A Graph Search Heuristic for Shortest Distance Paths

Edmond Chow
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Box 808, L-560, Livermore, CA 94551, USA
echow@llnl.gov

Abstract

This paper presents a heuristic for guiding A* search for finding the shortest distance path between two vertices in a connected, undirected, and explicitly stored graph. The heuristic requires a small amount of data to be stored at each vertex. The heuristic has application to quickly detecting relationships between two vertices in a large information or knowledge network. We compare the performance of this heuristic with breadth-first search on graphs with various topological properties. The results show that one or more orders of magnitude improvement in the number of vertices expanded is possible for large graphs, including Poisson random graphs.

Introduction

Background and motivation.

The structure of the interconnections among a set of entities is increasingly being recognized as important data which can be used to infer latent information about the entities themselves or about the network as a whole. Examples of these network or graph structures include the World-Wide Web, the Internet, and social networks. Structures called bipartite cores mined from the Web, for example, can identify a set of web sites that form a knowledge base on a common subject (Kumar *et al.* 1999); see also (Kuramochi & Karypis 2002; Abello, Resende, & Sudarsky 2002). The now well-known PageRank (Brin & Page 1998) and HITS (Kleinberg 1999) algorithms use network structure to determine which web pages are more important than others in a specific sense. For networks of heterogeneous data, sometimes called *relational data graphs*, a query on the data may be posed as a subgraph search problem, with the query expressed in the form of a subgraph (Coffman, Greenblatt, & Marcus 2004). A *connection subgraph* is a connected subgraph that consists of a set of "interesting" paths between two non-adjacent vertices (Faloutsos, McCurley, & Tomkins 2004). For a social network, a connection subgraph between two individuals represents how they are related.

This paper addresses the problem of finding relationships between two vertices in an explicitly defined graph, which may be a knowledge or information network. Given an undirected graph and two vertices, we consider the problem of finding a path between the given vertices with the smallest number of edges. We call such a path a *shortest distance path* and also call the number of edges in this path the *distance* between the two vertices. A simple variant is to find all edges on all the shortest distance paths between two vertices. This set of edges may be used, for instance, as a starting point for a connection subgraph calculation.

We note that although "interesting" paths tend to be shortest distance paths, this is not always the case. In particular, paths through hubs, or highly connected vertices, often do not give insightful information (Faloutsos, McCurley, & Tomkins 2004). It is not difficult in this case, however, to constrain the path search to avoid certain vertices or edges.

The straight-forward approach for solving a shortest distance path problem is to use breadth-first search (BFS) starting at one of the vertices, or bi-directional search starting at both vertices simultaneously. However, for almost all real-world data of interest, the graphs have the *small-world* property (Watts & Strogatz 1998), that is, they have small average distance between a pair of vertices, commonly called small *average path length*. Thus, a breadth-first search will cover a large portion of the entire graph after a small number of steps, and is very costly for very large graphs. (Searching graphs with large average path length is not considered costly since searches are stopped after a given threshold distance; it is assumed no interesting relationship exists between vertices that are far apart.)

This paper advocates the use of heuristic search to guide and accelerate path search. We present a heuristic for shortest distance paths that does not need any information except the structure of the graph itself. The heuristic is constructed as a preprocessing step, and this cost is amortized in the usual case that many searches on the same graph are performed. The method is useful when the graph is very large and unguided search procedures are too costly.

Many graph search algorithms utilize specific information about the graph to be efficient. For example, in shortest path search in transportation networks, the coordinates of the vertices are utilized (Schulz, Wagner, & Weihe 2000). Many networks are hierarchical, and finding a short path between two vertices using this hierarchy can be very efficient. In particular, sending a message from one computer to another on the Internet uses the Internet's hierarchical structure. Algorithms have also been designed for searching peer-to-peer networks that utilize the power-law property of these networks, a property that implies the ex-

istence of highly-connected nodes (Adamic *et al.* 2001; Simsek & Jensen 2005). In contrast to the above, the heuristic presented here does not depend on any specific properties of the graph.

Our heuristic is inspired by the small-world experiment (Milgram 1967; Travers & Milgram 1969). In this experiment, participants were asked to send a letter to an unfamiliar stockbroker in Boston by forwarding the letter to an acquaintance who might know the target or be able to send the letter to someone else who knows the target. The surprising result at that time was that the target was reached via a short chain of people. Recently, however, researchers have focused on the problem of *how* these short chains were found (the reverse small-world experiment), and what properties of the network allow these short chains to be found (Kleinberg 2000a; Watts, Dodds, & Newman 2002; Kleinberg 2000b; 2001; Huberman & Adamic 2004). Naturally, people used information about the similarity of their acquaintances to the target to decide how to forward the letter. Using this type of *decentralized search*, it was found that various network properties will affect how readily short chains could be uncovered.

In the absence of attribute information about the vertices (such as occupation or location in the case of the small-world experiment) and how vertices may be connected to other vertices with similar attributes (a concept called *homophily*), we present a heuristic that constructs a proxy for attribute information. In particular, the heuristic constructs the distance of all vertices to a set of arbitrary "influential" vertices, called centers. This will be described and tested in the remainder of this paper.

This heuristic appears to have been originally proposed for Internet routing in a manuscript that was never published (Hotz 1994 draft). The heuristic, however, is described in some subsequent publications by researchers interested in predicting Internet network distance (Ng & Zhang 2002). In this paper, we prove some simple properties for this heuristic, and then demonstrate its performance on a variety of graphs with a wide range of average path length. Results on an Internet graph are also shown.

Heuristic search.

A* search is a well-known graph search algorithm that utilizes a heuristic to guide its search (Hart, Nilsson, & Raphael 1968). With a heuristic that satisfies certain conditions, A* search is guaranteed to find a path to the target vertex with smallest path cost and does so optimally efficiently in a specific sense. In our case, the cost of a path between two vertices is defined to be the distance between those two vertices. To explain A* search briefly, at each step, the algorithm expands the vertex n from a list called OPEN that has the lowest value of the function

$$f(n) = g(n) + h(n)$$

where $g(n)$ is the actual cost of reaching vertex n from the start vertex, and $h(n)$ is the estimated cost of reaching the target vertex from vertex n. The function $h(n)$ is the heuristic function. To expand a vertex n, the algorithm computes f for all the neighbors of n that have not been encountered earlier. These neighbors are put on the OPEN list, and n is removed from the OPEN list and put on the CLOSED list. If the heuristic is not monotone (to be described later), then it may be necessary also to recompute f for vertices on the CLOSED list. For more precise details, see (Hart, Nilsson, & Raphael 1968).

In summary, this paper describes a heuristic $h(n)$ for arbitrary unweighted and undirected graphs that can be used with A* search to find the shortest distance path between two vertices. The heuristic requires a small amount of information to be stored at each vertex. Next, we present this heuristic. Later in the paper we will show the performance of this heuristic with respect to some general topological properties.

Level-difference heuristic

Heuristic.

Given a vertex c, the *level* of a vertex n relative to c is the distance between n and c, and is denoted by $l_c(n)$. The level of every vertex in a connected graph relative to c can be computed by a breadth-first algorithm starting at c. Here a vertex c is called a *center*, and the level of the center itself is 0.

Given a center c and a target vertex t, a lower bound on the distance between any vertex n and t is

$$h_c(n) \equiv | \, l_c(n) - l_c(t) \, |$$

which can be used as a heuristic function. To improve this heuristic, select a set of center vertices C, and define the new heuristic

$$h(n) \equiv \max_{j \in C} \{h_j(n)\} \, .$$

We refer to this heuristic as the *level-difference*, or LD heuristic. This heuristic is better than any of the individual $h_j(n)$ in the sense that A* search using $h(n)$ will never expand more vertices than using any subset of the $h_j(n)$. Note that the value of $h(n)$ is generally *not* equal to $h_j(n)$ for the center j closest to the target or for the center j closest to n.

If it is inexpensive to determine whether or not a vertex n is a target vertex t, then we can define a *modified* level-difference heuristic

$$h'(n) \equiv \begin{cases} \max\{h(n), 1\}, & n \neq t \\ 0, & \text{otherwise.} \end{cases}$$

This modification may be useful if a very small number of centers is used. We use this heuristic for the simulations later in this paper.

Properties of the heuristic.

Property 1 *The LD heuristic is admissible.*

A heuristic is *admissible* when it never overestimates the actual distance to the target. A* using an admissible heuristic is optimal and will return the shortest distance path.

Property 2 *The LD heuristic is* monotone *(also called* consistent*), that is,*

$$h(n) \leq 1 + h(n') \quad \forall n$$

where n' is a neighbor of n and the 1 implies unit edge costs.

Proof. First, we note that for a given c, $l_c(n)$ and $l_c(n')$ cannot differ by more than unity and thus $h_c(n)$ and $h_c(n')$ cannot differ by more than unity.

For some n, let c be the center used to estimate $h(n)$ and let d be the center used to estimate $h(n')$. The centers c and d may be different or identical. Then,

$$h_d(n') \geq h_c(n') \geq h_c(n) - 1$$

or in other words,

$$h_c(n) \leq 1 + h_d(n').$$

□

Monotonicity implies that once A* expands a vertex, it is guaranteed to have found the shortest distance path to that vertex. This somewhat simplifies A* search by not needing to re-open CLOSED vertices.

Property 3 *The error in the LD heuristic at a vertex of distance l from the target is bounded by l.*

Proof. For a vertex that is distance l from the target, the heuristic estimate can take on values from 0 to l. Thus, the largest error that can be incurred is l. □

The above property says that the heuristic is more accurate closer to the target vertex. The bound is poor, but it says that the heuristic cannot make A* perform worse than breadth-first search.

Property 4 *The error in the LD heuristic at a vertex of distance k from a center is bounded by $2k$.*

Proof. Let l_{AB} denote the length of the shortest path between vertices A and B, and let h denote the heuristic estimate of the distance between A and B. The error in the heuristic estimate is $e = l_{AB} - h \geq 0$. If $l_{AB} \leq 2k$, then $e \leq 2k$ as required. Thus it remains to consider the case where $l_{AB} > 2k$.

Let A be k steps from a center vertex D. D may be the center used in the heuristic estimate, or it may be a different center. By definition,

$$l_{DA} = k.$$

Since the shortest distance between A and B is l_{AB}, we have

$$l_{DA} + l_{DB} \geq l_{AB}.$$

The above two statements imply that

$$l_{DB} \geq l_{AB} - k. \quad (1)$$

Along with $l_{AB} > 2k$, we also have the implication $l_{DB} \geq l_{DA}$. Now, the difference in level numbers using center D must not be greater than the heuristic estimate,

$$l_{DB} - l_{DA} \leq h = l_{AB} - e$$

thus

$$\begin{aligned} e &\leq l_{AB} - l_{DB} + l_{DA} \\ &= l_{AB} - l_{DB} + k \\ &\leq 2k \end{aligned}$$

where the last inequality utilized (1). □

It is not difficult to see that the bound of $2k$ can be attained. A corollary to the above property is that the error in the heuristic value is bounded by $2k$, where k is the maximum distance of any vertex to its nearest center vertex.

An inadmissible heuristic.

A related heuristic is the following. Given a center c and a target vertex t, an *upper* bound on the distance between any vertex n and t is

$$H_c(n) \equiv l_c(n) + l_c(t)$$

and if multiple centers are used, a heuristic can be defined as

$$H(n) \equiv \min_{j \in C} \{H_j(n)\}.$$

A weighted combination of h and H has been suggested, but is not recommended, according to (Ng & Zhang 2002). It is easy to see that H and thus the combined heuristic are inadmissible. The heuristic also violates the condition that the heuristic value for a target vertex is zero. This latter condition is used to prove that an inadmissible heuristic cannot be monotone. (Interestingly, H would otherwise be monotone in our case.) Empirically, we did not find that a combined heuristic was more effective than the LD heuristic by itself.

Selecting the centers.

The center vertices should be chosen to minimize the error in the heuristic function. This may be accomplished approximately by choosing the center vertices to minimize the *average* distance of all vertices to their nearest centers. In practice, it may be easier to minimize the *maximum* distance. Other options are described in (Ng & Zhang 2002), although for a different application. Different algorithms are suitable for graphs with different properties. For the tests performed in this paper, the centers were selected randomly. We note that if it is known that certain vertices are repeatedly used as targets, then these target vertices should be selected as center vertices.

Empirical tests

In this section, we empirically test A* with the LD heuristic and compare its performance with BFS in terms of number of vertices visited. We wish to understand the performance of the LD heuristic with respect to the type of the graph, its size, and its average degree. For this reason, we test model graphs that were generated with specific properties. We also test one sample graph (a small portion of the Internet) and show results for this graph that are qualitatively similar to the results of one of the model graphs.

We consider three classes of graphs: Poisson random graphs, spatial graphs, and meshes. These three classes cover the wide range of very disordered graphs with small diameter (Poisson random graphs) and very structured graphs with large diameter (meshes). *Small-world* networks (Watts & Strogatz 1998) parameterize between these two extremes, and we consider the example of spatial graphs in this case. All the graphs we consider are undirected.

In a Poisson random graph, the probability of an edge between any two vertices is constant (Bollobás 2001). We generated random graphs with 64,000 and 128,000 vertices and with average degree 2 or 6. We then extracted the largest connected components of each graph. Table 1 shows the details of the largest connected components of three random graphs that we used in the following tests.

In spatial graphs, vertices are associated with geometric coordinates. The probability of an edge between two vertices in a spatial graph depends on the distance between those vertices. For our spatial graphs, we select 2-D coordinates for each vertex randomly, and set the probability of an edge between vertices i and j to be proportional to $d(i,j)^{-\alpha}$ where $d(i,j)$ is the Euclidean distance (computed with periodic boundary conditions) between vertices i and j, and $\alpha \geq 0$ is a parameter that controls the diameter and clustering in the graph. For $\alpha = 0$, we have Poisson random graphs, and for large α, we have mesh-like graphs that have large diameter and a high measure of clustering called *clustering coefficient* (Watts & Strogatz 1998). In practice, however, graphs defined exactly this way are costly to generate. We used the following approximate algorithm. For each vertex in order, we link it to m other vertices with probability proportional to $d(i,j)^{-\alpha}$ (same as above). The parameter m controls the average degree of the graph. For our test, we used a graph with $\alpha = 4$, which is large enough to show significant differences from both Poisson random graphs and meshes. The average degree of this graph was set to 6.

We generated a 2-D mesh graph by using a triangular mesh generator over a circle (approximated by a polygon). Table 1 shows the details for this graph.

For our realistic graph we use a small sample of the Internet obtained from traceroutes collected by the Internet Mapping project at Lucent Bell Laboratories circa November 1999. The network has 112969 routers and 181639 links between them. This real-world network has moderate clustering coefficient and a scale-free degree distribution (Albert & Barabási 2002). We will show that with respect to graph search, this network is qualitatively similar to a spatial network.

	vertices	edges	ave degree	ave path length	diam
Random	63848	191999	6.0	6.4	12
Random	51065	61415	2.4	15.0	39
Random	127664	383997	6.0	6.8	12
Spatial	64000	192000	6.0	21.3	39
2-D Mesh	64304	192043	6.0	116.4	278
Internet	112969	181639	3.2	9.9	27

Table 1: Test graphs, listing the number of vertices, number of edges, average degree, average path length, and diameter. The latter two were computed by sampling the result of 1000 breadth-first expansions.

Figures 1–4 show results for 4 different graphs. The figures plot the number of vertices visited by BFS and by A* as a function of the distance between a pair of source and target vertices. Distances of 1 to at most the average path length were used. (Larger distances have results significantly affected by the finite size of the graphs.) For each graph, 10 trials were used. In each trial, the centers were selected randomly, and 10 pairs of vertices were randomly selected as the sources and targets for each given distance. Each figure plots the average for each given distance over all trials. This data may be used to compute the effective branching factors, but we believe the results presented this way are more informative.

In Figure 1, the results are shown for a Poisson random graph with 63,848 vertices and average degree 6. The results show that A* performs significantly better than BFS, even with 1 center. Using additional centers further improves the results but give diminishing returns. (The downward curve of the plot for BFS is attributed to the finite size of the graph.) By comparing the slopes of the plots, the complexity of A* appears smaller than the complexity of BFS for small distances, but appear to be the same for large distances. Thus in the initial stages of the search, the branching factors for A* are much lower than the branching factors for BFS, but then increases as the search progresses.

In Figure 2, the results are shown for the spatial graph. Again, A* gives a significant improvement, even for 1 center. Compared to random graphs, in this class of graphs, the complexity of both BFS and A* decreases as the search progresses. Also, the complexity of A* is better than the complexity of BFS and improves when more centers are used. The random graph of Figure 1 and the spatial graph have similar numbers of vertices and edges, but the average path length for the spatial graph is larger than that for the random graph. For a search of distance 6, much fewer vertices are visited when searching spatial graphs compared to searching random graphs.

Figure 3 shows comparable results for a 2-D mesh. The results are qualitatively similar to the results for the spatial graph of Figure 2 (which has a relatively high value of α). An exception is that for the mesh graph, the diminishing returns when using more centers is more dramatic: the results are very similar whether using 16 or 64 centers. Also, for the same search distance, fewer vertices are visited compared to spatial graphs.

Figure 4 shows results for a small portion of an Internet graph. The results are very similar to the results for the spatial network. The diminishing returns for adding centers is more similar to that of the spatial network than that of the mesh. In terms of graph search, it appears possible to model the Internet graph as a spatial network. Although we did not attempt this, it may be possible to determine an equivalent value of α for the Internet graph.

In Figure 5, we wish to understand the scaling behavior of the LD heuristic, that is, how it behaves for large graphs. Figure 5 plots the results for two random graphs, with approximately 64,000 vertices and 128,000 vertices and approximately the same average degree. The figure shows that for both BFS and heuristic search with 4 centers the results are changed very little for the larger graph. This implies that much larger graphs may be searched with nearly the same performance.

In Figure 6, we wish to understand the effect of average degree on the behavior of graph search. We tested random graphs with average degree 6 and average degree 2.4. For larger average degree, the effective branching factors are larger. The improvement of A* over BFS appears comparable in both cases, but appears slightly greater for lower average degree.

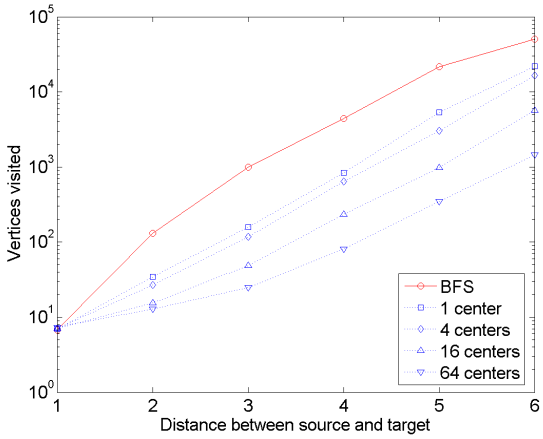

Figure 1: Test results for a random graph with 63,848 vertices and average degree 6.

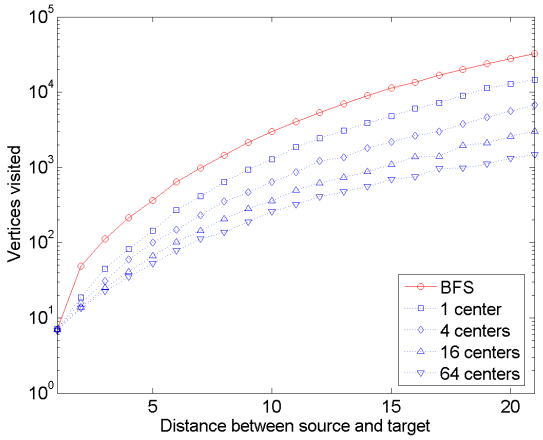

Figure 2: Test results for a spatial graph with $\alpha = 4$, 64,000 vertices and average degree 6.

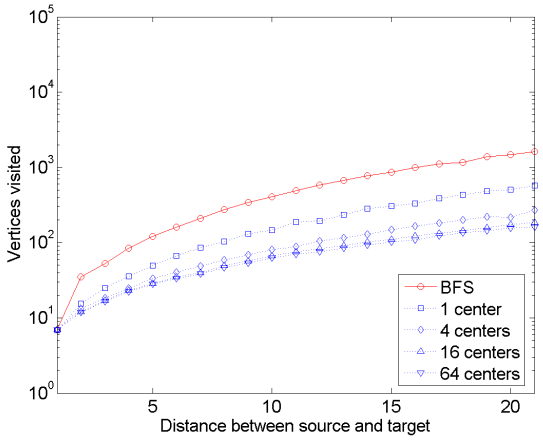

Figure 3: Test results for a 2-D mesh with 64,304 vertices and average degree 6.

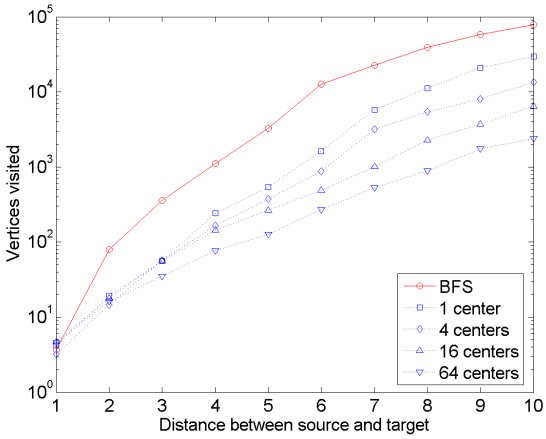

Figure 4: Test results for a portion of the Internet graph with 112,969 vertices and average degree 3.

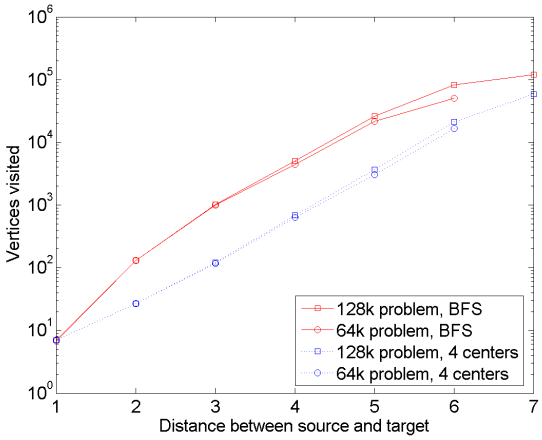

Figure 5: Test results for two random graphs, with approximately 64,000 and 128,000 vertices and average degree 6.

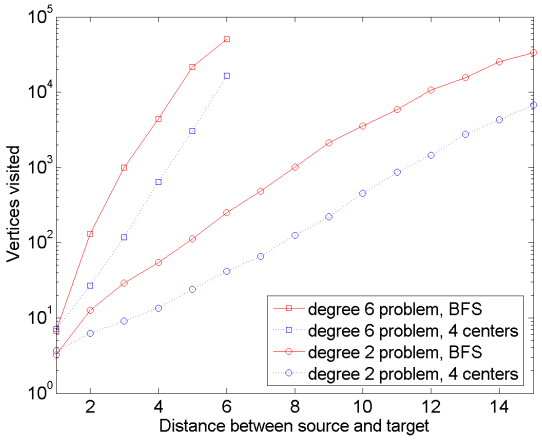

Figure 6: Test results for two random graphs with differing average degree.

Conclusions and future work

In this paper, we presented a heuristic search procedure for finding the shortest distance path between two vertices in a connected, undirected graph. The method is useful when the graph is very large and unguided search procedures are too costly. Compared to breadth-first search, orders of magnitude improvement in the number of vertices expanded are possible, using only a small amount of storage at each vertex. We empirically demonstrated these results for different classes of graphs. For a given distance between source and target vertices, graphs with larger diameter and higher clustering coefficient are less costly to search.

We observed that a significant reduction in the number of vertices visited can be achieved with heuristic search with even 1 center vertex. Additional center vertices give diminishing returns, but may be worthwhile if the cost of expanding vertices is high, e.g., in the case of distributed parallel search. The performance of the heuristic appears insensitive to graph size for random graphs and a fixed number of centers. We also observed that, for graph search, the Internet graph appears similar to a spatial graph with large α.

This work opens a number of unanswered questions. Various methods can be devised for selecting the center vertices, and it is anticipated that different methods are suitable for random graphs and for graphs with power-law degree distributions, for example. Also, in this paper, we did not try to correlate the performance of heuristic search with any measure of the error in the heuristic. Again, we anticipate that these correlations will be very dependent on topological properties of the graphs.

Acknowledgments

The author wishes to thank Lada Adamic for referring us to (Ng & Zhang 2002), as well as Tina Eliassi-Rad, David Eppstein, and Keith Henderson for helpful comments. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

Abello, J.; Resende, M. G. C.; and Sudarsky, S. 2002. Massive quasi-clique detection. In Rajsbaum, S., ed., *LATIN 2002: Theoretical Informatics*, 598–612. Springer-Verlag.

Adamic, L. A.; Lukose, R. M.; Puniyani, A. R.; and Huberman, B. A. 2001. Search in power-law networks. *Physical Review E* 64.

Albert, R., and Barabási, A.-L. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74:47–97.

Bollobás, B. 2001. *Random Graphs*. Cambridge: Cambridge University Press, 2nd edition.

Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual (web) search engine. In *Proc. 7th International World Wide Web Conference (WWW7)/Computer Networks*, volume 30, 107–117.

Coffman, T.; Greenblatt, S.; and Marcus, S. 2004. Graph-based technologies for intelligence analysis. *Comm. ACM* 47:45–47.

Faloutsos, C.; McCurley, K.; and Tomkins, A. 2004. Fast discovery of connection subgraphs. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 118–127. Seattle, WA, USA: ACM Press.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basic for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Sci. and Cybernetics* 4:100–107.

Hotz, S. M. 1994 (draft). *Routing information organization to support scalable interdomain routing with heterogeneous path requirements*. Ph.D. Dissertation, University of Southern California.

Huberman, B. A., and Adamic, L. A. 2004. Information dynamics in the networked world. In Ben-Naim, E.; Frauenfelder, H.; and Toroczkai, Z., eds., *Complex Networks*, 371–398. Springer.

Kleinberg, J. M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46(5):604–632.

Kleinberg, J. 2000a. Navigation in a small world. *Nature* 406:845.

Kleinberg, J. 2000b. The small-world phenomenon: An algorithmic perspective. In *32nd ACM Symposium on Theory of Computing*. ACM.

Kleinberg, J. 2001. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS) 14*. MIT Press.

Kumar, R.; Raghavan, P.; Rajagopalan, S.; and Tomkins, A. 1999. Extracting large-scale knowledge bases from the web. In *The VLDB Journal*, 639–650.

Kuramochi, M., and Karypis, G. 2002. An efficient algorithm for discovering frequent subgraphs. Technical Report UMSI 02-026, Department of Computer Science, University of Minnesota.

Milgram, S. 1967. The small world problem. *Psychology Today* 2:60–67.

Ng, T. S. E., and Zhang, H. 2002. Predicting internet network distance with coordinates-based approaches. In *INFOCOM'02*.

Schulz, F.; Wagner, D.; and Weihe, K. 2000. Dijkstra's algorithm on-line: An emperical case study from public railroad transport. *ACM J. Exp. Algorithmics* 5.

Simsek, O., and Jensen, D. 2005. Decentralized search in networks using homophily and degree disparity. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*.

Travers, J., and Milgram, S. 1969. An experimental study of the small world problem. *Sociometry* 32:425–443.

Watts, D. J., and Strogatz, S. H. 1998. Collective dynamics of small-world networks. *Nature* 393:440–442.

Watts, D. J.; Dodds, P. S.; and Newman, M. E. J. 2002. Identity and search in social networks. *Science* 296:1302–1305.