Degrees of Separation in Social Networks

Reza Bakhshandeh
Shiraz University
Shiraz, Iran
bakhshandeh@cse.shirazu.ac.ir

Mehdi Samadi
Carnegie Mellon University
Pittsburgh, United States
msamadi@cs.cmu.edu

Zohreh Azimifar
Shiraz University
Shiraz, Iran
azimifar@cse.shirazu.ac.ir

Jonathan Schaeffer
University of Alberta
Edmonton, Canada
jonathan@cs.ualberta.ca

Abstract

Social networks play an increasingly important role in today's society. Special characteristics of these networks make them challenging domains for the search community. In particular, social networks of users can be viewed as search graphs of nodes, where the cost of obtaining information about a node can be very high. This paper addresses the search problem of identifying the degree of separation between two users. New search techniques are introduced to provide optimal or near-optimal solutions. The experiments are performed using Twitter, and they show an improvement of several orders of magnitude over greedy approaches. Our optimal algorithm finds an average degree of separation of 3.43 between two random Twitter users, requiring an average of only 67 requests for information over the Internet to Twitter. A near-optimal solution of length 3.88 can be found by making an average of 13.3 requests.

1 Introduction

Social networks are a collection of users that are connected to each other by relations such as "friendship" or "following". Some of these Internet-based networks are special-purpose in nature (e.g., sharing videos using YouTube; micro-blogging with Twitter) while others are general purpose (e.g., Facebook and Myspace). The role of social networks in daily life has been growing at an impressive rate. For example, Facebook has over 500 million user accounts.

An interesting search problem in a social network is to identify the degree of separation ("distance") between two users. Stanley and Milgram, in their pioneering work in the 1960s, concluded that people in the United States are approximately three "steps" (distance) away from each other (Milgram 1967). Today, the claim is that that the average number of steps between any two people on Earth is six (Kleinberg 2000b), and this is commonly referred to as "six degree of separation" or "small world phenomenon". We expect to observe a smaller degree of separation in virtual networks, such as Twitter and Facebook, since a connection between people can be created quickly and easily, as compared to real-world connections.

From the artificial intelligence point of view, a social network of users can be thought of as a *social graph* of nodes. Each node corresponds to a user in the social network, with an edge showing direct connectivity of two users. Thus the degree of separation between two users becomes a search for the minimum cost connection between them. Computing the degree of separation in a social network is an unusual search problem. The tree has a shallow search depth but can have an enormous branching factor (the number of "friends" that each user has). The cost of obtaining the information on the search graph means issuing a query over the Internet (which is many orders of magnitude slower than CPU-based search). An unwelcome complication is that most social networks limit the number of queries that can be issued per hour (to reduce the abuse of resources). For example Twitter only allows sending 150 queries per hour from each unique IP address (Twitter 2011). Finally, the graph doesn't have a well-defined structure, the graph is dynamically changing, and there are no obvious and effective distance heuristics.

Although the degree of separation is most commonly thought of in the social context, being able to efficiently solve this problem has important applications for other network-based applications. For example, in Voice over Internet Protocol (VoIP) networks, when a user calls another user in the network, he/she is first connected to a VoIP carrier (i.e., a main node in the network). The VoIP carrier connects the call to the destination either directly or, more commonly, through another VoIP carrier. The length of the path from the caller to the receiver is important since it affects both the quality and price of the call. The algorithms that are developed in this paper can be used to find a short path (fewest carriers) between the initial (sender) and the goal (receiver) nodes in the network. In the social network context, each edge is assumed to have unit cost. In the VoIP example, the costs need not be uniform: the cost of an edge could be tied, for example, to the speed of the connection.

In this paper we propose new approaches that are able to efficiently search in social networks. Twitter is used as the experimental domain for this research since other social networks are either too small (e.g., FriendFeed) or too restrictive in letting an application access network data (e.g., Facebook). Our solution techniques are general and can be applied to other social networks (or search graphs with similar characteristics). Researchers have shown that many other social networks (e.g., MSN messenger network) and technological networks (e.g., a power grid) have similar character-

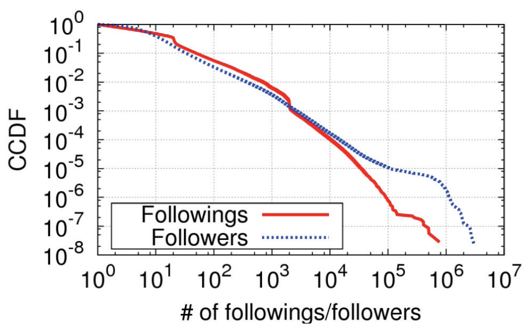

Figure 1: Twitter profile [Kwak *et al.,* 2010].

istics to what we have observed in Twitter: a small diameter and a very large branching factor (Watts and Strogatz 1998; Leskovec and Horvitz 2008).

The baseline for our work is two greedy algorithms that have been widely used for search applications in social networks and also other types of networks (Adamic et al. 2001; Kim et al. 2002; Adamic, Lukose, and Huberman 2003; Watts, Dodds, and Newman ; Kleinberg 2000a; Şimşek and Jensen 2008). We then show that search can be dramatically improved through the use of perimeter (Manzini 1995) and bidirectional (Pohl 1971) search. The bidirectional algorithm yields optimal results (minimum degree of separation). However, for many applications the time taken to achieve optimality is unacceptable. Hence we introduce a version that runs over an order of magnitude faster, but is 13% sub-optimal. The resulting algorithm, bidirectional search with a probabilistic heuristic, can determine the non-optimal degree of separation on Twitter using an average of 13.3 Internet queries.

This paper expands our understanding of algorithms that search Internet graphs as follows:

1. Applying well-known search techniques to find optimal solutions to the degree of separation problem;
2. Introducing a new heuristic approach that finds a near-optimal degree of separation;
3. Performance results that are over an order of magnitude better than previous results.

Section 2 overviews the basics of Twitter and previous work on search in social networks. Section 3 presents algorithms and their performance for finding optimal and near-optimal degrees of separation. Section 5 presents conclusions and future research directions.

2 Background and Related Work

Twitter (twitter.com), the experimental domain used in this paper, is a real-time information network that allows users to send short (140 character) messages ("tweets") to their set of followers. To many, 140 characters does not seem like much information, but around 200 million users, generating 95 million tweets per day, seems to disagree!

Kwak *et al.* collected and analyzed 41.7 million Twitter users' profiles (Kwak et al. 2010). Their results are important to understanding the structure of the social graph and give insights for choosing an appropriate algorithm to search in this network. Figure 1 shows the distribution for the number of *followings* (people whose tweets you receive—solid line) and the number of *followers* (people that receive your tweets—dotted lines). More formally, for a Twitter user s receiving tweets from t, the *number of followers* is defined as $|s|$ that subscribe to t's tweets and the *number of followings* is defined for s as $|t|$. The y-axis shows the complementary cumulative distribution function (CCDF). From the figure we can observe that a small number of users have more than 10,000 followings. Kwak *et al.* showed that among the 41.7 million user profiles that they studied, only 40 had more than one million followers.

Twitter has an Application Programming Interface (API) that allows programmers to query information about the network of users. A single Twitter query can obtain the profile of at most 100 users. A user's profile may include a brief biography, their location, and the number of followers. All the followers and followings of a user can be obtained, but with a maximum of 5,000 per query. A user's tweets can be obtained, to a maximum of 200 per query. Twitter has restricted the number of API requests that an application is allowed to send. Currently the limit for regular users is 150 per hour and for registered Twitter applications is 350 per hour (Twitter 2011). A search algorithm that makes 150 queries might take a few minutes; one that executes 151 will take more than an hour.

Although social networks have became wildly popular, to the best of our knowledge there have been few research projects that study how to efficiently search in such an environment. Most of the research has concentrated on identifying heuristics to be used by a greedy algorithm and studying their performance (Adamic et al. 2001; Kim et al. 2002; Adamic, Lukose, and Huberman 2003; Kleinberg 2000a; Şimşek and Jensen 2008; Adamic and Adar 2005; Liben-Nowell et al. 2005). These papers view search in networks as a message-passing problem, i.e., the algorithm builds the path in a greedy manner. Popular heuristics that are used to select a node for expansion include: 1) maximum out-going degree, 2) similarity to the goal node, 3) selecting a node that is directly connected to the goal (i.e., using breadth-first search up to level one as a heuristic), and 4) geographical information. Watts et al. presented a model for search in networks that have a hierarchical structure (Watts, Dodds, and Newman). All these work have been tested either on the small networks (e.g., email) or on the networks that are constructed by the authors by using a mathematical model (e.g., power-law and scale-free networks).

Kwak et al. created a static snapshot of Twitter, saved it on disk, and then analyzed the resulting 41.7 million user profiles (Kwak et al. 2010). They reported the average shortest path of 4.12 based on analysis of 8,000 pairs of users. Our results give a shortest path of 3.43 based on 1,500 random Twitter pairs. There are two main reasons for this difference. First, the number of Twitter users at 2010 year end is estimated to be more than 200 million, 4.7 times larger than the dataset that was used by Kwak *et al.* As the number of users in the social network increases, the number of connections between users increases, reducing the average length of the shortest path. Second, our test data was sampled from Twitter's June 2010 "public timeline", which is biased towards active Twitter users. Active users tend to have more connections in the network. Having more connections can only help decrease the length of the shortest paths from a user. In contrast, Kwak *et al.* used a uniform sample of users. Our sampling technique doesn't affect our contributions since our experiments were designed to evaluate search algorithm performance; the precise value of the shortest path is of peripheral interest.

3 Search Algorithms

There are two goals for this research, which may conflict:

1. Find the least cost path (degree of separation) between two users, and
2. Find a solution by making the least number of requests to Twitter for information.

The former implies optimality; the latter implies speed. It is important to recognize that sending a request over the Internet is slow—many orders of magnitude slower than the cost of a graph query for a CPU-based search algorithm. Depending on network traffic, it is not uncommon for a single query to take several seconds. Under these circumstances, it is critical to minimize the number of requests for information.

In this section we describe different search techniques that can be used to find a path between two users in a social network (nodes in the search graph). The performance of these algorithms are given in Table 1. Each algorithm is implemented using Twitter's APIs and is run over the Internet. All algorithms are compared using 50 random pairs of Twitter users. These pairs were chosen randomly during Twitter's June 2010 "public timeline". Since Twitter allows us to only send 150 requests per hour, running all these algorithms on a larger dataset is not practical; some algorithms required sending more than 10,000 Twitter requests for each of the problem instances. Time is dominated by the cost of sending/receiving messages over the network; hence, local CPU processing time is essentially irrelevant.

Algorithm performance is compared using several metrics, including the length of the path (degree of separation), the number of generated nodes, and the number of requests for information sent to Twitter. In a social graph, a node is a user. Hence, the number of generated nodes is defined to be the number of users whose Twitter profiles are downloaded by the algorithm. The subset of the Twitter graph used by any of our algorithms to solve a problem instance is small enough that memory size is not a consideration.

Greedy: Maximum Number of Followings

We start with a simple greedy algorithm that at each step chooses a node with the maximum number of followings to expand (i.e., maximum number of users that this node follows). The goal is to find a path between the initial and the goal node. This algorithm and the one in the next subsection (greedy enhanced with geographic information) are used as a baseline for our comparisons.

Starting from the initial node, the algorithm first checks if the goal node appears as one of the neighbors (i.e., is a following). If so, then the algorithm ends. Otherwise, a node is selected that has the maximum number of followings (ties are broken randomly). The algorithm continues this process until it finds the goal or it reaches a node with no following. In the latter case, the algorithm reports failure and exits. The algorithm will report failure if the path length exceeds 325.

A limitation of the greedy algorithm is that it may enter into a loop by visiting a node that has already been visited. For example, 4.9 million Twitter users follow "Barack Obama", and he also follows 717 thousand people. In our experiments we observed that in many cases the greedy algorithm falls into a loop and revisits "Barack Obama" after a few steps. To avoid loops, we record which nodes have been visited in the search (the path from the start node to the current node), and then choose a node with the largest number of followings that have not previously been visited.

Row #1 in Table 1 shows the experimental results for this algorithm. The third column is the number of pairs (out of 50) that are solved by the algorithm. The next column shows the degree of separation (path length) averaged over all the pairs that were solved. The standard deviation of the degree of separation over all the pairs that were solved is shown in column five. The number of generated nodes and the the number of Twitter requests are shown in the next two columns. The last column shows the average time (in minutes) to solve one of the pairs. To measure the time, we sent 600 requests to Twitter and calculated the average time that each request took to complete. It took an average of 1.314 seconds to respond to a request (obtained during non-peak network usage time). There was no significant difference in the time required for different types of requests. The value in the last column is calculated by multiplying the number of Twitter requests (Column 7) by the time that each request needs to be completed (CPU time is negligible). The time that is written in parenthesis also includes the *wall clock time*, i.e. considering the fact the algorithm is able to only send 150 requests/hour.

The greedy algorithm using the maximum number of followings heuristic can only solve 21 of the 50 problems (42%) with an average path length of 92.23. This is achieved by generating 6,103,060 nodes and submitting 61,050 Twitter requests. Taking into account the Twitter limitation of 150 requests per hour, the average greedy solution took over 400 hours ($61,050/150$) of real time!

Greedy: Geographic Heuristic

The simple greedy algorithm can be improved by taking advantage of information in the user's profile. The geographic heuristic selects a node for expansion based on the user's geographic location. The city and the country are extracted from the user's profile. The algorithm first looks at the geographic information of the goal node. In some cases the goal user has not posted any location information in their profile. For these cases the algorithm extracts all the users that the target node is following and chooses as the target location the location that most frequently occurs in this set of users. A database that contains all the cities of different countries is used by our algorithm. If user has posted only the city name in his profile, then our algorithm extracts the country name

Row	Heuristic for tie-breaking	Hits (out of 50)	Degree of Separation	Standard Deviation	Node Generation (average)	Twitter Requests (average)	Time (minutes) (average)
Greedy							
1	Max #followings	21	92.23	103.6	6,103,060	61,050	1,336 (25,756)
2	Geographic	25	100.88	141.3	1,004,773	10,073	220 (4,240)
Greedy: Perimeter Depth 1							
3	Max #followings	50	7.50	13.6	1,254,024	12,551	274 (5,254)
4	Geographic	50	18.94	58.1	1,321,512	13,231	289 (5,569)
Greedy: Perimeter Depth 2							
5	Max #followings	48	3.46	0.7	46,612	1,463	32 (1,382)
6	Geographic	48	3.60	1.0	55,898	1,555	34 (634)
Bidirectional Search: Breadth-first (optimal)							
7	Max #followings	50	3.18	0.7	157	164	3.6 (63)
Bidirectional Search: Probabilistic							
8	Max #followings	50	3.68	0.9	165	9.5	0.2 (0.2)

Table 1: Experimental results on 50 random pairs chosen from Twitter.

from this database. For the cases that multiple countries have the same city name, the algorithm considers all of them.

In the node expansion phase, the greedy algorithm extracts the location of all the neighbors of the current node. If there exists a node with the same city as the target city, then it is selected for expansion, otherwise we check if there exists any node with the same country as that of the target node. If none of the users are from the same city or country, a node with the maximum number of followings is selected. In all cases we break ties by selecting a node that has the maximum number of followings.

Row #2 shows the result of the greedy algorithm using the geographic heuristic. Compared to Row #1, we can see that this version solves a larger number of problems (25 versus 21), but the average length of the degree of separation increases from 92.23 to 100.88. Using the geographic heuristic improves the speed of the algorithm by roughly a factor of six, both in the number of generated nodes and the number of Twitter requests. However, even this algorithm is far too slow.

Greedy: Perimeter Search

Perimeter search is well known in the heuristic search community (Manzini 1995). In this section we propose to enhance greedy search with a perimeter around the goal node. The perimeter is built by running a backward breadth-first search starting from the the goal node. Experimentally, we observed that on average there are 377 nodes in a perimeter of depth one from the goal, and 177,315 nodes in a perimeter of depth two. Perimeters beyond depth two are not considered: the number of nodes in the perimeter grows exponentially.

In the first phase of the algorithm, a perimeter of depth one or two is built around the goal node. In the second phase, a greedy search is started from the initial node. At each step of the search a check is made to see if any of the neighbors of the current node are present in the perimeter. If such a node is found then the search can stop with success. The path from the start node to the current node and the path from the current node to the goal node (which exists in the perimeter) are merged and returned as the final path.

Rows #3 and #4 show the results of greedy search enhanced with a perimeter of depth one. Row #3 shows the result when using the maximum number of followings as the heuristic. Compared to the simple greedy algorithm, the addition of a perimeter has resulted in a dramatic improvement in the search: all the problems are now solved with an average 12-fold improvement in path length and a five-fold reduction in search effort. Row #4 shows the results of using the geographic heuristic. All problems are now solved with an average five-fold improvement in path length. However, surprisingly the search effort is slightly more. Note the real-time cost of running this algorithm is now under 100 hours.

Rows #5 and #6 show the results for both heuristics when the perimeter is of depth two. Surprisingly the number of solved problems reduces from 50 to 48. Two of the problem instances had enormous perimeters that had to be built, resulting in over two million requests being sent to Twitter before being aborted. The solution quality is excellent: 3.46 for the followings heuristic and 3.60 for the geographic heuristic. Clearly, a perimeter of depth two is a huge performance enhancement since, effectively the perimeter represents over half of the average path length. Both node generation (factor of over 25 for followings and over 20 for geographic) and Twitter requests (factor of eight) are greatly reduced. The wall clock time for the solved problems is down to an average of 10 hours.

Although greedy perimeter search has achieved significantly better results comparing to pure greedy methods, the number of requests sent to Twitter is still too high for an online application.

Bidirectional Search: Breadth-first

When the start and the goal nodes are explicitly defined and the search operators are reversible then we can use bidirectional search; these conditions hold for the degree of separation problem. Bidirectional search proceeds simultaneously from both the start and the goal nodes (Pohl 1971). Given that the depth of search for the degree of separation is relatively shallow, the simplest bidirectional search would run breadth-first searches simultaneously from the start and goal nodes. Two open lists can be used, one for each search direction. The algorithm alternates between the open lists at each step. For each node in the selected open list, all of its children are expanded. Each child is checked to see if it exists in the other open list; if so then a solution has been found. A solution, when found, is guaranteed to be optimal.

Bidirectional search is an efficient algorithm for domains where the branching factor is large and the diameter of the graph is small. Twitter is an example of a domain with such properties. Our experimental results support this claim as bidirectional search outperforms all the previous greedy-based algorithms. Row #7 shows that bidirectional search finds an average optimal path length of 3.18. Only 157 nodes are generated using 164 Twitter requests.[1] The number of nodes is reduced by over two orders of magnitude and the number of Twitter requests decreases by roughly one order of magnitude over the best greedy result (Row #5), while producing an optimal result.

Although bidirectional search represents a major performance gain, the real time cost is still unacceptable for an online application. The number of requests for information (164) exceeds Twitter's hourly limit (150), meaning on average a degree of separation calculation requires more than one hour of wall clock time. Another issue is the open list: it can get too big. In effect, bidirectional search is simultaneously building a perimeter for the initial and goal nodes. If a particular problem instance hits a node with a huge branching factor or has a large solution length (say six), memory can become exhausted.

Bidirectional Search: Probabilistic

Requiring an optimal solution implies additional search effort: one needs to find the best solution and guarantee that there is no better. Relaxing optimality can reduce search effort. Bidirectional search, as described above, has an implicit heuristic ordering of its open lists. Because of the breadth-first search, nodes are ordered on their distance to the initial/goal state. Bidirectional search with the probabilistic heuristic injects a greedy aspect to the search. At each step, the algorithm alternates between lists selecting the node with the highest score (greedy) for expansion without enforcing the breadth-first search constraint. By using the distance from the initial/goal state as its heuristic, bidirectional search can guarantee optimality; the probabilistic heuristic does not.

The probabilistic bidirectional search algorithm uses the following node evaluation criteria. For each node in the open list, the distance of the node from the initial/goal node is maintained (breadth-first search information). Consider set D that contains all the distances of the nodes in a given open list ($|D|$ is very small). At each step, our algorithm first chooses one of the $d \in D$ with probability $p(d)$, where $p(d)$ depends on the value of d and $\sum_{d \in D} p(d) = 1$. Intuitively, the smaller the d, the higher the chance it should be selected. The probability of depth d being selected ($p(d)$) is calculated as follows. For each $d \in D$:

$$p(d) = \frac{e^{\alpha d}}{\sum_{i=1}^{D_{max}} e^{\alpha i}} \tag{1}$$

where D_{max} is the maximum depth of the nodes in the open list and α is the decay parameter. The decay parameter determines the magnitude of the probability difference between depths. The probabilistic bidirectional search combines exploitation (greedy) and exploration (breadth-first), with the decay parameter defining this trade-off. The algorithm is biased towards expanding nodes in a breadth-first order, but with a small probability it will make a greedy decision. In our experiments, the decay parameter is set to -2.

Having chosen a depth d, then from all the nodes that have distance d select for expansion a node that has the maximum number of followings. After expanding this node, the algorithm checks if any of its children exists in the other open list. If so, then a solution has been found (possibly non-optimal) and the search terminates.

Row #8 shows the result of bidirectional search using the probabilistic heuristic. Comparing its results to that of bidirectional search using the breadth-first search heuristic (Row #7) shows that the new version found an average path length of 3.68, 15% larger than the optimal path length of 3.18. An average of 9.5 requests were sent to Twitter, an improvement of 17-fold. This version can be considered usable in real time, as the average degree of separation search takes less than 15 seconds.

4 Experiments on Larger Dataset

The previous section showed results for different search approaches using 50 randomly selected pairs of Twitter users. We were unable to test all of our algorithms on a larger dataset since some of the algorithms took many days to run. In this section we present the result of different versions of bidirectional search algorithm on a larger dataset.

Table 2 shows results for using bidirectional search algorithms on 1,500 random pairs chosen from Twitter. The results are obtained by taking the average over the instances that are solved by each algorithm. Row #1 shows that the optimal version of bidirectional search could not solve 20 of 1,500 instances, the consequence of exhausting the number of requests to Twitter. The search is stopped if it sends more than 10,000 requests to Twitter. Row #2 shows that by using the probabilistic heuristic and relaxing optimality, all 1,500 problem instances are solved. The average degree of separation found is 3.88 (13% suboptimal). This is achieved by sending an average of 13.3 requests to Twitter, a 27-fold reduction over the cost of computing the optimal answer. This translates to a wall clock time of less than a minute for the non-optimal answer, to more than two hours for an optimal answer (assuming memory is not exhausted).

Given that the probabilistic version is so fast and the quality of its answers are quite good, this suggests a simple way

[1]It may sound odd to have fewer nodes than Twitter requests. Recall that a node reflects information from Twitter about a user. However, a single user may generate multiple requests. For example, a user may have more than 5,000 followers, exceeding the Twitter maximum per request. Retrieving all of Barak Obama's followers (one user) requires multiple Twitter requests (roughly 1,000).

Row	Heuristic for tie-breaking	Hits (out of 1,500)	Degree of Separation	σ	Node Generation (average)	Twitter Requests (average)	Time (minutes) (average)
Bidirectional Search: Breadth-first (optimal)							
1	Max #followings	1480	3.43	0.68	337	371	8.1 (128)
Bidirectional Search: Probabilistic							
2	Max #followings	1,500	3.880	0.77	204	13.3	0.3 (0.3)
Bounded Bidirectional Search (optimal)							
3	Max #followings	1,500	3.435	0.67	222	67	1.4 (1.4)

Table 2: Experimental results on 1,500 random pairs chosen from Twitter.

to reduce the cost of obtaining an optimal solution. First, run the probabilistic search, obtaining a solution of length K. Second, run the breadth-first bidirectional search to a maximum combined depth of $K - 1$. If this latter search is successful then the optimal solution has been found. If it fails, then the probabilistic search result of K is optimal.

This technique, called Bounded Bidirectional Search, can significantly reduce the number of Twitter requests needed to obtain an optimal solution by avoiding the last (and most expensive) iteration of the breadth-first bidirectional search. Row #3 shows the result of this algorithm. An optimal solution for all 1,500 problem instances is found. Comparing to Row #1, bounded bidirectional search reduces the number of Twitter requests by a factor of 5.5, enabling optimal search with an average time 1.4 minutes (very good, but perhaps still not practical).

5 Conclusions

Social networking is a transformative Internet phenomenon. With networks like Facebook and its more than 500 million users, researchers will want to investigate properties of this evolving web of interconnected users and communities. The degree of separation is an important property of such a network and one that is of great interest, especially to social scientists. Twitter's 3.43 degree of separation is surprisingly small (or, rather, we were surprised) and perhaps is indicative of changing social norms. It would be very interesting to do a more definitive study of this number for Twitter and other social networking sites, and then monitor how this changes over time. It reflects the truism that the world gets smaller every day.

This paper has discussed search algorithms and heuristics for dynamic Internet-based graphs. In this environment, many of the traditional assumptions of heuristic search algorithms are not applicable. Two in particular stand out: nodes are expensive to evaluate and the cost can vary dramatically deepening on conditions out of the application's control (e.g., network load), and there is the potential for an enormous branching factor. Our research represents a step forward in applying traditional search algorithms designed for memory-based search (or even disk-based search (Korf 2004)) to the challenging world of the Internet.

Acknowledgements

Financial support from iCORE and NSERC is greatly appreciated.

References

Adamic, L. A., and Adar, E. 2005. How to search a social network. *Social Networks* 27:2005.

Adamic, L. A.; Lukose, R. M.; Puniyani, A. R.; and Huberman, B. A. 2001. Search in power-law networks. *PHYS.REV.E* 64:046135.

Adamic, L. A.; Lukose, R. M.; and Huberman, B. A. 2003. Local search in unstructured networks. In *Handbook of Graphs and Networks*, 295–317. Wiley-VCH.

Kim, B. J.; Yoon, C. N.; Han, S. K.; and Jeong, H. 2002. Path finding strategies in scale-free networks. *Phys. Rev. E* 65(2):027103.

Kleinberg, J. 2000a. Navigation in a small world. *Nature* 406(6798):845.

Kleinberg, J. 2000b. The small-world phenomenon: an algorithm perspective. In *STOC'00*, 163–170. New York, NY, USA: ACM.

Korf, R. E. 2004. Best-first frontier search with delayed duplicate detection. In *AAAI'04*, 650–657. AAAI Press.

Kwak, H.; Lee, C.; Park, H.; and Moon, S. 2010. What is twitter, a social network or a news media? In *WWW'10*, 591–600. ACM.

Leskovec, J., and Horvitz, E. 2008. Planetary-scale views on a large instant-messaging network. In *WWW'08*, 915–924.

Liben-Nowell, D.; Novak, J.; Kumar, R.; Raghavan, P.; and Tomkins, A. 2005. Geographic routing in social networks. *Proceedings of the National Academy of Sciences* 102(33):11623–11628.

Manzini, G. 1995. Bida: an improved perimeter search algorithm. *Artif. Intell.* 75:347–360.

Milgram, S. 1967. The small world problem. *Psychology Today* 1:61.

Pohl, I. 1971. Bi-directional search. *Machine Intelligence* 6:127–140.

Şimşek, Ö., and Jensen, D. 2008. Navigating networks by using homophily and degree. *Proceedings of the National Academy of Sciences* 105(35):12758–12762.

Twitter. 2011. http://dev.twitter.com/pages/rate-limiting.

Watts, D. J., and Strogatz, S. H. 1998. Collective dynamics of 'small-world' networks. *Nature* 393(6684):440–442.

Watts, D. J.; Dodds, P. S.; and Newman, M. E. J. Identity and search in social networks. *Science*.