On Multipath Routing with Transit Hubs

A. Sen[1], B. Hao[1], B. H. Shen[1], S. Murthy[1], and S. Ganguly[2]

[1] Dept. of Computer Science and Engineering
Arizona State University, Tempe, AZ 85287-8809, USA
{asen, binhao, bao, sudhi}@asu.edu
Tel: (480) 965-6153, Fax: (480) 965-6630
[2] Dept. of Broadband and Mobile Networks
NEC Laboratories, USA
samrat@nec-lab.com

Abstract. Empirical studies report frequent occurrences of path failure in the Internet. In providing resilience to such failures, we propose the computation of alternate backup end-to-end path that is disjoint to the default IP path. This disjoint path is created using transit hubs that can be located at diverse points in the Internet. Transit hubs provide better utilization of network resources. Assuming an IP layer routing between any two nodes, we show that the problem of computing such a disjoint path is NP-complete. We present an exact and a heuristic solution for the problem. Using routing data obtained from PlanetLab, we evaluate the efficacy of our heuristic solution.

Keywords: transit hub, disjoint path, multipath routing, network resilience.

1 Introduction

Current Internet routers select only a single path between a source and destination node. The choice of this default IP path is not left to the end hosts, instead it is left to the Administrative (AS) domain operators or on the BGP level policies. Often it is desirable for the end hosts to have better control over the route selected in context of traffic engineering and QoS controlled applications. A solution framework that has been gaining immense interest recently in the research community is using transit hubs. Transit hub routing allows routing between end hosts via a set of dedicated transit nodes that are placed at diverse locations on the Internet.

The benefits of transit hub routing are multidimensional. Transit hubs can forward packets, compute an end-to-end alternate path by chaining a set of default IP paths and thus facilitate better utilization of network resources. It provides better control over the load distribution in a network and can route packets over less congested areas. An interesting application area is bulk data transfers. Transit caches were deployed on the Internet in experiments [7], which resulted in transmission speeds of up to 47.6 Mbps during transfers of 3 TB of data between SanDiego and Urbana-Champaign. Bulk data transfer is not

readily supported by current Internet routers and would have been improbable without transit hubs. Another ongoing research effort in this direction is the Logistical Networking project [2] in which the transit hubs (called depots) offer middleware-like storage services that can be used in many applications.

Relying just on the single default IP path may lead to various end-to-end performance bottlenecks. Experimental studies conducted by authors of *detour* indicate that in many cases, alternate paths have better latency and throughput characteristics than direct default IP paths. Albeit the strong case for alternate paths, they cannot be directly constructed using existing routers as they do not provide any rerouting flexibilities. Transit hubs provide an elegant solution framework (using techniques like IP-in-IP encapsulation [4], overlay networks [1] or flexible extension headers of IPv6 datagrams) for creating alternate paths between end hosts and works seamlessly with the existing routers.

There is immense literature on traditional multipath routing problem[3, 10, 8] (others not provided due to space constraints). The advantages of multipath routing can be exploited to its fullest extent if the paths are link (or node) disjoint. Suurballe [10] presented polynomial time algorithms for computation of a pair of disjoint paths such that the sum of the path lengths is minimum. The idea of using intermediate nodes to provide a level of indirection in creating an alternate path was proposed in [1, 5, 12, 2]. These intermediate nodes have been referred with various nomenclature: as *overlay nodes* [1, 5], *rendezvous points* [12], *depots* [2], *hubs* [4] and *transit hubs* in this paper.

Our first contribution is to establish the NP-completeness of the K-transit hub routing problem. Secondly, we present an exact algorithm for solving the K-Transit hub routing problem. Our third contribution is a heuristic based solution that is effective for large networks. We evaluate the efficacy of our heuristic through extensive experimentation on *PlanetLab's Abilene* network and various randomly generated topologies.

2 Disjoint Path Routing Using Transit Hubs

The objective of the K-Transit hub routing problem is to find out if it is possible to construct a path from s to d by concatenating at most $K+1$ paths (from the set of $n*(n-1)$ paths) so that (i) each of these paths is edge-disjoint with the original s to d path and (ii) the paths are mutually edge-disjoint. In other words, can we find a set of paths $\{P_{s,v_1}, P_{v_1,v_2}, P_{v_2,v_3}, P_{v_3,v_4}, \ldots, P_{v_{k-1},v_k}, P_{v_k,d}\}$, $1 \leq k \leq K+1$ such that the concatenation of these paths will produce a path from s to d and condition (i), (ii) above are satisfied.

The idea is illustrated with the help of an example overlay network (Fig. 1) obtained from the PlanetLab. The overlay network has five nodes 1 through 5. The nodes a through j represent routers through which the overlay nodes establish paths between each other. In this example, the primary path for data transfer from overlay node 1 to node 4 is through the link 1-4 (path P_3). If the following question is asked: *"Is it possible to construct an alternate path from node 1 to 4, disjoint from the default path, by concatenating at most two mutually*

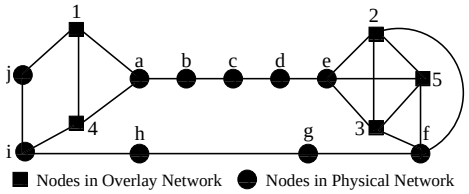

Fig. 1. Overlay Network from the PlanetLab (See legend)

Legend: Nodes in Fig 1	**Paths connecting overlay nodes**
1: PlanetLab 1, University of Arizona	P1: 1 - a - b - c - d - e - 2;
2: PlanetLab 2, Carnegie Mellon University	P2: 1 - a - b - c - d - e - 3;
3: PlanetLab 2, Duke University	P3: 1 - 4; P4: 1 - a - b - c - d - e - 5;
4: PlanetLab 2, University of Washington	P5: 2 - f - g - h - i - j - 1; P6: 2 - 3;
5: PlanetLab 2, Princeton University	P7: 2 - f - g - h - i - 4; P8: 2 - 5;
a: kscyng-dnvrng.abilene.ucaid.edu	P9: 3 - f - g - h - i - j - 1;
b: iplsng-kscyng.abilene.ucaid.edu	P10: 3 - 2;
c: chinng-iplsng.abilene.ucaid.edu	P11: 3 - f - g - h - i - 4; P12: 3 - 5;
d: nycmng-chinng.abilene.ucaid.edu	P13: 4 - 1;
e: washng-nycmng.abilene.ucaid.edu	P14: 4 - a - b - c - d - e - 2;
f: nycmng-washng.abilene.ucaid.edu	P15: 4 - a - b - c - d - e - 3;
g: chinng-nycmng.abilene.ucaid.edu	P16: 4 - a - b - c - d - e - 5;
h: iplsng-chinng.abilene.ucaid.edu	P17: 5 - f - g - h - i - j - 1;
i: kscyng-iplsng.abilene.ucaid.edu	P18: 5 - 2; P19: 5 - 3;
j: dnvrng-kscyng.abilene.ucaid.edu	P20: 5 - f - g - h - i - 4;

disjoint paths?", the answer to the question is *"yes"* because such a path can be constructed by concatenating paths P_4 and P_{20}.

3 Problem Formulation and Complexity Analysis

As indicated earlier, the input to the K-Transit hub routing problem is (i) an undirected network graph $G = (V, E)$, (ii) a set of $n * (n-1)$ paths ($|V| = n$) between every source-destination node pair (the path from node i to j may notbe same as the path from j to i) and (iii) specified source and destination nodes s and d respectively. The objective of the K-Transit hub routing problem is to find out if it is possible to construct a path from s to d by concatenating at most $K+1$ paths (from the set of $n * (n-1)$ paths) so that (i) each of these paths is edge-disjoint with the original s to d path and (ii) the paths are mutually edge-disjoint.

In order to find an answer to this question, we first remove all the edges used by the path from s to d from the graph $G = (V, E)$. Let $\mathcal{P}$ be the set of all $n * (n-1)$ paths given as the input. After removal of the edges belonging to the s to d path, many of the paths in $\mathcal{P}$ may become disconnected. We refer

to such paths as *"unavailable"* (P_{unav}). The other subset of paths in $\mathcal{P}$ are the *"available"* paths (P_{av}).

Note that here we make no attempt to consider future connection requests and the bandwidth required to satisfy them. It could very well happen that the alternate path computed by our algorithms may not be able to satisfy a connection request due to other network traffic at that time. In our approach, we merely try to ascertain whether such an alternate path exists in the network. The rationale behind this decision is that the K-Transit hub problem by itself without considering any of these parameters is NP-complete. In order to keep the problem tenable, we focus on just the computation of an alternate path by concatenating at most $K+1$ paths. In this regard, we do not consider the capacity of the links in our model.

3.1 Definitions and Notations

Definition 1. Intersection set of paths: *The intersection set of two paths P_i and P_j is the set of edges common between the paths and is denoted by $P_i \cap P_j$.*

Definition 2. Compatible Paths*: Two paths P_i and P_j are said to be compatible if their intersection set is empty.*

Definition 3. Concatenation of Paths*: If P_i is a path from s_i to d_i and P_j is a path from s_j to d_j, they can be concatenated if $d_i = s_j$ and the result of the concatenation operation is a path from s_i to d_j.*

Definition 4. K-Transit Hub Routing Problem
Instance: Given an undirected graph $G = (V, E)$, a set of triples $(s_i, d_i, P_i), 1 \leq i \leq r$, where s_i is a source node, d_i is destination node and P_i is a path from s_i to d_i and r is the number of such triples, specified source, destination nodes s and d respectively and an integer K.
Question: Suppose $\mathcal{P}_{av} = \{P_1, \ldots, P_r\}$. Is there a subset $\mathcal{P}'_{av} \subseteq \mathcal{P}_{av}$ such that:
(i) $\mid \mathcal{P}'_{av} \mid \leq K+1$
(ii) The paths in $\mathcal{P}'_{av}$ are mutually compatible, i.e., if $P_i, P_j \in \mathcal{P}'_{av}$, then $P_i \cap P_j = \emptyset, \forall i \neq j$ and
(iii) A path from s to d can be constructed by concatenating the paths in $\mathcal{P}'_{av}$.

3.2 Complexity Analysis

Theorem. The K-Transit Hub Routing Problem is NP-Complete.
Proof. It is not difficult to verify that the K-Transit hub routing problem is in NP. We show that the K-Transit Hub Routing Problem is NP-complete by a polynomial transformation from the 3SAT problem. From a given instance of the 3SAT problem, specified by a set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a set of clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$, we construct an instance of the K-Transit Hub Routing Problem in the following way. First, we classify each edge in the graph $G = (V, E)$ as a path-edge or a non-path-edge.

Definition 5. *An edge $(u, v) \in E$ is called a path-edge, if $\exists P_i = (u, v)$ where $P_i \in \mathcal{P}_{av}$. Otherwise, the edge is known as a non-path-edge.*

It may be noted that a path between a source-destination node pair may comprise of both of these types of edges. The instance $(G, \mathcal{P}_{av}, s, d, K)$ of the K-Transit hub routing problem can be generated from the instance of the 3-SAT problem in three steps: (i) We construct a subgraph G_1 of G. (ii) Paths in $\mathcal{P}_{av}$ consisting of more than one edge are specified. (iii) We augment G_1 with additional nodes and edges to construct G. It may be noted that all paths in $\mathcal{P}_{av}$ consisting of exactly one edge are specified in (i) and (iii), and all paths with more than one edge are specified in (ii).

Step 1. $\forall x_i \in \mathcal{X}$ and $\forall C_j \in \mathcal{C}$, construct a 4-node subgraph with node set $\{u_{i,j}, u'_{i,j}, v_{i,j}, v'_{i,j}\}$ and edge set $\{(u_{i,j}, u'_{i,j}), (v_{i,j}, v'_{i,j})\}$, where both edges are path-edges. For each fixed x_i, $\forall j = 1, \ldots, m-1$, we connect the subgraph for x_i, C_j and the one for x_i, C_{j+1} with two non-path-edges $(u'_{i,j}, u_{i,j+1})$ and $(v'_{i,j}, v_{i,j+1})$. Then for each x_i, we add six more vertices: $a_i, b_i, c_i, a'_i, b'_i$ and c'_i. We connect a_i, b_i, c_i with path-edges (a_i, b_i) and (a_i, c_i) and connect a'_i, b'_i, c'_i with path-edges (a'_i, b'_i) and (a'_i, c'_i). For each x_i, we add four more non-path-edges: $(b_i, u_{i,1}), (c_i, v_{i,1}), (u'_{i,m}, b'_i)$ and $(v'_{i,m}, c'_i)$. In addition, $\forall i = 1, \ldots, n-1$, we add a path-edge (a'_i, a_{i+1}) to connect the subgraphs corresponding to x_i and x_{i+1}. If the instance of the 3SAT problem is given by $\phi = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$, then the graph G_1 corresponding to ϕ is shown in Fig. 2.

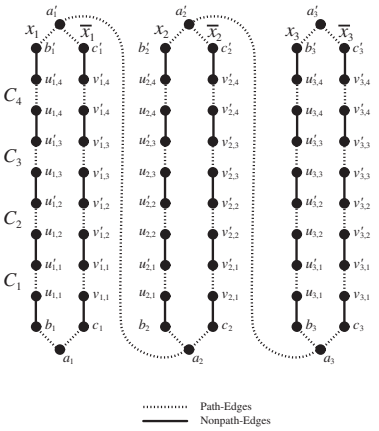

Fig. 2. Subgraph G_1 of G

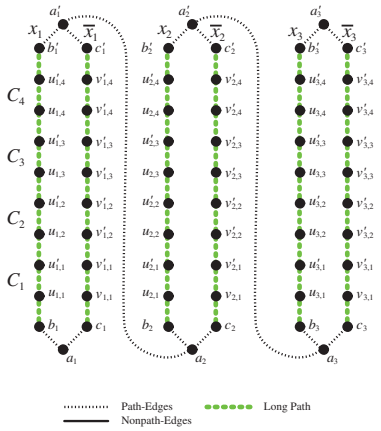

Fig. 3. Long paths consisting of more than one edge

Step 2. In this step, we specify all the paths in $\mathcal{P}_{av}$ consisting of more than one edge. $\forall i = 1, \ldots, n$, a path between b_i and b'_i: $P_{b_i} = b_i - u_{i,1} - u'_{i,1} \ldots - u'_{i,m} - b'_i$ and a path between c_i and c'_i: $P_{c_i} = c_i - v_{i,1} - v'_{i,1} \ldots - v'_{i,m} - c'_i$ are added into $\mathcal{P}_{av}$.

For the example used in Step 1, the paths specified in this step are highlighted in Fig. 3.

Step 3. This step has two parts. First, a set of nodes $\{s, w_0, w_1, \ldots, w_m, d\}$ is added to the graph G_1 (Recall that m is the number of clauses in the 3SAT instance). Second, a set of path-edges is added as follows: (i) connect s and w_0 by a path-edge (s, w_0), connect a'_n and d by a path-edge (a'_n, d) (ii) $\forall i = 1, \ldots, n, \forall j = 1, \ldots, m$, if $x_i \in C_j$, then add $(w_{j-1}, u_{i,j})$, $(u'_{i,j}, w_j)$ and if $\bar{x}_i \in C_j$, then add $(w_{j-1}, v_{i,j})$, $(v'_{i,j}, w_j)$ (iii) connect w_m to a_1 by a path-edge (w_m, a_1). This completes the construction procedure of G with all of paths in $\mathcal{P}_{av}$. The resulting graph G is shown in Fig. 4.

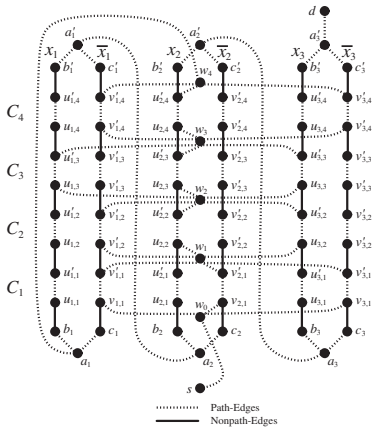

Fig. 4. Graph G

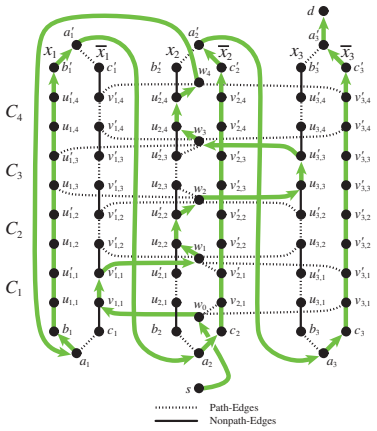

Fig. 5. The transit hub path corresponding to the truth assignment of ϕ

Set s, d to be the source node and destination node respectively in graph $G = (V, E)$.Set K=$|E|$. Construction of the instance of the K-Transit hub problem is now complete.

Claim: There exists a truth assignment satisfying the instance of the 3SAT problem, if and only if a path from s to d can be constructed in the generated instance of the K-Transit hub routing problem by concatenating at most $K+1$ mutually compatible paths.

Proof of the claim: Suppose that there is a truth assignment satisfying the instance of the 3SAT problem. We can construct a path from s to d by concatenating a subset of paths in the following way: (i)Go from s to w_0 following the path-edge between them. (ii)Each C_j, $j = 1, \ldots, m$, has at least one literal, z that has been assigned "true" by the truth assignment. This implies that we can go from w_{j-1} to w_j using the corresponding path-edges (i.e., $w_{j-1} - u_{i,j} - u'_{i,j} - w_j$ or $w_{j-1} - v_{i,j} - v'_{i,j} - w_j$). (iii)Go from w_m to a_1 using the path-edge between them. (iv)If x_1 = "true", then no edge on the path from c_1 to c'_1 has been used so far; otherwise, if x_1 = "false", then no edge on the path from b_1 to b'_1 has

been used yet. Hence, we can go from a_1 to a'_1 using one of the following two sequences of paths: if the path from b_1 to b'_1 is unused, then take (a_1, b_1), path from b_1 to b'_1, and then (b'_1, a'_1); if the path from c_1 to c'_1 is unused, then take (a_1, c_1), path from c_1 to c'_1, and then (c'_1, a'_1). (v)$\forall i, 1 \leq i \leq n-1$, go from a'_i to a_{i+1}, using the path-edge between them. (vi)$\forall i, 2 \leq i \leq n$ go from a_i to a'_i following the same process as in step (iv). (vii)Go from a'_n to d following the path-edge between them. Thus, we find a $s - d$ path, which is a concatenation of a sequence of mutually compatible paths.

To prove the converse, suppose that we can go from s to d by concatenating a sequence of mutually compatible paths. It is not hard see that we must first go from s to w_m by following the path-edges incident to w_j's. Then, from w_m, we have to go through each subgraph corresponding to x_i's, from a_i to a'_i, by using the long paths from b_i to b'_i, or the ones from c_i to c'_i. $\forall i = 1, \ldots, n$, if the path from b_i to b'_i is used, then assign x_i to be "false" and if the path from c_i to c'_i is used, then assign x_i to be "true". It is not hard to check this assignment satisfies the corresponding 3SAT problem. This completes the proof of the theorem. □

In the sample 3SAT instance ϕ considered in Steps 1, 2, and 3, the truth assignment $f(x_1) = FALSE$, $f(x_2) = TRUE$, $f(x_3) = TRUE$ satisfies ϕ. The corresponding $s - d$ path is shown in Fig. 5

4 Exact Solution for the K-Transit Hub Routing Problem

In this section, we provide an exact algorithm for the solution of the K-Transit hub routing problem. As a first step in that direction, we first construct a *Path Intersection Graph (PIG).*

Definition 6. *A* Path Intersection Graph *is the intersection graph of paths in the set $\mathcal{P}_{av}$. This is a graph $G_{pig} = (V_{pig}, E_{pig})$, where each node represents a path in the set $\mathcal{P}_{av}$ and two nodes have an edge between them, if the corresponding paths have any common edge.*

Definition 7. *An independent set (or a stable set) in a graph $G = (V, E)$ is a subset $V' \subseteq V$, such that no two nodes in V' are adjacent to each other in the graph $G = (V, E)$.*

Definition 8. *An independent set in a graph $G = (V, E)$ is called a* maximal independent set *if it is not a proper subset of any other independent set in the graph.*

As a second step towards construction of the alternate s to d path, we compute all the maximal independent sets of the path intersection graph. The maximal independent sets of the path intersection graph will correspond to the sets of maximal compatible paths in $\mathcal{P}_{av}$. Let $\{MIS_1, MIS_2, \ldots\}$ represent the set of maximal independent sets of the path intersection graph.

As a third step in the process to construct an alternate s to d path, we construct a *Path Construction Graph* corresponding to each maximal independent set $MIS_1, MIS_2, \ldots, MIS_t$ computed in the previous step.

Definition 9. *Each node in a* Path Construction Graph *corresponding to a* $MIS_i, 1 \leq i \leq t$, $G_{pcg}(i) = (V_{pcg}(i), E_{pcg}(i))$, *corresponds to a path in* MIS_i *and two nodes have an edge between them if the corresponding paths have a common terminating point, i.e., if the terminating points of a path are* v_i, v_j *and the terminating points of another path are* v_k, v_j, *then the nodes corresponding to these two paths will have an edge between them in the graph* $G_{pcg}(i)$.

Let $V_{pcg}(i, s) = \{v_{s,1}, v_{s,2}, \ldots, v_{s,p}\}$ denote the set of nodes that correspond to paths whose one terminating point is the designated source node s. Similarly, let $V_{pcg}(i, d) = \{v_{d,1}, v_{d,2}, \ldots, v_{d,q}\}$ denote the set of nodes that correspond to paths whose one terminating point is the designated destination node d. Now in the graph $G_{pcg}(i)$, we compute the shortest path between the nodes $v_{s,j}, 1 \leq j \leq p$ and $v_{d,k}, 1 \leq k \leq q$. If any of these paths have length at most $K + 1$, then it is *possible* to construct an alternate path from s to d, disjoint from the original path $P_{s,d}$ in the graph $G = (V, E)$, by concatenating compatible paths in the set $\mathcal{P}_{av}$. This process of building a path construction graph $G_{pcg}(i)$ from MIS_i followed by the computation of shortest path needs to be repeated $\forall i, 1 \leq i \leq t$, where t is the number of maximal independent sets. If a shortest path of length at most $K + 1$ cannot be found in any one of these graphs $G_{pcg}(i), 1 \leq i \leq t$, then it is *impossible* to construct an alternate path from s to d, disjoint from the original path $P_{s,d}$ in the graph $G = (V, E)$ by concatenating compatible paths in the set $\mathcal{P}_{av}$.

Algorithm 1 K-Transit Hub Routing Exact Solution$(G, \mathcal{P}_{av}, s, d, K)$

step_1	Compute Path Intersection Graph, $G_{pig} = (V_{pig}, E_{pig})$ for the paths in $\mathcal{P}_{av}$.
step_2	Compute all Maximal Independent Sets of G_{pig}, $\mathcal{MIS} = \{MIS_1, MIS_2, \ldots, MIS_t\}$.
step_3	Compute a subset $\mathcal{MIS}' \subseteq \mathcal{MIS}$, such that all elements of $\mathcal{MIS}'$, contain at least one path whose terminating point is s and another path whose terminating point is d.
step_4	Repeat steps 5-7 for each elements MIS_i of $\mathcal{MIS}'$,
step_5	Compute the Path Construction Graph $G_{pcg}(i)$ corresponding to MIS_i
step_6	Let $V_{i,s}$ be the set of nodes in $G_{pcg}(i)$ that corresponds to those paths whose one terminating point is s and $V_{i,d}$ be the set of nodes in $G_{pcg}(i)$ that corresponds to those paths whose one terminating point is s. Repeat step 5 for each element $v_{i,s} \in V_{i,s}$ and for each element $v_{i,d} \in V_{i,d}$
step_7	Compute the shortest path from $v_{i,s}$ to $v_{i,d}$. If the shortest path length is at most $K + 1$, then an alternate path from s to d using compatible paths from the set $\mathcal{P}_{av}$ exists. EXIT from the loop.
step_8	If no path of length at most $K+1$ can be found in any of the combinations of $v_{i,s}$ and $v_{i,d}$, then an alternate path from s to d using compatible paths from the set $\mathcal{P}_{av}$ does not exist.
step_9	EXIT

4.1 Algorithm Analysis

The algorithm first computes the Path Intersection Graph of the set of available paths $\mathcal{P}_{av}$ and then computes all maximal independent sets of this graph. The maximal independent sets give the set of compatible paths that can be concatenated for constructing the path from the source s to destination d. In step 5 of the algorithm the Path Construction Graph is constructed and in step 7, the shortest path between a $v_{i,s}$ and $v_{i,d}$ is computed. Since the process is repeated for all maximal independent sets that contains a $v_{i,s}$ and $v_{i,d}$ and for all $v_{i,s}$ and $v_{i,d}$, if a path between s to d can be obtained by concatenating at most $K+1$ compatible paths in the set $\mathcal{P}_{av}$, this process will find it. This ensures the correctness of the algorithm.

For generating all maximal independent sets of a graph, algorithms such as the ones presented in [11] and [6] can be used. Both the algorithms produce the maximal independent sets one after another in such a way that the delay between generation of two consecutive maximal independent sets is bounded by a polynomial function of the input size. The computation complexity of the algorithm in [11] is $O(n*m*\alpha)$ and the algorithm in [6] is $O(n^3*\alpha)$ where n, m and α represents the number of nodes, edges and the maximal independent sets of the graph respectively. We use the algorithm in [6] for generating all maximal independent sets in step 2 of the K-Transit hub routing algorithm.

Let α, β represent the number maximal independent sets of the path intersection graph and the paths (i.e. $|\mathcal{P}_{av}|$) respectively. The worst case computational complexity of step 1 of the algorithm is $O(\beta^2)$, step 2 is $O(\beta^3*\alpha)$ and step 3 is $O(\beta^2*\alpha)$. Thus, the overall complexity of the algorithm is $O(\alpha*\beta^4)$.

5 Heuristic Solution for the K-Transit Hub Routing Problem

The main overhead involved in the exact algorithm is in the computation of all the maximal independent sets of the path intersection graph. In this section, we present a heuristic solution using randomization technique for the K-Transit Hub Routing problem that produces a solution with high probability. The complexity of the solution is bounded by a polynomial function of the number of nodes in the overlay network.

5.1 Complexity Analysis

As in the exact algorithm, the heuristic solution starts by determining the Path Intersection Graph of the available paths $\mathcal{P}_{av}$. However, instead of finding all maximal independent sets involving the two nodes(paths) v_s and v_d that terminate in s and d respectively, the algorithm randomly generates a maximal independent set for each pair-wise combination of v_s and v_d. The random generation procedure first includes the two nodes v_s and v_d into a working set $\mathcal{MIS}$ of independent nodes. It then randomly selects a node from all the remaining

Algorithm 2 Heuristic for K-Transit Hub Routing Problem $(G, \mathcal{P}_{av}, s, d, K)$

step_1 Compute Path Intersection Graph $G_{pig} = (V_{pig}, E_{pig})$ for paths in P_{av}.

step_2 Compute set of nodes, $V_s \in V_{pig}$ that correspond to the paths whose one terminating point is s and $V_d \in V_{pig}$ as nodes that correspond to the paths whose one termination point is d.

step_3 Repeat steps 4 through 7 for every node-pair $(v_s, v_d) \in V_s \times V_d$.

step_4 Construct a maximal independent set with two nodes v_s and v_d, $\mathcal{MIS} = \{v_s, v_d\}$

step_5 Let $\mathcal{NNS}(S)$, the "Non-Neighborhood Set" of S be defined as $\mathcal{NNS}(S) = V_{pig} \backslash (\mathcal{N}(S) \bigcup S)$, where $\mathcal{N}(S)$ represents the neighborhood set of S. Select with equal probability a node $v \in \mathcal{NNS}(S)$. Augment the maximal independent set, $\mathcal{MIS} = \mathcal{MIS} \bigcup \{v\}$.

step_6 If $\mathcal{MIS}$ is not a maximal independent set, go back to step_5. Otherwise, form the Path Construction Graph G_{pcg}. Compute $V_{i,s}, V_{i,d} \in V(G_{pcg})$ as the set of nodes corresponding to paths having one terminating point in s, d respectively.

step_7 Compute the shortest path between every pair $v_{i,s} \in V_{i,s}$ and $v_{i,d} \in V_{i,d}$. If there exists a shortest path of length at most $K+1$ between any $v_{i,s} \in V_{i,s}$ and $v_{i,d} \in V_{i,d}$, then an alternate path from s to d using compatible paths from the set $\mathcal{P}$ exists. EXIT from the loop.

step_8 If none of the combinations of v_s and v_d report a path of length at most $K+1$, then there is no alternate path from s to d using compatible paths from the set $\mathcal{P}_{av}$.

step_9 EXIT

non-neighboring nodes of $\mathcal{MIS}$ in the Path Intersection Graph and includes it into $\mathcal{MIS}$. This process is continued until $\mathcal{MIS}$ is maximally independent.

Let β is the number of nodes in the Path Intersection Graph. Step 1 has worst case computational complexity of $O(\beta^2)$. Steps 5 and 6 perform $O(\beta^2)$ operations in the worst case to compute a Maximal Independent Set. Step 7 of the algorithm performs $O(\beta^2)$ operations to check if there exist compatible paths in the Path Construction Graph between the source node and the destination node. Thus, the overall complexity of the algorithm is $O(\beta^4)$.

5.2 Performance of the Heuristic Solution

To evaluate the performance of our proposed exact and heuristic solutions, we conducted experiments for the K-transit hub routing problem on randomly generated topologies and the Abilene network.

The problem instances were generated in 3 steps:

Step 1. Georgia-Tech Internet Topology Model topology generator was used to generate the random physical layer topologies having 30 nodes and average node degree varying between 2 and 6.

Step 2. A subset of these nodes were randomly chosen with uniform distribution as the set of overlay nodes.

Step 3. Shortest Paths using Dijkstra's algorithm between every pair of the overlay nodes were computed. These act as the primary paths in our experiments.

One of the metrics used for the evaluation of the performance of the heuristic is the *success ratio*. *Success ratio* is the ratio of number of source-destination pairs for which a path was found by the algorithm to total number of source-destination pairs.

Three sets of experiments were conducted to study the performance of the heuristic solutions. In the first set (6(a)), the number of overlay nodes were varied from 3 to 7 and success ratio of both the exact and the heuristic solutions were measured for all source-destination pairs. The value of K was chosen to be greater than the number of overlay nodes. In the second set of experiments 6(b), different physical topologies consisting of 30 nodes were chosen with varying average node degrees. In each case, 6 nodes were chosen to be overlay nodes and the success ratio of the exact and heuristic algorithms were measured for all the 30 source-destination pairs. The aim of this experiment was to study the impact of the average node degree on the performance of the algorithm. The third set of experiments (7) were conducted with two data sets. For various values of K, the success ratio of both the algorithms were recorded. The physical topology had 30 nodes with an average node-degree of 4 and the overlay structure had 7 nodes.

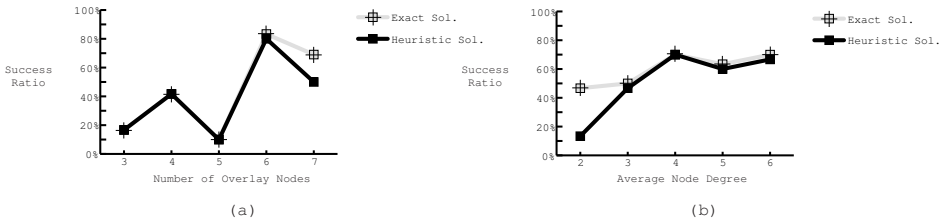

Fig. 6. Performance of the Heuristic Solution, (a) Success Ratio vs. Number of overlay nodes; (b) Success ratio vs. Average node degree

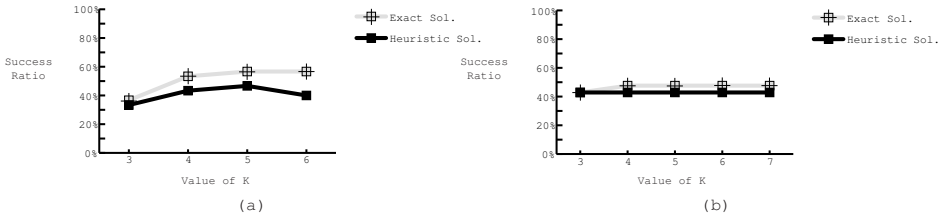

Fig. 7. Performance of the Heuristic Solution, (a) Success ratio vs. Value of K for instance 1; (b) Success ratio vs. Value of K for instance 2

In most of the cases, the success ratio of the heuristic was close to the exact algorithm. Increasing average node-degree in the physical topology (6(b)) has a positive effect on finding alternate paths in the overlay. The success ratio for both the heuristic and exact algorithms increase with increased average node-degree.

The results (7(a), (b)) indicate that the performance of the heuristic solution is not significantly dependent on the value of K, the number of paths that are allowed to be concatenated to construct the source to destination path. In all these experiments, the execution times of the heuristic and exact solution were noted. In many instances, the execution time of the exact solution was almost 1000 times more than that of the heuristic. We thus conclude that our heuristic technique almost always produces a very high quality solution in a fraction of time needed to find the exact solution.

6 Conclusion

In this paper, we consider the problem of computing an alternate path that is disjoint to the default IP path. Such an alternate path can be computed by exploiting transit hubs placed at opportunistic locations on the Internet. We show that the problem of finding such a path with constraint on the number of transit hubs is NP-complete. We provide an exact and approximate solution for the problem. Our experimentations demonstrate that our heuristic produces near optimal solution for most of the instances in a fraction of time needed to find the optimal solution.

References

1. D. Anderson, H. Balakrishnan, M. Kaashoek and R. Morris, "Resilient Overlay Networks," *In Proc. 18th ACM SOSP,* Canada, October 2001.
2. M. Beck, J. Dongarra, J. Plank and R. Wolski, Logistical Network Project, *http://loci.cs.utk.edu/scidac.*
3. I. Cidon, R. Rom and Y. Shavitt, Analysis of Multi-path Routing, *IEEE/ACM Trans. on Networking*, vol. 7, no. 6, pp. 885-896, 1999.
4. R. Cohen and G. Nakibli, On the computational complexity and effectiveness of "N-hub shortest path routing", *Proc. of IEEE Infocom*, 2004.
5. N. Feamster, D. Anderson, H. Balakrishnan and M. Kaashoek, "Measuring the Effects of Internet Path Faults on Reactive Routing," *In Proc. of ACM SIGMETRICS,* San Diego, CA, June 2003.
6. D. S. Johnson, M. Yannakakis and C. H. Papadimitriou, On Generating All Maximal Independent Sets, *Information Processing Letters*, vol 27, pp. 119-123, 1988.
7. T. Kosar, G. Kola and M. Livny, A Framework for Self-optimizing, Fault-tolerant, High Performance Bulk Data Transfers in a Heterogenous Grid Environments, *Proc. ISPDC* 2003.
8. S. Lee and M. Gerla, Split Multipath Routing with Maximally Disjoint Paths in Ad-hoc Networks, *Proc. of IEEE ICC* 2001.
9. C. Perkins, "IP encapsulation within IP," *IETF RFC 2003,* October 1996.
10. S. Suurballe and R. Tarjan, A Quick Method for Finding Shortest Pair of Disjoint Paths, *Networks*, vol. 14, pp. 325–336, 1984.
11. S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A New Algorithm for Generating All Maximal Independent Sets, *SIAM Journal of Computing*, vol. 6, pp. 505-517, 1977.
12. S. Z. S. S. I. Stoica, D. Adkins and S. Surana, "Internet indirection infrastructure," *in Proceedings of ACM SIGCOMM 2002,* August 2002.