A Link-Based Storage Scheme for Efficient Aggregate Query Processing on Clustered Road Networks

Engin Demir, Cevdet Aykanat, B. Barla Cambazoglu

Abstract

The need to have efficient storage schemes for spatial networks is apparent when the volume of query processing in some road networks (e.g., the navigation systems) is considered. Specifically, under the assumption that the road network is stored in a central server, the adjacent data elements in the network must be clustered on the disk in such a way that the number of disk page accesses is kept minimal during the processing of network queries. In this work, we introduce the link-based storage scheme for clustered road networks and compare it with the previously proposed junction-based storage scheme. In order to investigate the performance of aggregate network queries in clustered spatial networks, we extend our recently proposed clustering hypergraph model from junction-based storage to link-based storage. We propose techniques for additional storage savings in bidirectional networks that make the link-based storage scheme even more preferable in terms of the storage efficiency. We evaluate the performance of our link-based storage scheme against the junction-based storage scheme both theoretically and empirically. The results of the experiments conducted on a wide range of road network datasets show that the link-based storage scheme is preferable in terms of both storage and query processing efficiency.

Index Terms

Storage management, spatial databases and GIS, clustering, hypergraphs.

I. Introduction

A. Motivation

An important issue involved in large-scale spatial network database design is storage modeling, which directly affects the performance of querying processing on spatial network data. Spatial networks, which include network elements such as data nodes and their pairwise connections, are generally represented as directed graphs, where vertices correspond to nodes and edges correspond to connections between the nodes. In this work, without loss of generality, we focus on road networks, a typical type of spatial networks. A road network is represented as a two-tuple $(\mathcal{T}, \mathcal{L})$, where $\mathcal{T}$ and $\mathcal{L}$ respectively indicate the junctions and the road segments (links) between pairs of junctions.

In road networks, search queries form a major portion of the overall cost of daily queries since these networks have static topologies and hence the maintenance queries are rare. Basic search queries include aggregate network queries, i.e., route evaluation and path computation queries, which are processed to derive an aggregate property over the network elements. In processing aggregate network queries, a vast amount of data should be iteratively accessed and retrieved from the disk to the memory. Concurrently accessing the data of the connected elements is expected to decrease the disk access cost of the queries.

The disk access cost in large databases is higher than the cost of in-memory computations even in multi-dimensional data processing. If the access frequencies of the network elements can be modeled from previous query logs, storing frequently accessed data in the same disk pages can decrease the total disk access cost in query processing. This can be achieved by data clustering, with an upper bound (equal to the disk page size) on individual cluster sizes. Hence, clustering can produce an efficient data allocation for large databases.

Computer Engineering Department, Bilkent University, Ankara 06800, Turkey
e-mail: {endemir,aykanat,berkant}@cs.bilkent.edu.tr

In the literature, for efficient query processing in road networks, extensive studies have been carried out on indexing [12], [14]–[16], [27] and data allocation schemes [10], [18], [25]. Efficient storage schemes should also be adopted to increase the query performance along with the data allocation schemes and efficient index structures. However, so far, disk storage schemes are not explored separately from indexing.

B. Related Work

There are a few works that study the disk-based storage schemes of road networks. In the storage scheme of [11], links of the network are stored in a separate link table. The link table is clustered in disk pages such that pages store the links of which origin nodes are closely located. This approach is based on spatial locality, and clustering does not utilize the connectivity information.

In the following studies, the importance of connectivity information in networks is realized, and graph clustering models [18], [25] are proposed to partition the data into disk pages. In [18], the authors propose the junction-based storage scheme in which each record corresponds to a junction together with its connectivity information in the network. They evaluate their graph clustering model for the junction-based storage scheme by both uniform access frequencies and frequencies extracted from the previous query logs, yielding better performance results. In [25], in clustering the network, the minimum number of disk pages is achieved based on the assumption that records have fixed size. The graph clustering models for the junction based-storage scheme are used in the recent spatial query processing and clustering papers [1], [13], [26], [27].

Recently, in [10], we showed that graph clustering models do not correctly capture the disk access cost of aggregate network operations. We proposed a clustering hypergraph model for the junction-based storage scheme. In this model, records are clustered in disk pages by hypergraph partitioning, where the partitioning objective corresponds to minimizing the disk access cost of aggregate network operations in network queries.

C. Contributions

In this work, our contributions are five-fold. First, we propose the link-based storage scheme relying on the dual network concept, originally proposed in [6] and later used in [23], [24] to express relations between consecutive links along paths. In this storage scheme, each record stores the data associated with a link together with the link's connectivity information. Second, we extend our recently proposed clustering hypergraph model from the junction-based storage scheme to the link-based storage scheme. Third, we present a detailed comparative analysis on the properties of the junction- and link-based storage schemes and show that the link-based storage scheme is more amenable to clustering. Fourth, we introduce storage enhancements for bidirectional networks. We show that the link-based storage scheme is more amenable to our enhancements than the junction-based storage scheme and results in better data allocation for processing aggregate network queries. Finally, extensive experimental comparisons are carried out on the effects of page size, buffer size, path length, record size, and dataset size for the junction- and link-based storage schemes. Each parameter is explored for both storage schemes, and relative improvements are observed on real-life datasets with synthetic queries. According to the experimental results, the link-based storage scheme can be a good alternative to the widely-used junction-based storage scheme.

The rest of this paper is organized as follows: Section II presents some background material. In Section III, the link-based storage scheme and its advantages over the junction-based storage scheme are discussed. Section IV presents our clustering hypergraph model for the link-based storage scheme. Section V overviews the experimental framework and presents the experimental results. Finally, we conclude the paper in Section VI.

II. Preliminaries

A. Hypergraph Partitioning

The proposed clustering model heavily relies on hypergraph partitioning. Here, we provide a brief description of hypergraphs and hypergraph partitioning. A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ consists of a set of

vertices $\mathcal{V}$ and a set of nets $\mathcal{N}$ [4]. Each net $n_j \in \mathcal{N}$ connects a subset of vertices in $\mathcal{V}$, which are referred to as the pins of n_j and denoted as Pins(n_j). The size of a net n_j is the number of vertices connected by n_j, i.e., $|n_j| = |\text{Pins}(n_j)|$. The size of a hypergraph $\mathcal{H}$ is defined as the total number of its pins, i.e., $|\mathcal{H}| = \sum_{n_j \in \mathcal{N}} (|n_j|)$. Each vertex v_i has a weight $w(v_i)$, and each net n_j has a cost $c(n_j)$.

$\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ is a K-way vertex partition if each part $\mathcal{V}_k$ is non-empty, parts are pairwise disjoint, and the union of parts gives $\mathcal{V}$. In a given K-way vertex partition Π, a net is said to connect a part if it has at least one pin in that part. The connectivity set $\Lambda(n_j)$ of a net n_j is the set of parts connected by n_j. The connectivity $\lambda(n_j) = |\Lambda(n_j)|$ of a net n_j is equal to the number of parts connected by n_j. If $\lambda(n_j) = 1$, then n_j is an internal net. If $\lambda(n_j) > 1$, then n_j is said to be cut.

In K-way hypergraph partitioning, the partitioning objective is to minimize a cutsize metric defined over the cut nets. In the literature, a number of cutsize metrics are employed. In connectivity-1 metric, which is widely used in VLSI layout design [2], [9] and in scientific computing [3], [7], [20], [21], each net n_j contributes $c(n_j)(\lambda(n_j) - 1)$ to the cutsize of a partition Π. That is,

$$\text{Cutsize}(\Pi) = \sum_{n_j \in \mathcal{N}} c(n_j)(\lambda(n_j) - 1). \tag{1}$$

The partitioning constraint is to maintain an upper bound on the part weights, i.e., $W_k \leq W_{\max}$, for each $k = 1, \ldots, K$, where $W_k = \sum_{v_i \in \mathcal{V}_k} w(v_i)$ denotes the weight of part $\mathcal{V}_k$ and $W_{\max}$ denotes the maximum allowed part weight.

B. Aggregate Network Queries in Road Networks

Path computation and route evaluation queries are shown to be highly frequent in intelligent transportation systems [17]. In route evaluation queries, a prespecified path is traversed to compute an objective function (e.g., the total travel time). In path computation queries, a path which satisfies a given objective function (e.g., the shortest path in terms of travel time) is determined. These types of queries are named as aggregate network queries as they depend on the evaluation of a number of nodes at a time.

There are two network operations specific to aggregate queries: Get-a-Successor *GaS*(t_i, t_j) operation retrieves the network element t_j among the successors of t_i and Get-Successors *GSs*(t_i) operation retrieves all successor elements of t_i. *GaS* operations are used in route evaluation queries, where a *Find* operation is followed by a sequence of *GaS* operations. *GSs* operations are used in path computation queries, where a sequence of *Find* and *GSs* operation pairs is performed.

Fig. 1 illustrates a sample network with 8 junctions and 15 links, where squares represent the junctions and directed edges represent the links. In the figure, the access frequencies of *GaS* and *GSs* operations are respectively given on the directed edges and inside the squares. These numbers indicate the number of operations performed on these elements. Typically, distribution of queries over the network elements is not uniform, and individual access frequencies of the network elements are different. Hence, if the previous query logs are available, they can be utilized to estimate the access frequencies of the network elements that will be retrieved by the forthcoming queries.

C. Junction-Based Storage Scheme

A frequently used approach for storing a road network in the secondary storage is to use the adjacency list data structure, where a record is allocated for each junction of the network. Each record r_i stores the data associated with junction t_i and its connectivity information including the predecessor and successor lists. The data associated with junction t_i contains the coordinate of junction t_i and its attributes. A predecessor list denotes the list of incoming links of a junction t_i, whereas a successor list denotes the list of outgoing links of t_i. Each element in the predecessor list stores the coordinates of the source junction t_h of an incoming link ℓ_{hi}. The predecessor lists are used in maintenance operations to update the successor lists. In the successor list, each element stores the coordinates of the destination junction

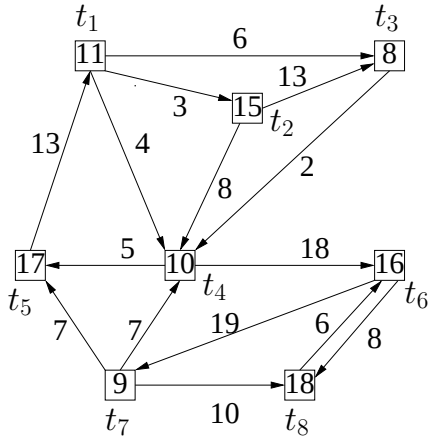

Fig. 1. A sample primal road network.

t_j of an outgoing link ℓ_{ij} as well as the attributes of ℓ_{ij}. The record sizes are not fixed because of the variation in the storage size of the connectivity information of each junction. A storage saving can be achieved if all links of a junction t_i are bidirectional, because all junctions in the predecessor list of t_i also appear in the successor list of t_i and vice versa. Hence, it suffices to store only the successor list of t_i.

D. Data Allocation Problem in Road Networks

The record-to-page allocation problem that we focus on can be defined as follows: Given a road network and data access frequencies extracted from the query logs, allocate a set of data records $\mathcal{R} = \{r_1, r_2, \ldots\}$ to a set of disk pages $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$ such that the expected disk access cost is minimized as much as possible while the number of allocated disk pages is kept reasonable. Typically, allocation of data to disk pages can be modeled as a clustering problem, where the clustering objective is to try to store the records that are likely to be concurrently accessed in the same pages. This way, efficiency in query processing can be achieved since the records relevant to the query can be fetched with fewer disk accesses.

E. Clustering Hypergraph Model for the Junction-Based Storage Scheme

In our earlier study [10], we propose a clustering hypergraph model for the junction-based storage scheme. The proposed model is shown to eliminate the flaws of the clustering graph model [18], [25] and to yield effective results in minimizing the number of disk page accesses. Here, we briefly summarize this model.

For a given road network $(\mathcal{T}, \mathcal{L})$, a clustering hypergraph $\mathcal{H}_{\mathrm{T}} = (\mathcal{V}_{\mathrm{T}}, \mathcal{N}_{\mathrm{T}})$ is created, where a vertex $v_i \in \mathcal{V}_{\mathrm{T}}$ exists for each record $r_i \in \mathcal{R}$ storing the data associated with junction $t_i \in \mathcal{T}$. Each vertex v_i has a weight $w(v_i)$ denoting the size of record r_i. In $\mathcal{H}_{\mathrm{T}}$, the net set $\mathcal{N}_{\mathrm{T}}$ is the union of two disjoint sets of nets, $\mathcal{N}_{\mathrm{T}}^{GaS}$ and $\mathcal{N}_{\mathrm{T}}^{GSs}$.

$\mathcal{N}_{\mathrm{T}}^{GaS}$ encapsulates the disk access costs of *GaS* operations. If junctions t_i and t_j are connected by at least one link, *GaS*(t_i, t_j) operations incur a two-pin net $n_{ij} \in \mathcal{N}_{\mathrm{T}}^{GaS}$ with $\mathrm{Pins}(n_{ij}) = \{v_i, v_j\}$, for $i < j$. The cost $c(n_{ij})$ associated with n_{ij} is

$$c(n_{ij}) = \begin{cases} f(t_i, t_j), & \text{if } \ell_{ij} \in \mathcal{L}, \ell_{ji} \notin \mathcal{L}; \\ f(t_j, t_i), & \text{if } \ell_{ji} \in \mathcal{L}, \ell_{ij} \notin \mathcal{L}; \\ f(t_i, t_j) + f(t_j, t_i), & \text{if } \ell_{ij}, \ell_{ji} \in \mathcal{L} \end{cases} \tag{2}$$

for capturing the costs of *GaS*(t_i, t_j) and *GaS*(t_j, t_i) operations. Here, $f(t_i, t_j)$ denotes the access frequency of the link from junction t_i to junction t_j in *GaS*(t_i, t_j) operations.

$\mathcal{N}_{\mathrm{T}}^{GSs}$ encapsulates the disk access costs of *GSs* operations. For each junction t_i with $d_{\mathrm{out}}(t_i)>0$ successor(s), *GS*(t_i) operations incur a multi-pin net $n_i \in \mathcal{N}_{\mathrm{T}}^{GSs}$ with $d_{\mathrm{out}}(t_i)+1$ pins such that $\mathrm{Pins}(n_i) = \{v_i\} \cup \{v_j : t_j \in \mathrm{Succ}(t_i)\}$, where $\mathrm{Succ}(t_i)$ is the set of successors of t_i. Each net n_i is associated with a cost

$$c(n_i) = f(t_i) \tag{3}$$

for capturing the cost of *GSs*(t_i) operations. Here, $f(t_i)$ denotes the access frequency of junction t_i in *GSs*(t_i) operations.

After modeling the network $(\mathcal{T}, \mathcal{L})$ as a clustering hypergraph $\mathcal{H}_{\mathrm{T}}$, we partition $\mathcal{H}_{\mathrm{T}}$ with the disk page size P being the upper bound on part weights. As shown in [10], this model correctly captures the aggregate disk access cost of *GaS* and *GSs* operations under the single-page buffer assumption. That is, minimizing the cost Cutsize(Π) according to (1) corresponds to minimizing the total number of disk accesses.

III. Link-Based Storage Scheme

A. Definition

In the proposed link-based storage scheme, a record is allocated for each link of the network. Each record r_{ij} stores the data associated with link ℓ_{ij} and its connectivity information. The data associated with a link ℓ_{ij} typically contains the coordinates of junctions t_i and t_j, attributes of the destination junction t_j and attributes of ℓ_{ij}. The connectivity information includes the predecessor and successor lists. The predecessor list of a link ℓ_{ij} includes the set of incoming links of its source junction t_i, whereas the successor list of ℓ_{ij} includes the set of outgoing links of its destination junction t_j. Each element in the predecessor list of a link ℓ_{ij} stores the coordinates of the source junction t_h of an incoming link ℓ_{hi}, whereas each element in the successor list stores the coordinates of the destination junction t_k of an outgoing link ℓ_{jk}.

In this representation, storage savings can be achieved if the network contains bidirectional links where the link attributes are the same for both directions. For example, if $\ell_{ij}, \ell_{ji} \in \mathcal{L}$, the information in records r_{ij} and r_{ji} can be stored as a single record, where the predecessor and successor lists are updated accordingly. Further savings can be achieved if all links of both junctions of a bidirectional link are also bidirectional. In that case, the predecessor and successor lists of both ℓ_{ij} and ℓ_{ji} can be stored only once since the predecessor list of link ℓ_{ij} corresponds to the successor list of link ℓ_{ji} and vice versa.

B. Comparison of Storage Schemes

We should note that the link-based storage scheme of a network corresponds to the junction-based storage scheme of its dual network. In a dual network, a junction t_{hi} exists for each link ℓ_{hi} of the original (primal) network. For each incoming and outgoing link pair ℓ_{hi} and ℓ_{ij} of a junction t_i of the primal network, there exists a link ℓ_{hij} from junction t_{hi} to t_{ij} in the dual network. The dual representation of the sample network given in Fig. 1 is shown in Fig. 2 with 15 junctions and 26 links.

In practice, the storage size of the link attributes is greater than that of the junction attributes, and the number of links is greater than the number of junctions. Depending on these network-specific parameters, one of the two storage schemes may be favorable in terms of the total storage size and/or the average record size. The average record size plays an important role in reducing the number of disk page accesses in query processing as it enables more records to be packed in a page. Below, we provide a detailed comparative analysis of the storage schemes in terms of both the total storage size and average record size.

The total storage sizes S_T and S_L of the junction- and link-based storage schemes can be computed as

$$\begin{aligned} S_{\mathrm{T}} &= \sum_{t \in \mathcal{T}} (C_{\mathrm{id}} + C_{\mathrm{T}} + |Pre(t)|C_{\mathrm{id}} + |Succ(t)|(C_{\mathrm{id}} + C_{\mathrm{L}})) \\ &= |\mathcal{T}|(C_{\mathrm{id}} + C_{\mathrm{T}}) + |\mathcal{L}|(2C_{\mathrm{id}} + C_{\mathrm{L}}) \end{aligned} \tag{4}$$

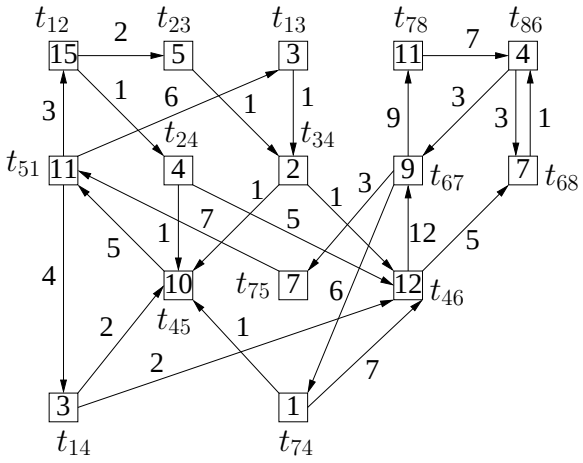

Fig. 2. Dual of the primal network given in Fig. 1.

and

$$\begin{aligned} S_{\mathrm{L}} &= \sum_{\ell \in \mathcal{L}} (2C_{\mathrm{id}} + C_{\mathrm{L}} + C_{\mathrm{T}} + |Pre(\ell)|C_{\mathrm{id}} + |Succ(\ell)|C_{\mathrm{id}}) \\ &= |\mathcal{L}|(2C_{\mathrm{id}} + C_{\mathrm{L}} + C_{\mathrm{T}}) + 2C_{\mathrm{id}} \sum_{t \in \mathcal{T}} (|Pre(t)||Succ(t)|), \end{aligned} \tag{5}$$

where C_{id} denotes the storage size of junction coordinates. C_{T} and C_{L} refer to the fixed storage size of junction and link attributes, respectively. The difference between the total storage sizes of the two schemes is

$$\begin{aligned} S_{\mathrm{L}} - S_{\mathrm{T}} &= 2C_{\mathrm{id}} \sum_{t \in \mathcal{T}} (|Pre(t)||Succ(t)|) + |\mathcal{L}|C_{\mathrm{T}} - |\mathcal{T}|(C_{\mathrm{id}} + C_{\mathrm{T}}) \\ &= C_{\mathrm{T}}(|\mathcal{L}| - |\mathcal{T}|) + 2C_{\mathrm{id}} \sum_{t \in \mathcal{T}} (|Pre(t)||Succ(t)|) - C_{\mathrm{id}}|\mathcal{T}|. \end{aligned} \tag{6}$$

The first term in (6) is always positive since the number of links is greater than the number of junctions in a typical road network. In general, both predecessor and successor lists of most junctions contain more than one link. In (6), the second term is expected to be greater than the last term. Thus, the difference is positive. As a result, in general, the link-based storage scheme requires more disk space than the junction-based storage scheme.

The average record sizes s_{T} and s_{L} of the junction- and link-based storage schemes can be computed as follows under the simplifying assumption that the number of incoming and outgoing links for each junction are both equal to $d_{\mathrm{avg}} = \mathcal{L}/\mathcal{T}$.

$$\begin{aligned} s_{\mathrm{T}} = \frac{S_{\mathrm{T}}}{|\mathcal{T}|} &= C_{\mathrm{id}} + C_{\mathrm{T}} + \frac{|\mathcal{L}|}{|\mathcal{T}|}(2C_{\mathrm{id}} + C_{\mathrm{L}}) \\ &= C_{\mathrm{id}} + C_{\mathrm{T}} + d_{\mathrm{avg}}(2C_{\mathrm{id}} + C_{\mathrm{L}}) \end{aligned} \tag{7}$$

and

$$\begin{aligned} s_{\mathrm{L}} = \frac{S_{\mathrm{L}}}{|\mathcal{L}|} &= 2C_{\mathrm{id}} + C_{\mathrm{L}} + C_{\mathrm{T}} + 2C_{\mathrm{id}} d_{\mathrm{avg}}^2 \frac{|\mathcal{T}|}{|\mathcal{L}|} \\ &= 2C_{\mathrm{id}} + C_{\mathrm{L}} + C_{\mathrm{T}} + 2C_{\mathrm{id}} d_{\mathrm{avg}}. \end{aligned} \tag{8}$$

The difference between the average record sizes of the two schemes is

$$s_{\mathrm{T}} - s_{\mathrm{L}} = (d_{\mathrm{avg}} - 1)C_{\mathrm{L}} - C_{\mathrm{id}}. \tag{9}$$

In a typical road network, since this difference is almost always positive, average record size in the link-based storage scheme is less than that of the junction-based storage scheme. As seen by this comparative analysis, although the link-based storage scheme requires more disk space, its average record size is likely to be smaller. Hence, the link-based storage scheme can be expected to perform better than the junction-based storage scheme in terms of disk access cost.

In bidirectional networks, the storage savings described in Sections II-C and III-A are expected to increase the efficiency of both storage schemes. However, the link-based storage scheme is expected to benefit more from the storage savings compared to the junction-based storage scheme, because the link-based storage scheme becomes more amenable to clustering. We illustrate this with an example. Consider a junction t_j with d links all of which are bidirectional. In the junction-based storage scheme, junction t_j will have d successors. We should cluster record r_j storing t_j together with all the records storing the d successor junctions to the same page to avoid the page access cost for the *GSs*(t_j) operation. That is, these $d+1$ records need to be clustered in the same page. On the other hand, in the link-based storage scheme scheme, each link incident to junction t_j has $d-1$ successors excluding itself. Since r_{ij} stores both ℓ_{ij} and ℓ_{ji}, we should cluster record r_{ij} together with $d-1$ records storing the links incident to t_j other than ℓ_{ji} in the same page to avoid the page access cost for the *GSs*(ℓ_{ij}) operation. This holds for all records storing the links incident to junction t_j. Hence, it is sufficient to cluster these d records in the same page to avoid the page access cost for the *GSs* operations invoked from the links incident to junction t_j. By definition, for a given query distribution, the sum of the frequencies of the *GSs* operations to be invoked from the links incident to junction t_j in the link-based storage scheme is equal to the frequency of the *GSs* operations to be invoked from t_j in the junction-based storage scheme. This explains why the link-based storage scheme will be more amenable to clustering than the junction-based storage scheme even when the average record sizes are equal in the two storage schemes.

IV. Clustering Hypergraph Model for the Link-Based Storage Scheme

In this section, we present our clustering hypergraph model for the general case of directed networks, where an individual record is stored for each directed link. This model can easily be extended to the bidirectional case, where a single record is stored for each bidirectional link.

A. Hypergraph Construction

A clustering hypergraph $\mathcal{H}_{\mathrm{L}} = (\mathcal{V}_{\mathrm{L}}, \mathcal{N}_{\mathrm{L}})$ is created to model the network $(\mathcal{T}, \mathcal{L})$. In $\mathcal{H}_{\mathrm{L}}$, a vertex $v_{ij} \in \mathcal{V}_{\mathrm{L}}$ exists for each record $r_{ij} \in \mathcal{R}$ storing the data associated with link $\ell_{ij} \in \mathcal{L}$. The size of a record r_{ij} is assigned as the weight $w(v_{ij})$ of vertex v_{ij}. The net set $\mathcal{N}_{\mathrm{L}}$ is the union of two disjoint sets of nets, $\mathcal{N}_{\mathrm{L}}^{GaS}$ and $\mathcal{N}_{\mathrm{L}}^{GSs}$, which respectively encapsulate the disk access costs of *GaS* and *GSs* operations, i.e., $\mathcal{N}_{\mathrm{L}} = \mathcal{N}_{\mathrm{L}}^{GaS} \cup \mathcal{N}_{\mathrm{L}}^{GSs}$.

In $\mathcal{N}_{\mathrm{L}}^{GaS}$, we employ two-pin nets to represent the cost of *GaS* operations. For each incoming and outgoing link pair ℓ_{hi} and ℓ_{ij} of each junction t_i, *GaS*(ℓ_{hi}, ℓ_{ij}) operations incur a two-pin net n_{hij} with $\mathrm{Pins}(n_{hij}) = \{v_{hi}, v_{ij}\}$. If the source junction of the incoming link is the same as the destination junction of the outgoing link (i.e., $h = j$), the two two-pin nets incurred by the *GaS*(ℓ_{hi}, ℓ_{ij}) and *GaS*(ℓ_{ij}, ℓ_{hi}) operations can be coalesced into a single two-pin net with appropriate cost adjustment. Thus, the cost $c(n_{hij})$ associated with net n_{hij} can be written as

$$c(n_{hij}) = \begin{cases} f(\ell_{hi}, \ell_{ij}), & \text{if } \ell_{hi}, \ell_{ij} \in \mathcal{L} \wedge h \neq j; \\ f(\ell_{hi}, \ell_{ij}) + f(\ell_{ij}, \ell_{hi}), & \text{if } \ell_{hi}, \ell_{ij} \in \mathcal{L} \wedge h = j. \end{cases} \tag{10}$$

Here, $f(\ell_{hi}, \ell_{ij})$ denotes the total access frequency of path $\langle \ell_{hi}, \ell_{ij} \rangle$ in *GaS*(ℓ_{hi}, ℓ_{ij}) operations. Fig. 3(a) shows the two-pin net construction for a pair of neighbor links ℓ_{12} and ℓ_{23}, and Fig. 3(b) shows the two-pin net construction for the cyclic paths $\langle \ell_{12}, \ell_{21} \rangle$ and $\langle \ell_{21}, \ell_{12} \rangle$.

In $\mathcal{N}_{\mathrm{L}}^{GSs}$, we employ multi-pin nets to represent the cost of *GSs* operations. For each link ℓ_{hi} with a destination junction t_i having $d_{\mathrm{out}}(t_i) > 0$ successor(s), *GSs*(t_i) operations incur a $(d_{\mathrm{out}}(t_i)+1)$-pin net

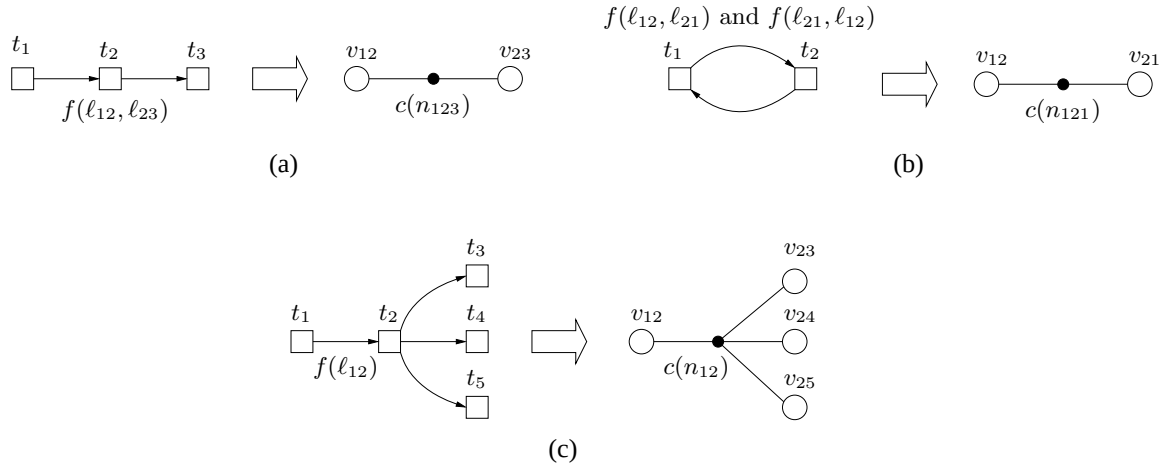

Fig. 3. The clustering hypergraph construction (a) Two-pin net n_{123} for the *GaS*(ℓ_{12}, ℓ_{23}) operations and (b) Coalescence of two two-pin nets incurred by *GaS*(ℓ_{12}, ℓ_{21}) and *GaS*(ℓ_{21}, ℓ_{12}) into net n_{121} (c) Multi-pin net n_{12} for the *GSs*(ℓ_{12}) operations.

n_{hi}, which connects vertex v_{hi} and the vertices corresponding to the records of the links that are in the successor list of ℓ_{hi}. That is,

$$\text{Pins}(n_{hi}) = \{v_{hi}\} \cup \{v_{ij} : t_j \in \text{Succ}(t_i)\}. \tag{11}$$

Each net n_{hi} is associated with a cost

$$c(n_{hi}) = f(\ell_{hi}) \tag{12}$$

for capturing the cost of *GSs*(ℓ_{hi}) operations. Here, $f(\ell_{hi})$ denotes the total access frequency of link ℓ_{hi} in *GSs*(ℓ_{hi}) operations. Fig. 3(c) displays the multi-pin net construction for link ℓ_{12}, which has the successor list $\{\ell_{23}, \ell_{24}, \ell_{25}\}$.

B. Clustering Hypergraph Model

After $\mathcal{H}_{\text{L}} = (\mathcal{V}_{\text{L}}, \mathcal{N}_{\text{L}})$ is constructed, it is partitioned into a number of parts $\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots\}$, where each part $\mathcal{V}_k \in \Pi$ corresponds to the subset of records to be assigned to disk page $\mathcal{P}_k \in \mathcal{P}$. The partitioning constraint is to enforce the page size as the upper bound on the weight of the vertex parts so that the disk page size is not exceeded in record allocation. The partitioning objective is to minimize the cutsize according to the connectivity-1 metric as defined in Section II-A. Under the single-page buffer assumption, the connectivity-1 cost incurred to the cutsize by the two-pin cut nets in $\mathcal{N}_{\text{L}}^{GaS}$ and multi-pin cut nets in $\mathcal{N}_{\text{L}}^{GSs}$ exactly corresponds to the disk access cost incurred by the *GaS* operations in the route evaluation queries and *GSs* operations in the path computation queries, respectively. Thus, in our model, minimizing Cutsize(Π) given in (13) exactly minimizes the total number of disk accesses. In the following two paragraphs, we show the correctness of our model for the two-pin and multi-pin net cases.

$$\begin{aligned} \text{Cutsize}(\Pi) &= \sum_{n_i \in \mathcal{N}_{\text{L}}^{GaS}} c(n_i)(\lambda(n_i) - 1) + \sum_{n_i \in \mathcal{N}_{\text{L}}^{GSs}} c(n_i)(\lambda(n_i) - 1) \\ &= \sum_{n_i \in \mathcal{N}_{\text{L}}} c(n_i)(\lambda(n_i) - 1). \end{aligned} \tag{13}$$

Consider a partition Π and a two-pin net $n_{hij} \in \mathcal{N}_{\text{L}}^{GaS}$ with $\text{Pins}(n_{hij}) = \{v_{hi}, v_{ij}\}$. If n_{hij} is internal to a part $\mathcal{V}_k$, then records r_{hi} and r_{ij} both reside in page $\mathcal{P}_k$. Since both r_{hi} and r_{ij} can be found in the memory when $\mathcal{P}_k$ is in the page buffer, neither *GaS*(ℓ_{hi}, ℓ_{ij}) nor *GaS*(ℓ_{ij}, ℓ_{hi}) operations incur any disk

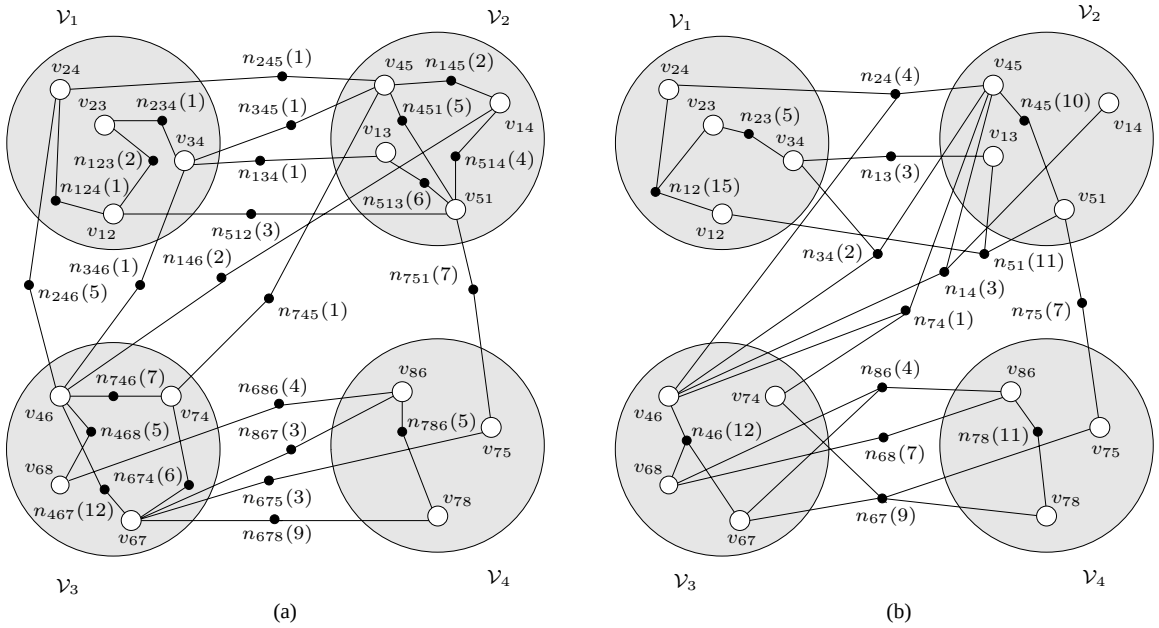

Fig. 4. The clustering hypergraph $\mathcal{H}_{\mathrm{L}}$ for the network given in Fig. 1 and a 4-way vertex partition separately shown on net-induced subhypergraphs (a) $(\mathcal{V}_{\mathrm{L}}, \mathcal{N}_{\mathrm{L}}^{GaS})$ and (b) $(\mathcal{V}_{\mathrm{L}}, \mathcal{N}_{\mathrm{L}}^{GSs})$ respectively modeling the disk access cost of *GaS* and *GSs* operations.

access. Note that *GaS*(ℓ_{ij}, ℓ_{hi}) operations are possible only if $h=j$. If n_{hij} is a cut net with connectivity set $\Lambda(n_{hij})=\{\mathcal{V}_k, \mathcal{V}_m\}$, r_{hi} and r_{ij} reside in separate pages $\mathcal{P}_k$ and $\mathcal{P}_m$. Without loss of generality, assume that $r_{hi} \in \mathcal{P}_k$ and $r_{ij} \in \mathcal{P}_m$. In this case, *GaS*(ℓ_{hi}, ℓ_{ij}) operations incur $f(\ell_{hi}, \ell_{ij})$ disk accesses in order to replace the current page $\mathcal{P}_k$ in the buffer with $\mathcal{P}_m$ in the disk. In a similar manner, *GaS*(ℓ_{ij}, ℓ_{hi}) operations incur $f(\ell_{ij}, \ell_{hi})$ disk accesses in order to replace the current page $\mathcal{P}_m$ in the buffer with $\mathcal{P}_k$ in the disk. Hence, cut net n_{hij} incurs a cost of $c(n_{hij})$ to the cutsize since $\lambda(n_{hij})-1=1$.

Now, consider the same partition Π and a multi-pin net $n_{ij} \in \mathcal{N}_{\mathrm{T}}^{GSs}$. If n_{ij} is internal to a part $\mathcal{V}_k$, then record r_{ij} and all records storing the links in the successor list of ℓ_{ij} reside in page $\mathcal{P}_k$. Consequently, *GSs*(ℓ_{ij}) operations do not incur any disk access since page $\mathcal{P}_k$ is already in the page buffer. If n_{ij} is a cut net with connectivity set $\Lambda(n_{ij})$, record r_{ij} and the records storing the links in the successor list of ℓ_{ij} are distributed across the pages corresponding to the vertex parts that belong to $\Lambda(n_{ij})$. Without loss of generality, assume that r_{ij} resides in page $\mathcal{P}_k$, where $\mathcal{V}_k$ must be in $\Lambda(n_{ij})$. In this case, each *GSs*(ℓ_{ij}) operation incurs $\lambda(n_{ij})-1$ page accesses in order to retrieve the records storing the links in the successor list of ℓ_{ij} by fetching the pages corresponding to the vertex parts in $\Lambda(n_{ij}) - \{\mathcal{V}_k\}$. Hence, cut net n_{ij} incurs a cost of $c(n_{ij})(\lambda(n_{ij})-1)$ to the cutsize.

Fig. 4 shows the clustering hypergraph $\mathcal{H}_{\mathrm{L}}$ for the network given in Fig. 1 in two parts, which separately show the net sets $\mathcal{N}_{\mathrm{L}}^{GaS}$ and $\mathcal{N}_{\mathrm{L}}^{GSs}$ with the associated costs of *GaS* and *GSs* operations shown in parentheses. In Fig. 4(a), consider two-pin cut net n_{246} with $\mathrm{Pins}(n_{246})=\{v_{24}, v_{46}\}$ and $\Lambda(n_{246})=\{\mathcal{V}_1, \mathcal{V}_3\}$. Since v_{24} is in vertex part $\mathcal{V}_1$, page $\mathcal{P}_1$ must be the single page in the buffer when *GSs*(ℓ_{24}) operations are invoked. Since v_{46} is in part $\mathcal{V}_2$, $\lambda(n_{246})-1=2-1=1$ disk access is required to retrieve record r_{46} into the buffer. Similarly, in Fig. 4(b), consider multi-pin cut net n_{24} with $\mathrm{Pins}(n_{24})=\{v_{24}, v_{45}, v_{46}\}$ and $\Lambda(n_{24})=\{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3\}$. Since v_{24} is in vertex part $\mathcal{V}_1$, page $\mathcal{P}_1$ must be the single page in the buffer when *GSs*(ℓ_{24}) operations are invoked. Since v_{45} and v_{46} are respectively in parts $\mathcal{V}_2$ and $\mathcal{V}_3$, each of the four *GSs*(ℓ_{24}) operations will incur $\lambda(n_{24})-1=3-1=2$ disk accesses for pages $\mathcal{P}_2$ and $\mathcal{P}_3$ to bring them into the buffer for processing records r_{45} and r_{46}. Note that internal nets do not incur any cost for neither *GaS* nor *GSs* operations since they have a connectivity of 1. The total cost of *GaS* operations, due to the cut nets $\{n_{134}, n_{146}, n_{245}, n_{246}, n_{345}, n_{346}, n_{512}, n_{675}, n_{678}, n_{686}, n_{745}, n_{751}, n_{867}\}$, is $(1+2+1+5+1+1+3+3+9+4+1+7+3)\times(2-1)=41$ and the total cost of *GSs* operations, due to the cut nets $\{n_{13}, n_{14}, n_{24}, n_{34}, n_{51}, n_{67}, n_{68}, n_{74}, n_{75}, n_{86}\}$, is $3\times(2-1)+3\times(2-1)+4\times(3-1)+2\times(3-1)+$

$11\times(2-1)+9\times(2-1)+7\times(2-1)+1\times(2-1)+7\times(2-1)+4\times(2-1)=57$.

In this work, similar to our earlier proposal on clustering in the junction-based storage scheme [10], we use recursive bipartitioning schemes to partition $\mathcal{H}_{\mathrm{L}}$ into parts. The use of recursive bipartitioning schemes is due to the fact that the number of parts is not known in advance. The reader is referred to [10] for the details of the two recursive bipartitioning schemes used for this purpose.

C. Comparison of Clustering Hypergraph Models

We should note that the clustering hypergraph model for the link-based storage of the primal network is equivalent to the clustering hypergraph model for the junction-based storage of the dual network. This equivalence stems from the fact that the link-based storage scheme is the dual of the junction-based storage scheme as mentioned earlier in Section III-B.

The sizes of the constructed hypergraphs in our clustering models depend on the topological properties of the network. These sizes play an important role in computational and space requirements of the partitioning process. In the clustering hypergraph $\mathcal{H}_{\mathrm{T}}$ for the junction-based storage scheme, the number $|\mathcal{N}_{\mathrm{T}}^{GaS}|$ of two-pin nets varies between $\lceil|\mathcal{L}|/2\rceil$ and $|\mathcal{L}|$. The number $|\mathcal{N}_{\mathrm{T}}^{GSs}|$ of multi-pin nets is equal to $|\mathcal{T}|-\alpha$, where $\alpha=|\{t_i : d_{\mathrm{out}}(t_i)=0\}|$ is the number of dead ends. The number of pins introduced by multi-pin nets is $|\mathcal{L}|+|\mathcal{T}|-\alpha$. Hence, we have

$$\begin{aligned} |\mathcal{V}_{\mathrm{T}}| &= |\mathcal{T}|, \\ \lceil|\mathcal{L}|/2\rceil+|\mathcal{T}|-\alpha \le |\mathcal{N}_{\mathrm{T}}| &\le |\mathcal{L}|+|\mathcal{T}|-\alpha, \\ 2\lceil 1.5\,|\mathcal{L}|\rceil+|\mathcal{T}|-\alpha \le |\mathcal{H}_{\mathrm{T}}| &\le 3|\mathcal{L}|+|\mathcal{T}|-\alpha. \end{aligned} \tag{14}$$

In the clustering hypergraph $\mathcal{H}_{\mathrm{L}}$ for the link-based storage scheme, the number $|\mathcal{N}_{\mathrm{L}}^{GaS}|$ of two-pin nets is $\sum_{t_i\in\mathcal{T}}(d_{\mathrm{in}}(t_i)\times d_{\mathrm{out}}(t_i))-\beta$, where $d_{\mathrm{in}}(t_i)$ denotes the number of predecessors of t_i and $\beta=|\{\ell_{ij} : \ell_{ij}\in\mathcal{L}\wedge\ell_{ji}\in\mathcal{L}\}|$ is the number of bidirectional links. The number $|\mathcal{N}_{\mathrm{L}}^{GSs}|$ of multi-pin nets is equal to $|\mathcal{L}|-\sum_{t_i\in\mathcal{T},d_{\mathrm{out}}(t_i)=0} d_{\mathrm{in}}(t_i)$. The number of pins introduced by multi-pin nets is $\sum_{t_i\in\mathcal{T},d_{\mathrm{out}}(t_i)>0} d_{\mathrm{in}}(t_i)\times(d_{\mathrm{out}}(t_i)+1)$. Hence, we have

$$\begin{aligned} |\mathcal{V}_{\mathrm{L}}| &= |\mathcal{L}|, \\ |\mathcal{N}_{\mathrm{L}}| &= \sum_{t_i\in\mathcal{T}}(d_{\mathrm{in}}(t_i)\times d_{\mathrm{out}}(t_i))-\beta+|\mathcal{L}|-\sum_{t_i\in\mathcal{T},d_{\mathrm{out}}(t_i)=0} d_{\mathrm{in}}(t_i), \\ |\mathcal{H}_{\mathrm{L}}| &= 3\sum_{t_i\in\mathcal{T}}(d_{\mathrm{in}}(t_i)\times d_{\mathrm{out}}(t_i))+\sum_{t_i\in\mathcal{T},d_{\mathrm{out}}(t_i)>0} d_{\mathrm{in}}(t_i)-2\beta. \end{aligned} \tag{15}$$

In this work, we claim that the clustering hypergraph model provides more flexibility in partitioning for the link-based storage scheme compared to the junction-based storage scheme. We illustrate this by the following example. Fig. 5(a) shows a sample sub-road network $(\mathcal{T},\mathcal{L})$ with a junction t_3 having two incoming and three outgoing links. Figs. 5(b) and 5(c) show the net-induced subhypergraphs $(\mathcal{V}_{\mathrm{T}},\mathcal{N}_{\mathrm{T}}^{GSs})$ and $(\mathcal{V}_{\mathrm{L}},\mathcal{N}_{\mathrm{L}}^{GSs})$ corresponding to the sub-road network given in Fig. 5(a) for the junction- and link-based storage schemes, respectively. Ten *GSs* operations are assumed to be performed on junction t_3, five *GSs* operations for each incoming link of t_3. As seen in the figure, junction t_3 induces only one net n_3 in $\mathcal{H}_T$, whereas the two incoming links ℓ_{13} and ℓ_{23} of t_3 induce nets n_{13} and n_{23} in $\mathcal{H}_L$.

Figs. 5(b) and 5(c) also show 2-way partitions for $\mathcal{H}_T$ and $\mathcal{H}_L$. In this example, if there were no part size constraints, moving vertex v_3 from $\mathcal{V}_1$ to $\mathcal{V}_2$ would remove net n_3 from the cut, thus reducing the cutsize by 10. However, this move may not be feasible due to the maximum part size constraint on $\mathcal{V}_2$. Since the record sizes in the link-based storage scheme are less than those in the junction-based storage scheme as shown in Section III-B, either v_{13} or v_{23} can move to $\mathcal{V}_2$ without violating the maximum part size constraint, respectively removing n_{13} or n_{23} from the cut with a saving of 5 on the cutsize. In general,

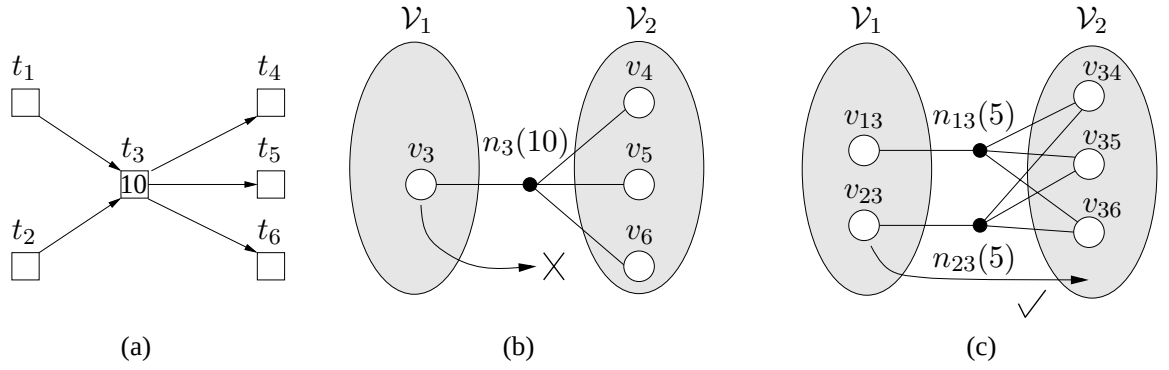

Fig. 5. (a) A sub-road network with *GSs*(t_3) (b) $\mathcal{H}_{\mathrm{T}}$: a four-pin net n_3 for the *GSs*(t_3) operations with $f(t_3)=10$ (c) $\mathcal{H}_{\mathrm{L}}$: two four-pin nets n_{13} for the *GSs*(ℓ_{13}) operations with $f(\ell_{13})=5$ and n_{23} for the *GSs*(ℓ_{23}) operations with $f(\ell_{13})=5$.

the partitioning of the clustering hypergraph for the link-based storage scheme has a better solution space as the flexibility of moving vertices between parts increases.

In bidirectional networks, the storage saving in the link-based scheme results in higher improvements in query processing performance compared to the junction-based scheme. We provide Fig. 6 to validate this claim. Fig. 6(a) shows a sample sub-road network $(\mathcal{T},\mathcal{L})$ with a junction t_1 having four bidirectional incoming/outgoing links. Figs. 6(b) and 6(c) show the net-induced subhypergraphs $(\mathcal{V}_{\mathrm{T}},\mathcal{N}_{\mathrm{T}}^{GSs})$ and $(\mathcal{V}_{\mathrm{L}},\mathcal{N}_{\mathrm{L}}^{GSs})$ corresponding to the sub-road network for the junction- and link-based storage schemes, respectively. Note that the sum of the number of *GSs* operations performed on the incoming links of junction t_1 in the link-based storage scheme is equal to the number of *GSs* operations performed on junction t_1. That is, $f(\ell_{21})+f(\ell_{31})+f(\ell_{41})+f(\ell_{51})=f(t_1)$.

As seen in Fig. 6(b), in $\mathcal{H}_{\mathrm{T}}$, for the *GSs*(t_1) operation, there is a five-pin net with $\mathrm{Pins}(n_1)=\{v_1,v_2,v_3,v_4,v_5\}$ and $c(n_1)=f(t_1)$. In the construction of the clustering hypergraph for the link-based storage scheme, two directional links between the same junctions (i.e., ℓ_{ij} and ℓ_{ji}) are represented with a bidirectional link ℓ_{ij}, where $i<j$. Hence, a vertex v_{ij} exists for each record r_{ij} storing link ℓ_{ij}. As seen in Fig. 6(c), $\mathcal{H}_{\mathrm{L}}$ has four four-pin nets n_{12},n_{13},n_{14}, and n_{15} to capture the costs of the *GSs*(ℓ_{21}), *GSs*(ℓ_{31}), *GSs*(ℓ_{41}), and *GSs*(ℓ_{51}) operations, respectively. Note that these four four-pin nets connect the same set of pins, i.e., $\mathrm{Pins}(n_{12})=\mathrm{Pins}(n_{13})=\mathrm{Pins}(n_{14})=\mathrm{Pins}(n_{15})=\{v_1,v_2,v_3,v_4,v_5\}$. Such nets, which connect exactly the same set of pins, are called identical nets. Identical nets can be coalesced into a single representative net. The representative net's cost is set to the total cost of all constituting nets. Here, n_{12},n_{13},n_{14}, and n_{15} can be coalesced into a representative net n'_1 with $\mathrm{Pins}(n'_1)=\{v_{12},v_{13},v_{14},v_{15}\}$ and $c(n'_1)=c(n_{12})+c(n_{13})+c(n_{14})+c(n_{15})$ as shown in Fig. 6(d). Comparison of Figs. 6(b) and 6(d) shows that, for *GSs* operations, the clustering hypergraphs for the two storage schemes have the same set of nets with equal costs. However, the size of each net in $\mathcal{H}_{\mathrm{L}}$ is one less than the size of the respective net in $\mathcal{H}_{\mathrm{T}}$. This finding conforms with the fact that, in query processing, each *GSs* operation in the link-based storage scheme accesses one record less compared to the junction-based storage scheme. Thus, the partitioning of $\mathcal{H}_{\mathrm{L}}$ is expected to lead to smaller cutsizes compared to that of $\mathcal{H}_{\mathrm{T}}$ because of smaller net sizes in the link-based storage scheme.

In bidirectional networks, the sizes of the clustering hypergraphs for the two storage schemes become

$$\begin{aligned} |\mathcal{V}_{\mathrm{T}}| &= |\mathcal{T}|, \\ |\mathcal{N}_{\mathrm{T}}| &= |\mathcal{L}|/2 + |\mathcal{T}|, \\ |\mathcal{H}_{\mathrm{T}}| &= 2|\mathcal{L}| + |\mathcal{T}|, \end{aligned} \tag{16}$$

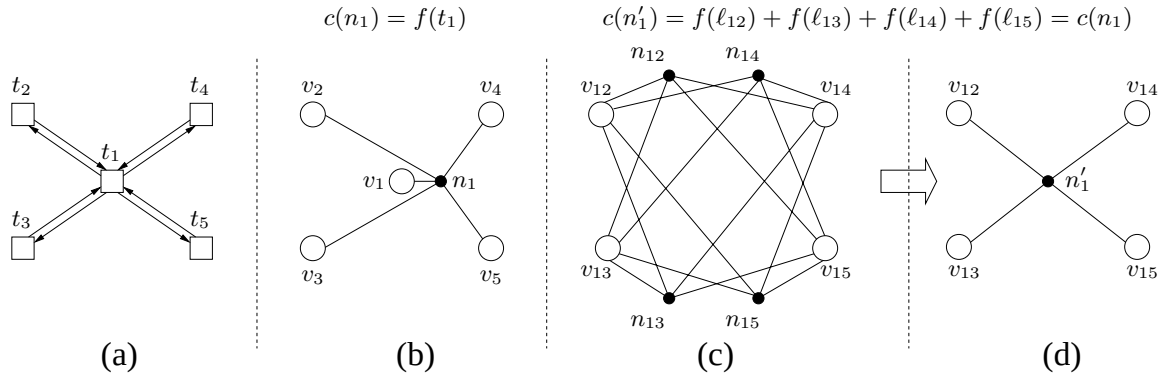

Fig. 6. (a) A bidirectional sub-road network with *GSs*(t_1) (b) $\mathcal{H}_{\mathrm{T}}$: a five-pin net n_1 for the *GSs*(t_1) operations with $c(n_1) = f(t_1)$ (c) $\mathcal{H}_{\mathrm{L}}$: four identical four-pin nets n_{12}, n_{13}, n_{14}, and n_{15} for *GSs*(ℓ_{12}), *GSs*(ℓ_{13}), *GSs*(ℓ_{14}), and *GSs*(ℓ_{15}), respectively (d) $\mathcal{H}_{\mathrm{L}}$: identical nets n_{12}, n_{13}, n_{14}, and n_{15} coalesced into net n_1' with cost $c(n_1') = c(n_1)$.

TABLE I
PROPERTIES OF ROAD NETWORK DATASETS

		Road network		
Tag	Dataset	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{L}\rvert$	d_{avg}
D1	California HPN	10141	28370	2.80
D2	SanJoaquin	17444	45974	2.64
D3	Minnesota7	34222	92206	2.69
D4	Sanfrancisco	166558	426742	2.56

and

$$\begin{aligned} |\mathcal{V}_{\mathrm{L}}| &= |\mathcal{L}|/2, \\ |\mathcal{N}_{\mathrm{L}}| &= \sum_{t_i \in \mathcal{T}} d^2(t_i) - |\mathcal{L}| + |\mathcal{T}| - \tau, \\ |\mathcal{H}_{\mathrm{L}}| &= 2 \sum_{t_i \in \mathcal{T}} d^2(t_i) - |L| - \tau, \end{aligned} \tag{17}$$

where $d(t_i) = d_{\mathrm{in}}(t_i) = d_{\mathrm{out}}(t_i)$ and $\tau = |\{t_i : d(t_i) = 1\}|$.

V. EXPERIMENTAL RESULTS

A. Experimental Setup

In order to show the validity of the proposed link-based storage scheme and the clustering model, we have conducted a wide range of experiments on four real-life road network datasets collected from U.S. Tiger/Line [19] (Minnesota7 including 7 counties Anoka, Carver, Dakota, Hennepin, Ramsey, Scott, Washington; Sanfrancisco), U.S. Department of Transportation [22] (California Highway Planning Network), and Brinkhoff's network data generator [5] (SanJoaquin). We perform a preprocessing to eliminate the self-loops and multi-links. The properties of the preprocessed datasets are given in Table I. In the table, d_{avg} refers to the average number of links per junction.

It is important to note that all links in our datasets are bidirectional. This fact enables the use of the storage savings mentioned in Sections II-C and III-A. In the junction-based storage scheme, we store only the successor list of each junction. In the link-based storage scheme, we combine the records storing the two directional links between two junctions into a single record and hence halve the number of records.

In the experiments, 4 bytes are reserved for the coordinates of a junction (i.e., $C_{\mathrm{id}} = 4$) and no space is reserved for junction attributes (i.e., $C_{\mathrm{T}} = 0$). We used three different sizes of 16, 28, and 40 bytes for the link attributes (i.e., $C_{\mathrm{L}} = 16$, $C_{\mathrm{L}} = 28$, and $C_{\mathrm{L}} = 40$) in both storage schemes. This way, we are able to evaluate the effect of the average record size and total storage size on the relative performance of the

TABLE II
STORAGE REQUIREMENTS OF JUNCTION AND LINK-BASED STORAGE SCHEMES IN BYTES

	Junction-based storage scheme						Link-based storage scheme					
	$C_{\mathrm{L}}=16$		$C_{\mathrm{L}}=28$		$C_{\mathrm{L}}=40$		$C_{\mathrm{L}}=16$		$C_{\mathrm{L}}=28$		$C_{\mathrm{L}}=40$	
	S_{T}	s_{T}	S_{T}	s_{T}	S_{T}	s_{T}	S_{L}	s_{L}	S_{L}	s_{L}	S_{L}	s_{L}
D1	607964	60.0	948404	93.5	1288844	127.1	813624	57.4	983844	69.4	1154064	81.4
D2	989256	56.7	1540944	88.3	2092632	120.0	1298856	56.5	1574700	68.5	1850544	80.5
D3	1981008	57.9	3087480	90.2	4193952	122.6	2650184	57.5	3203420	69.5	3756656	81.5
D4	9201072	55.2	14321976	86.0	19442880	116.7	11850952	55.5	14411404	67.5	16971856	79.5
Averages normalized w.r.t. storage sizes of the junction-based scheme												
	1.00	1.00	1.00	1.00	1.00	1.00	1.29	0.99	1.01	0.77	0.87	0.67

two storage schemes. Table II displays the total storage sizes (S_{T} and S_{L}) and the average record sizes (s_{T} and s_{L}) for the junction- and link-based storage schemes for each dataset and link attribute size pair.

As seen in Table II, for $C_{\mathrm{L}}=16$, the average record sizes are almost equal in the two storage schemes, whereas the link-based scheme requires 29% more total storage than the junction-based scheme, on the average. For $C_{\mathrm{L}}=28$, the total storage sizes are almost equal in the two storage schemes, whereas the average record size of the link-based scheme is 23% less than that of the junction-based scheme, on the average. For $C_{\mathrm{L}}=40$, both the total storage size and the average record size of the link-based scheme are less than those of the junction-based scheme (on the average 13% and 33%, respectively). Although the link-based scheme requires more storage than the junction-based scheme in general, the link-based scheme becomes more favorable than the the junction-based scheme for $C_{\mathrm{L}}=40$. This is mainly due to the fact that the proposed way of handling bidirectional links enables more storage saving in the link-based scheme compared to that in the junction-based scheme.

The clustering hypergraphs for the two storage schemes are constructed as described in Sections II-E and IV-A. The vertex weights are set to be equal to the size of the respective records. We generated a synthetic query log for each dataset in order to be able to obtain a cost distribution over the nets of the constructed hypergraphs. For this purpose, a set of source and destination junction pairs, which have a predetermined shortest path length, is generated by slightly modifying the network-based node selection option of Brinkhoff's Network Generator for Moving Objects framework [5]. For the route evaluation queries, the shortest path between the source and destination junctions is evaluated. For the path computation queries Dijkstra's algorithm is executed over the network between these junctions.

In order to have almost all elements in the networks accessed by at least one aggregate network operation, we adaptively determined a separate query count and a path length for each dataset. Here, we should note that the total net costs in the clustering hypergraphs generated for the two storage schemes are exactly equal for a given query log on the same dataset. This enables a fair comparison between the clustering hypergraph models for the two storage schemes.

According to the path lengths in the queries, we formed three sets of queries: Q_{short}, Q_{medium}, and Q_{long}. We selected the path lengths and the number of queries in each query set as follows: For Q_{short}, Q_{medium}, and Q_{long}, the path length is respectively set to the 1/18, 1/6, and 1/2 of the diameter of the road network. In order to span all the junctions in the network and hence to create a hypergraph large enough to represent the network, the number of queries in each dataset is picked linearly proportional to the number of junctions. For Q_{short}, Q_{medium}, and Q_{long}, the number of queries is respectively set to the 5/10, 3/10, and 1/10 of the number of junctions in the network. Table III displays the path length and the number of queries used for each dataset and query set pair. Table III also displays the number of *GaS* and *GSs* operations respectively invoked by the route evaluation and path computation queries for each dataset and query set pair.

Table IV displays the properties of the clustering hypergraphs used in the experiments for the junction- and link-based storage schemes. In this table, $|n|_{\mathrm{avg}}=|\mathcal{H}|/|\mathcal{N}|$ denotes the average net size of a hypergraph. Since the aggregate network operations incurred by the generated queries may not traverse all network elements, the number of nets for each hypergraph is less than the number of all possible nets that can be induced. As mentioned in Section IV-C, bidirectional links lead to identical nets in both storage

TABLE III
PROPERTIES OF QUERIES PERFORMED ON THE ROAD NETWORKS

	Q_{short}				Q_{medium}				Q_{long}			
	path length	number of queries	*GaS*()	*GSs*()	path length	number of queries	*GaS*()	*GSs*()	path length	number of queries	*GaS*()	*GSs*()
D1	8	5071	30420	498478	25	3042	69943	3108062	75	1014	74022	3977814
D2	8	8722	52230	823121	25	5233	119572	4830266	76	1744	127948	9033815
D3	26	17111	405910	14583559	78	10267	766892	61064163	233	3422	774053	70111055
D4	27	83279	2080352	129398112	81	49967	3944006	604478026	242	16656	3995328	959588281

TABLE IV
PROPERTIES OF THE CLUSTERING HYPERGRAPHS FOR THE JUNCTION- AND LINK-BASED STORAGE SCHEMES

		Q_{short}			Q_{medium}			Q_{long}		
	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert n\rvert_{\text{avg}}$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert n\rvert_{\text{avg}}$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert n\rvert_{\text{avg}}$
Junction-based storage scheme										
D1	10141	19344	56913	2.9	15691	49607	3.2	14576	47376	3.3
D2	17444	30033	88575	2.9	25926	80359	3.1	23987	76449	3.2
D3	34222	50970	159836	3.1	49439	156747	3.2	45128	148033	3.3
D4	166558	250116	760252	3.0	243853	747713	3.1	225476	710905	3.2
Link-based storage scheme										
D1	14185	18400	45302	2.5	14553	37603	2.6	13092	34680	2.6
D2	22987	28768	72090	2.5	22991	60526	2.6	20423	55367	2.7
D3	46103	47080	125054	2.7	44659	120200	2.7	38581	107968	2.8
D4	213371	222231	576712	2.6	211869	555947	2.6	186466	504947	2.7

schemes.These nets are detected and eliminated by a preprocessing step. Table IV displays the values after this identical net elimination step.

As seen in Table IV, $\mathcal{H}_{\text{L}}$ contains considerably more (25.1% on the average) vertices than $\mathcal{H}_{\text{T}}$. This is expected since the number of vertices corresponds to the number of records in a storage scheme and the number of records is equal to the number of junctions in the junction-based storage scheme, whereas it is equal to half of the number of links in the link-based storage scheme. In terms of the number of nets, $\mathcal{H}_{\text{L}}$ contains fewer (10.5% on the average) nets than $\mathcal{H}_{\text{T}}$. This is mainly due to the junctions with degree one, which do not incur multi-pin nets in $\mathcal{H}_{\text{L}}$. In Table IV, the average net size in $\mathcal{H}_{\text{L}}$ is smaller than that of $\mathcal{H}_{\text{T}}$ in accordance with the discussion given in Section IV-C on multi-pin nets.

We adopt the recursive bipartitioning scheme RB2 and page-packing approach described in [10]. In the RB2 scheme, we use the state-of-the-art hypergraph partitioning tool PaToH [8] for bipartitioning the hypergraphs. Partitioning quality for each dataset is evaluated for four different page sizes of P = 1, 2, 4, and 8 KB. Due to the randomized nature of the heuristics used in PaToH, the experiments are repeated 100 times, and the average performance results are reported in the following figures and tables.

Query processing simulations are performed using page buffers with a capacity of 1, 2, 4, and 8 pages. The Least Recently Used (LRU) page replacement algorithm is employed as the caching algorithm. The synthetic queries used for query log generation are also used in simulations for measuring the total disk access cost. Simulations are performed on a PC that is equipped with an Intel Pentium IV 2.6 GHz processor and 2GB of RAM.

We evaluate the performance of the clustering hypergraph models for the junction- and link-based storage schemes in two aspects. First, we evaluate the partition quality in terms of cutsize, which refers to the total number of disk accesses incurred by *GaS* and *GSs* operations under the single-page buffer assumption. Second, we assess the total number of disk accesses in aggregate network queries through simulations.

B. Partitioning Quality

Fig. 7 displays the partitioning quality of the clustering hypergraph models for the junction- and link-based storage schemes with the link attribute sizes $C_{\text{L}} = 16$ and $C_{\text{L}} = 28$. The results for the hypergraphs generated for the query sets Q_{short} and Q_{long} are presented. As seen in Fig. 7, in all cases the link-based storage scheme achieves smaller cutsize values than the junction-based storage scheme. As expected, the

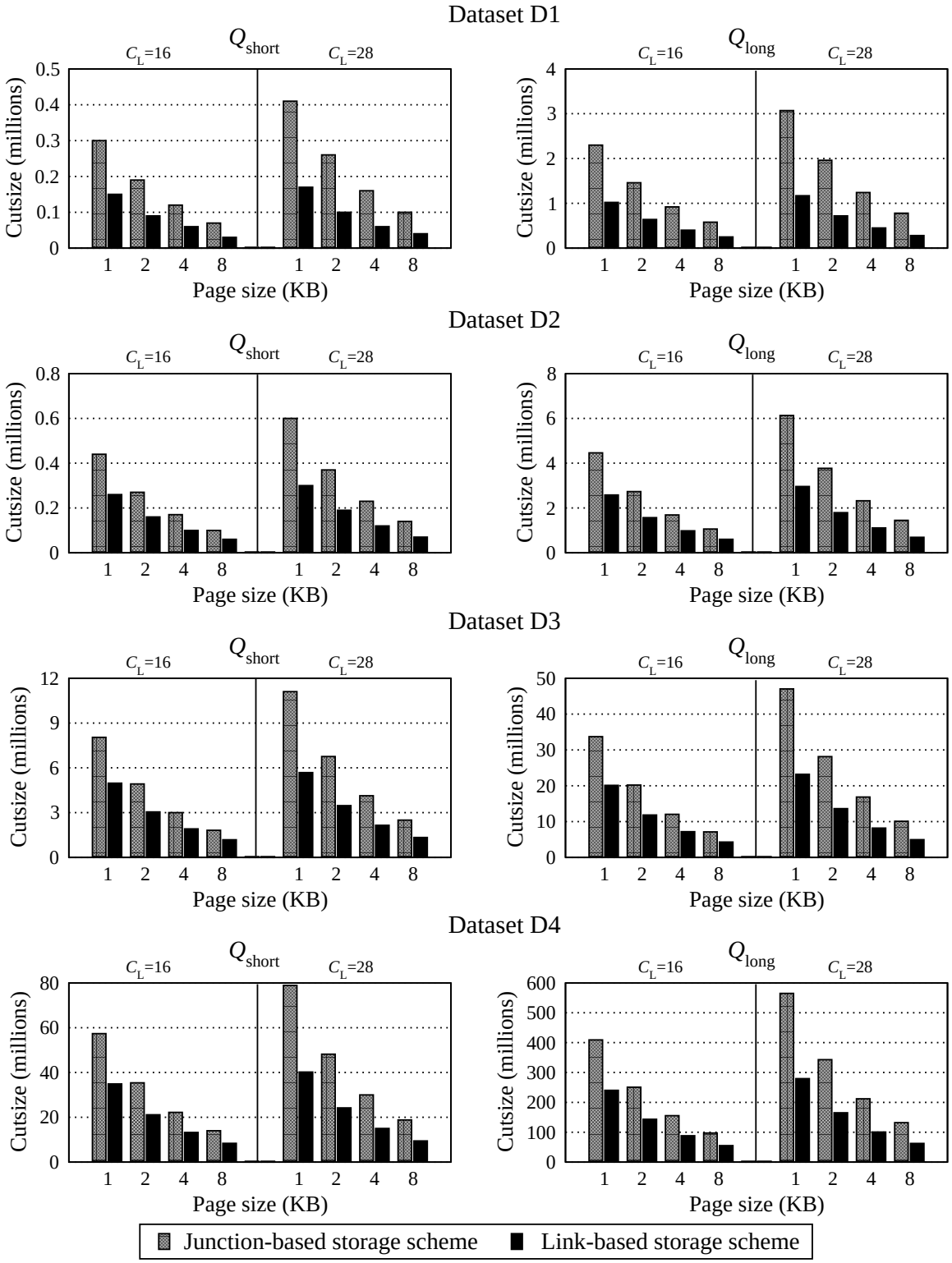

Fig. 7. Partitioning quality of the clustering hypergraph models for the junction- and link-based storage schemes. Cutsize is equal to the number of total disk accesses for aggregate network operations under the single-page buffer assumption.

TABLE V
THE AVERAGE PERFORMANCE IMPROVEMENTS OF THE CLUSTERING HYPERGRAPH MODEL FOR THE LINK-BASED STORAGE SCHEME OVER THE CLUSTERING HYPERGRAPH MODEL FOR THE JUNCTION-BASED STORAGE SCHEME

		Percent improvement					
		$C_{\mathrm{L}} = 16$		$C_{\mathrm{L}} = 28$		$C_{\mathrm{L}} = 40$	
	P	K	Cutsize	K	Cutsize	K	Cutsize
Q_{small}	1	-30.9	42.2	-1.6	51.6	12.8	56.4
	2	-31.0	42.6	-1.9	52.1	12.0	56.8
	4	-31.6	42.1	-2.2	52.0	11.6	57.0
	8	-31.2	41.5	-2.2	51.3	11.6	56.4
Q_{medium}	1	-30.8	43.7	-1.5	53.0	13.0	57.4
	2	-31.1	44.4	-1.9	53.7	12.1	58.2
	4	-31.5	44.0	-1.8	53.5	11.6	58.2
	8	-31.3	44.0	-2.0	53.0	11.5	57.8
Q_{large}	1	-30.7	44.8	-1.5	53.7	12.9	58.0
	2	-31.1	45.8	-1.8	54.8	12.1	59.3
	4	-31.1	45.6	-2.1	55.0	11.6	59.7
	8	-31.6	45.7	-2.4	54.9	11.2	59.4

cutsize values decrease with increasing page size in both storage schemes, whereas the performance gap between these two schemes does not vary significantly with varying page size.

Table V shows the average performance improvements of the clustering hypergraph model for the link-based storage scheme over that for the junction-based storage scheme for all query sets and C_{L} values. In the table, positive values indicate percent decrease in the K and cutsize values, whereas negative values indicate percent increase in the K values, achieved by the link-based storage scheme compared to the junction-based storage scheme. As seen in Table V, the two storage schemes achieve almost equal K values for the $C_{\mathrm{L}} = 28$ case. The junction-based storage scheme achieves 28.4% smaller K values for the $C_{\mathrm{L}} = 16$ case, whereas the link-based storage scheme results in 13.3% smaller K values for the $C_{\mathrm{L}} = 40$ case, on the average. These percent differences are approximately equal to the percent differences for the total storage sizes reported in Table II.

As seen in Table V, for the $C_{\mathrm{L}} = 28$ case, which incurs almost equal K values for both storage schemes, the link-based storage scheme achieves 54.1% less cutsize values than the junction-based storage scheme, on the average. The relative performance improvement of the link-based storage scheme over the junction-based storage scheme increases to 58.6% when the size of the link attributes increases to $C_{\mathrm{L}} = 40$. These experimental findings are in accordance with our expectations discussed in Section IV-C. However, it is interesting to note that, for $C_{\mathrm{L}} = 16$, although the link-based storage scheme leads to considerably higher K values, it achieves considerably lower cutsize values (45.2% on the average). This can be attributed to the properties of the clustering hypergraphs modeling the networks with bidirectional links.

The effect of query sets on the relative performance between the two storage schemes is also important. As seen in Table V, for fixed page size and C_{L} values, the performance gap between the two storage schemes increases as the path length increases in favor of the link-based storage scheme. This finding can be attributed to the increase in the number of *GSs* operations with increasing path length. As mentioned Section IV-C, the relative performance between the two storage schemes is expected to be higher in *GSs* operations compared to the *GaS* operations.

C. Disk Access Simulations

Figs. 8-11 display the relative performance comparisons of the two storage schemes in terms of the number of disk accesses for both route evaluation and path computation queries. The simulation results in these figures are presented for the link attribute sizes $C_{\mathrm{L}} = 16$ and $C_{\mathrm{L}} = 28$ with the varying page and buffer sizes. The query sets Q_{short} and Q_{long} are evaluated in detail to show the effect of path length and number of queries in simulations. The average improvements over all datasets are given in Table VI for all query sets and all C_{L} values.

As seen in Figs. 8-11, the link-based storage scheme outperforms the junction-based storage scheme for almost all simulation cases. In Figs. 8-11, for the $C_{\mathrm{L}} = 16$ case with a single-page buffer, the link-based

TABLE VI
THE AVERAGE PERFORMANCE IMPROVEMENTS OF THE CLUSTERING HYPERGRAPH MODEL FOR THE LINK-BASED STORAGE SCHEME OVER THE CLUSTERING HYPERGRAPH MODEL FOR THE JUNCTION-BASED STORAGE SCHEME

Buffer size	P	Percent improvement $C_{\mathrm{L}}=16$			$C_{\mathrm{L}}=28$			$C_{\mathrm{L}}=40$		
		Q_{short}	Q_{medium}	Q_{long}	Q_{short}	Q_{medium}	Q_{long}	Q_{short}	Q_{medium}	Q_{long}
1	1K	20.7	20.9	21.2	28.5	27.9	27.8	33.4	32.6	32.5
	2K	17.0	17.9	18.2	24.3	23.6	23.6	28.6	27.5	27.4
	4K	13.6	15.3	16.1	21.0	20.4	20.5	25.3	23.6	23.6
	8K	10.2	13.6	14.7	18.3	18.0	18.4	22.6	20.8	20.8
2	1K	19.7	20.6	21.0	29.1	28.2	28.2	34.5	33.1	33.0
	2K	15.0	17.1	17.7	24.8	23.9	23.9	30.1	28.1	28.0
	4K	9.8	13.7	15.0	21.4	20.5	20.6	27.0	24.3	24.2
	8K	4.4	11.1	12.8	17.9	17.8	18.3	24.5	21.6	21.4
4	1K	16.9	19.5	20.3	29.3	28.5	28.4	35.6	33.7	33.4
	2K	10.3	15.2	16.3	24.8	24.2	24.1	31.7	29.1	28.8
	4K	2.7	10.1	12.4	20.8	20.5	20.7	29.2	25.6	25.3
	8K	-4.3	5.4	8.3	16.3	17.2	17.9	26.2	23.1	22.6
8	1K	11.0	17.2	18.6	28.7	28.9	28.8	36.7	34.9	34.3
	2K	2.4	10.8	12.9	23.3	24.5	24.4	33.1	31.0	30.2
	4K	-4.7	1.3	5.9	18.3	20.5	20.6	29.9	28.1	27.2
	8K	-10.7	-10.3	-3.4	13.3	15.9	16.8	24.7	26.2	24.9

storage scheme performs better than the junction-based storage scheme in all simulations except for the case of D1 with $P=8$ and Q_{short}. For the $C_{\mathrm{L}}=16$ case with larger page and buffer sizes, especially with short queries, the junction-based storage scheme performs slightly better than the link-based storage scheme. This is due to the fact that average record sizes are almost equal but the total storage of the the link-based storage scheme is 29% larger than that of the junction-based storage scheme.

The comparison of the two storage schemes in Table VI is consistent with the results presented in Table V. However, the final improvements in the simulations are less than the improvements in actual total costs of *GaS* and *GSs* operations. As seen in Table V, the average improvement in the total disk access cost of *GaS* and *GSs* operations for a single-page buffer is 43.9% and 53.2% for $C_{\mathrm{L}}=16$ and for $C_{\mathrm{L}}=28$, respectively. Nevertheless, in Table VI, the average improvement in the total disk access cost of aggregate network queries for a single-page buffer is 11.7% and 22.6% for $C_{\mathrm{L}}=16$ and for $C_{\mathrm{L}}=28$, respectively. This is mainly due to the additional overhead of *Find* operations incurred by the internal steps of the shortest path algorithm used in path computation queries.

According to Figs. 8–11, as expected, increasing page size and increasing buffer size independently decrease the number of disk accesses in the two storage schemes. The performance gap between the storage schemes decreases with increasing P. There are even few cases in which the junction-based storage scheme performs better than the link-bases storage scheme. These cases occur for the simulations on the smaller datasets with larger buffer and page sizes because considerable portion of the data can reside in memory.

VI. CONCLUDING REMARKS

We proposed the link-based storage scheme for efficient aggregate query processing on clustered road networks. In this storage scheme, each record stores the data associated with a link together with the link's connectivity information. We extended our earlier clustering hypergraph model for the junction-based storage scheme to the link-based storage scheme. Our detailed comparative analysis on the properties of the junction- and link-based storage schemes showed that the link-based storage scheme is more amenable to clustering. Moreover, we introduced storage enhancements for bidirectional networks. We showed that the link-based storage scheme is more amenable to our enhancements than the junction-based storage scheme and results in better data allocation for processing aggregate network queries. Extensive experimental comparisons were carried out on the effects of page size, buffer size, path length, record size, and dataset size for the junction- and link-based storage schemes. Experimental results showed that

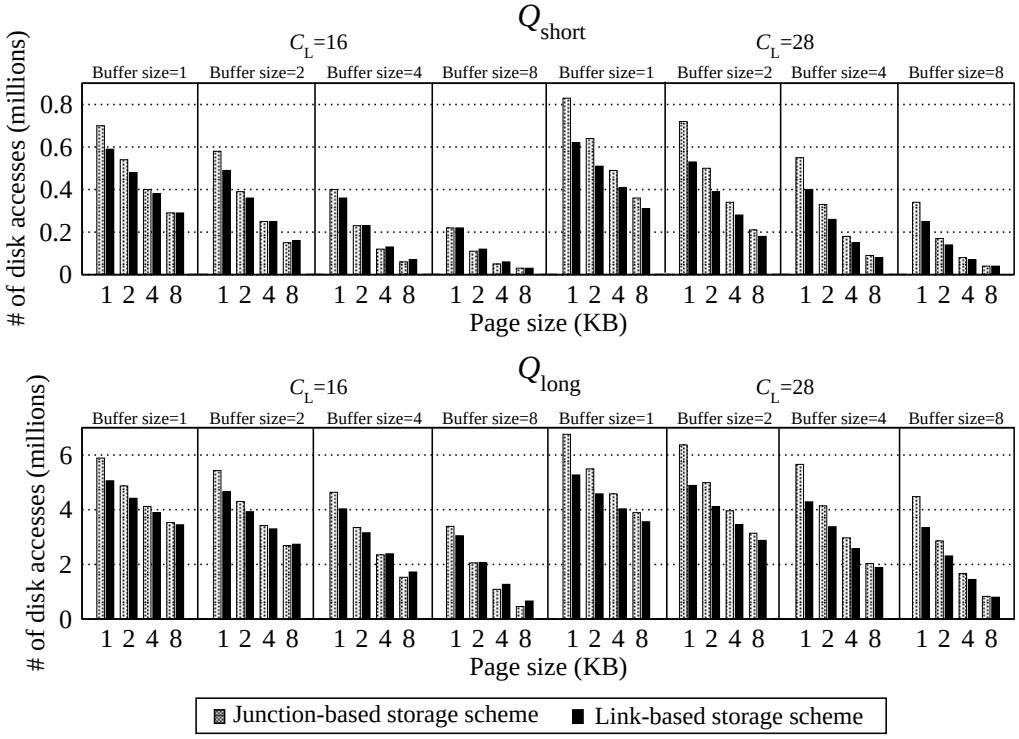

Fig. 8. Disk access comparisons of the clustering hypergraph models for the two storage schemes in aggregate network query simulations over dataset D1.

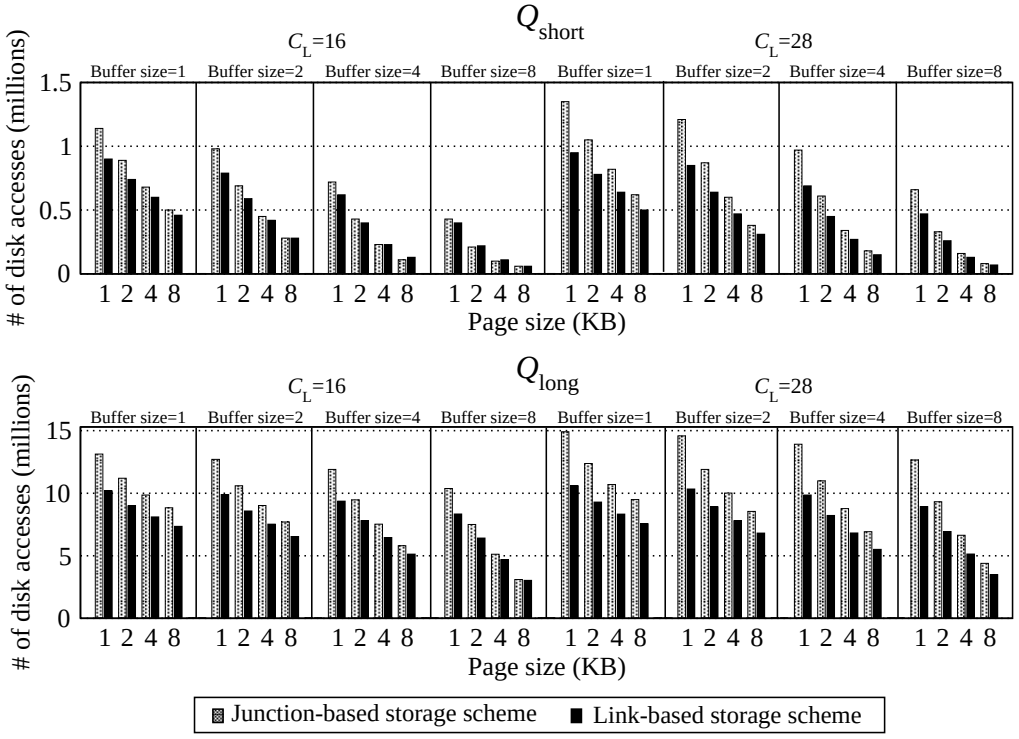

Fig. 9. Disk access comparisons of the clustering hypergraph models for the two storage schemes in aggregate network query simulations over dataset D2.

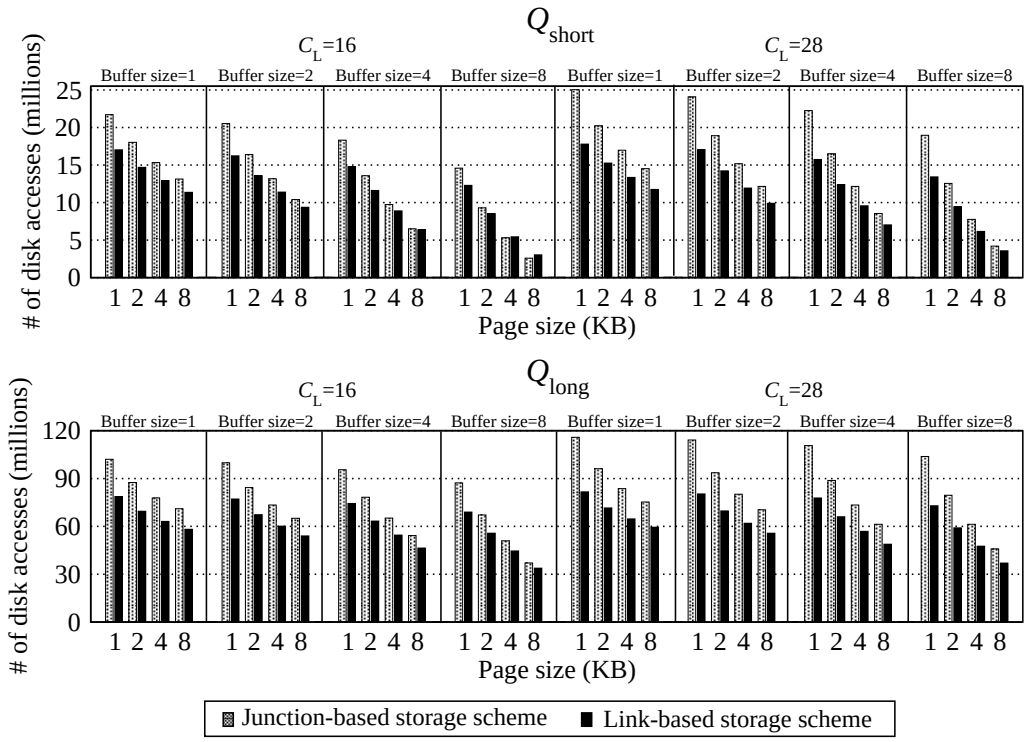

Fig. 10. Disk access comparisons of the clustering hypergraph models for the two storage schemes in aggregate network query simulations over dataset D3.

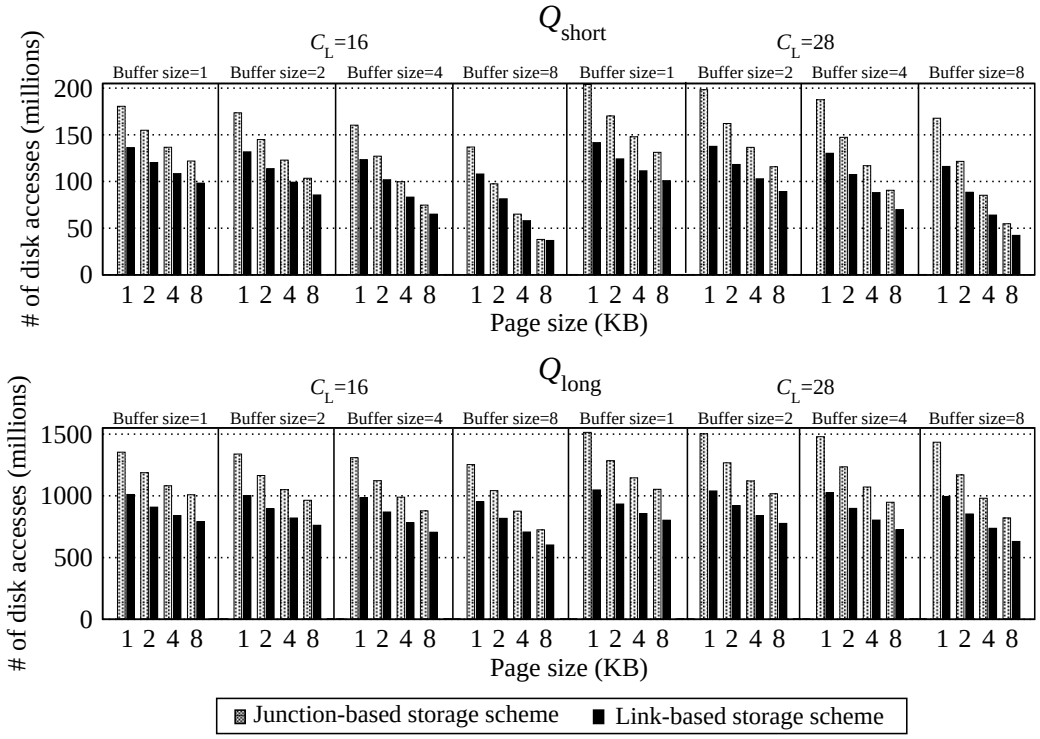

Fig. 11. Disk access comparisons of the clustering hypergraph models for the two storage schemes in aggregate network query simulations over dataset D4.

the link-based storage scheme outperforms the widely-used junction-based storage scheme in terms of both storage and query processing efficiency.

The storage schemes mentioned in this work are generic representations of networks, hence any index can be built on top of these storage schemes. Application of the link-based storage scheme in graph topologies may also be beneficial for research on problems in other research fields.

REFERENCES

[1] V. T. Almeida and R. H. Güting, "Using Dijkstra's algorithm to incrementally find the k-nearest neighbors in spatial network databases," in *Proc. ACM Int'l Symp. on Applied Computing*, 2006, pp. 23–27.

[2] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *VLSI Journal*, vol. 19, no. 1-2, pp. 1–81, 1995.

[3] C. Aykanat, A.Pinar, and U. V. Çatalyürek, "Permuting sparse rectangular matrices into block-diagonal form," *SIAM Journal of Scientific Computing*, vol. 25, no. 6, pp. 1860–1879, 2004.

[4] C. Berge, *Graphs and Hypergraphs*. North-Holland Publishing Company, 1973.

[5] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[6] T. Caldwell, "On finding minimum routes in a network with turn penalties," in *Communications of the ACM*, 1961, pp. 107–108.

[7] U. V. Çatalyürek and C. Aykanat, "Hypergraph-partitioning-based decomposition of parallel sparse-matrix vector multiplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 7, pp. 673–693, 1999.

[8] ——, "PaToH: Partitioning tool for hypergraphs," Computer Engineering Department, Bilkent University, Tech. Rep., 1999, http://www.cs.bilkent.edu.tr/~aykanat/pargrp/patoh/.

[9] A. Dasdan and C. Aykanat, "Two novel multiway circuit partitioning algorithms using relaxed locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 16, no. 2, pp. 169–178, 1997.

[10] E. Demir, C. Aykanat, and B. Cambazoglu, "Clustering spatial networks for aggregate query processing: A hypergraph approach," *Information Systems*, revised version under review.

[11] Y.-W. Huang, N. Jing, and E. Rundensteiner, "Effective graph clustering for path queries in digital map databases," in *Proc. ACM Int'l Conf. Information and Knowledge Management*, 1996, pp. 215–222.

[12] C. Jensen, J. Kolar, T. Pedersen, and I. Timko, "Nearest neighbor queries in road networks," in *Proc. ACM Int'l Workshop on Geographic Information Sytems*, 2003, pp. 1–8.

[13] S. Jung and S. Pramanik, "An efficient path computation model for hierarchically structured topographical road maps," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1029–1046, 2002.

[14] M. Kolahdouzan and C. Shahabi, "Alternative solutions for continuous K nearest neighbor queries in spatial network databases," *GeoInformatica*, vol. 9, no. 4, pp. 321–341, 2005.

[15] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng, "On trip planning queries in spatial databases," in *Proc. 9th Int'l Symp. on Spatial and Temporal Databases*, 2005, pp. 273–290.

[16] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. Int'l Conf. Very Large Data Bases*, 2003, pp. 790–801.

[17] S. Shekhar, A. Kohli, and M. Coyle, "Path computation algorithms for advanced traveler information systems," in *Proc. IEEE Int'l Conf. on Data Engineering*, 1993, pp. 31–39.

[18] S. Shekhar and D. R. Liu, "A connectivity-based access method for networks and network computation," *IEEE Trans. Knowl. Data Eng.*, vol. 9, no. 1, pp. 102–117, 1997.

[19] (2002) Topologically integrated geographic encoding and referencing system (TIGER). [Online]. Available: http://www.census.gov/geo/www/tiger/

[20] B. Ucar and C. Aykanat, "Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies," *SIAM Journal of Scientific Computing*, vol. 25, no. 6, pp. 1837–1859, 2004.

[21] ——, "Revisiting hypergraph models for sparse matrix partitioning," SIAM Review, December 2007, in press.

[22] (2004) US department of transportation federal highway administration, the national highway planning network. [Online]. Available: http://www.fhwa.dot.gov/planning/nhpn/index.html

[23] S. Winter, "Weighting the path continuation in route planning," in *Proc. 9th ACM Int'l Symp. on Advances in GIS*, 2001, pp. 173–176.

[24] ——, "Route specifications with a linear dual graph," in *Proc. Int'l Symp. Advances in Spatial Data Handling*, 2002, pp. 329–338.

[25] S.-H. Woo and S.-B. Yang, "An improved network clustering method for I/O-efficient query processing," in *Proc. ACM Symp. on GIS*, 2000, pp. 62–68.

[26] M. L. Yiu and N. Mamoulis, "Clustering objects on a spatial network," in *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, 2004, pp. 13–18.

[27] M. L. Yiu, N. Mamoulis, and D. Papadias, "Aggregate nearest neighbor queries in road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 820–833, 2005.