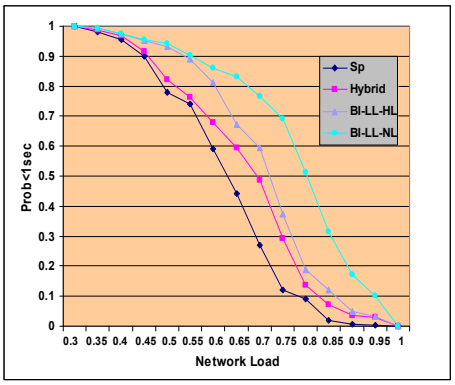

Fig.4. the probability of finding a solution within 1 second, give the tightness of the problem (22000 random problems with 21 nodes, 33 links 80 demands)

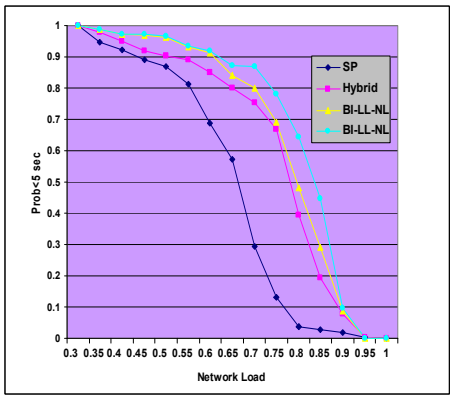

Fig.5 the probability of finding a solution within 5 second, give the tightness of the problem (22000 random problems with 21 nodes, 33 links 200 demands)

As shown in fig.4, we can see the shortest path methods, this is the case up to load of about 42% with a probability of 0.9, whereas the NL heuristic allow a load of up to about 57% and HL heuristic allows a load up to 53%, However, this rate for Hybrid Allocation is about 53%.

By theory of phase transition, this result can be expected for larger scale networks. Fig.5 shows the greater RAIN problems in which the network has 21 nodes and 33 links with 200 demands. From the curve of fig.5 we can see that for the shortest path methods, this is the case up a load of about 42% with a probability of 0.9. Whereas the NL heuristic allows a load of up to about 67% and HL heuristic allows a load of up to about 62%. However, this rate for Hybrid Allocation heuristic is 53%.

VI. CONCLUSION

The current technique for routing communication demands in a network is to select the shortest route for each particular demand. However, this strategy can lead to suboptimal routing or even highly congested network utilization as a whole. Information about the expected traffic allows making better use of network resources. However, on-line routing processes cannot make use of this knowledge, since they must be very fast to ensure customer satisfaction. Instead, bandwidth allocation can be planned in an off-line manner with this information, thanks to the systematic search algorithm that is capable of backtracking to faulty routing decisions in order to satisfy all demands. However, search must be guided carefully, since the search space to explore is exponentially large. For this reason and solving some of problems, we proposed combinational algorithms (Hybrid Allocation algorithm) that select shortest path heuristic for routing when decision time is limited and BIH base heuristics when this limitation does not exists.

This framework is technology free and independent of routing algorithms. As mentioned above this framework is concerned witch circuit switching networks. However, it can be used for problem solving of collaborative and interactive machines [17]. Further more it allows user to view the bottleneck and equipments in hierarchical abstraction and allow getting answer. This is one of the main advantages of abstraction based methods that separate them from others.

REFERENCES

[1] Zheng Wang and Jon Crowcroft: Quality-of-Service Routing for Supporting Multimedia Applications, IEEE Journal on Selected Areas in Communications, 14(7):1228-1235,September 1996.

[2] C.Frei and B.Faltings "Abstraction and Constraint Satisfaction Techniques for Planning Bandwidth Allocation" Artificial Intelligence Laboratory Swiss Federal Institute of Technology(EPFL) 2000.

[3] Kamil Vermirosky , AlgorithMS for Constraint Satisfaction ProbleMS , Master Thesis University Masarykiana ,April 2003, Pages 2- 20.

[4] R. J. Wallace, “Practical Applications of Constraint Programming,” Constraints. An International Journal, pp. 139–168, 1996.

[5] F. Giunchiglia and T.Walsh, “A Theory of Abstraction,” Artificial Intelligence, vol. 57, pp. 323–389, 1992.

[6] “Symposium on Abstraction, Reformulation and Approximation (SARA98),” Supported in Part by AAAI, Asilomar Conference Center, Pacific Grove, California, May 1998.

[7] G. R. Ash, Dynamic Routing in Telecommunications Networks, McGraw Hill, 1998.

[8] S. Cosares and I. Saniee, “An optimization problem related to balancing loads on SONET rings,” Telecommunication SysteMS, vol. 3, pp. 165–181, 1994.

[9] M. Herzberg, D. J.Wells, and A.Herschtal, “Optimal Resource Allocation for Path Restoration in Mesh-Type Self-Healing Networks,” International Teletraffic Congress (ITC), vol. 15, pp. 351–360, 1997.

[10] T.-H. Wu, Fiber Network Service Survivability, Artech House, Boston London, 1992.

[11] J. W. Mann and G. D. Smith, “A Comparison of Heuristics for Telecommunications Traffic Routing,” in Modern Heuristic Search Methods. 1996, pp. 235–254, John Wiley & Sons Ltd.

[12] B. T. Messmer, “A framework for the development of telecommunications network planning, design and optimization applications,” Technical Report FE520.02078.00 F, Swisscom, Bern, Switzerland, 1997.

[13] B. Y. Choueiry and B. Faltings, “A Decomposition Heuristic for Resource Allocation,” in Proceedings of the 11European Conference on Artificial Intelligence, ECAI-94, 1994, pp. 585–589.

[14] P. Cheeseman, B. Kanefsky, and W. M. Taylor, “Where the Really Hard ProbleMS Are,” in Proceedings of the 12 International Joint Conference on Artificial Intelligence, IJCAI-91, 1991, pp. 331–337.

[15] I. P. Gent, E. MacIntyre, P. Prosser, and T. Walsh, “Scaling Effects in the CSP Phase Transition,” in First International Conference on Principles and Practice of Constraint Programming (CP’95), 1995, pp. 70–87.

[16] I. P. Gent and T. Walsh, “Computational Phase Transitions from Real ProbleMS,” In Proceedings ISAI-95, pp. 356–364, 1995.

[17] Dean Allemang and Beat Liver. How can we communicate with computers? Abstractions:their purpose and application in telecommunications. ComTec: Technische Mitteilungen Swiss Telecom PTT, (10):948-956, 1995.

[18] R.M. Haralic and G.L. Elliott. “Increasing Tree Search Efficiency for Constraint Satisfaction Problems,” Artificial Intelligence, vol. 14, pp.263313, 1980.

[19] K.Vermirovsky , "Algorithms for Constraint Satisfaction Problems ", Master Thesis of Purdue University ,2003