Using Ontological Information to Accelerate Path-Finding in Large Semantic Graphs: A Probabilistic Approach

Tina Eliassi-Rad and **Edmond Chow**
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Box 808, L-560, Livermore, CA 94551
{eliassi, echow}@llnl.gov

Abstract

Many real-world graphs contain *semantics*. That is, they represent *meaningful* entities and relationships as vertices and directed edges, respectively. Moreover, such graphs (called *semantic graphs*) have meaningful types associated with their vertices and edges. These types produce an *ontology graph*, which specifies the types of vertices that may be connected via a given edge type. Path-finding in large real-world semantic graphs can be a non-trivial task since such graphs typically exhibit small-world properties. In this paper, we use ontological information, probability theory, and heuristic search algorithms to reduce and prioritize the search space between a source vertex and a destination vertex. Specifically, we introduce two probabilistic heuristics that utilize a semantic graph's ontological information. We embed our heuristics into A* and compare their performances to breadth-first search and A* with a simple non-probabilistic heuristic. We test our heuristics on both unidirectional and bidirectional search algorithms. Our experimental results on two real-world semantic graphs illustrate the merits of our approach.

Keywords: large semantic graphs, ontologies, path finding, probability theory, search, uncertainty

Introduction

Many real-world graphs contain both *semantical* and *topological* information. Semantical information are given as human-understandable attributes on vertices and edges. Topological information capture structure and connectivity. Such real-world graphs are known as *semantic graphs* (or *attributed relational graphs*). Semantic graphs are efficient structures for the detection of relationships within large collections of seemingly disjoint and heterogeneous entities (Coffman, Greenblatt, and Marcus 2004). Figure 1 illustrates a semantic graph corresponding to a small portion of the Internet Movies Database (IMDB) (`http://imdb.org`). Other examples include citation networks of authors, papers, journals, institutions and biological networks of genes, proteins, and molecules. In addition to meaningful attributes, semantic graphs have *type* information on their vertices and edges, which defines permissible relationships among the specified entities (i.e., edge types that may connect two given vertex types). By utilizing such information, we can construct an *ontology graph* (a.k.a., a schema) whose vertices and edges are, respectively,

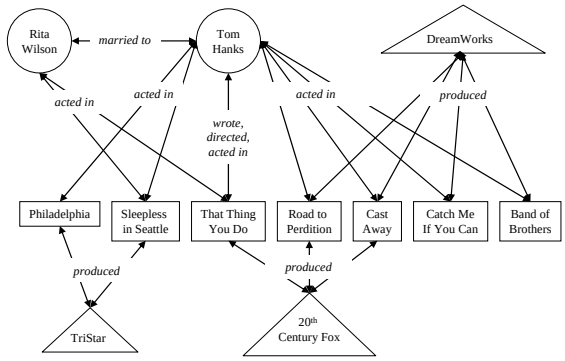

Figure 1: A small portion of the Internet Movies Database (IMDB) Semantic Graph.

the vertex types and edges types of one or more semantic graphs. Since the numbers of vertex types and edge types are small compared to the numbers of vertices and edges, ontology graphs provide efficient structures for representing the information contained in semantic graphs albeit at their type levels. In other words, a semantic graph contains instantiations of the vertex and edge types that are defined in its ontology. Figure 2 shows the ontology for the semantic graph depicted in Figure 1. A task well-suited to semantic graphs is the detection of relationships among a large collection of seemingly disjoint entities. In its simplest form, the task of relationship detection reduces to these two problems:

- $Q_1(G, s, d)$: Find a shortest path (*i.e.*, the fewest possible number of edges) between two vertices, s and d, in the semantic graph G.
- $Q_2(G, s, d)$: Find the subgraph consisting of all the shortest paths between two vertices, s and d, in the semantic graph G.

Depending on the size and topology of a semantic graph, the search space for $Q_1(G, s, d)$ and $Q_2(G, s, d)$ can be quite large. A popular approach for reducing search space (and consequently accelerating search) is to prioritize different path options between a source vertex and a destination vertex. Path-finding algorithms that use this approach are called heuristic (or informed) search algorithms. Such al-

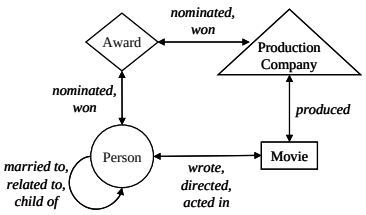

Figure 2: An Ontology for the IMDB Semantic Graph. Vertices and edges in the ontology graph are a superset of the vertex and edge types appearing in the semantic graph (see Fig 1).

gorithms typically use a cost function, whose value depends on a heuristic measure that estimates the cost of reaching the destination from a given vertex.

In order to prioritize different path options for semantic graphs, we use frequency statistics about vertex types and edge types to define a probability model for measuring the uncertainty of an edge's occurrence in the semantic graph. In particular, our heuristics are based on (1) the *likelihood* of a path existing from the current vertex to the destination vertex and (2) the *posterior probability* of a path existing from the current vertex to the destination vertex.

We test our probabilistic heuristics by embedding them into the A* search algorithm and compare their performances with a simple non-probabilistic heuristic. This simple heuristic uses the connectivity information available in the ontology graph to estimate the existence of a path from the current vertex c to the destination vertex d in the semantic graph. We also compare A* with our heuristics to breadth-first search, which is an uninformed search algorithm. We test our heuristics on two real semantic graphs, namely a version of IMDB graph (at `http://kdd.ics.uci.edu/`) and a terrorism data graph (at `http://teknowledge.com`). In addition, we illustrate how these heuristics perform in bidirectional A* and compare the results to bidirectional breadth-first search.

We use the following notation throughout the remainder of this paper:

- V denotes a set of vertices in the semantic graph.
- E denotes a set of edges in the semantic graph.
- $T_V = \{t_1, \ldots, t_n\}$ is a set of n vertex types.
- $T_E \in \{(t_i, t_j)$, where $t_i, t_j \in T_V\}$ is a set of edge types.
- vt denotes a mapping from V to T_V that associates a vertex type with each vertex. If s is a vertex in a semantic graph, vt_s denotes the type for vertex s.
- et denotes a mapping from E to T_E that associates an edge type to each edge. If k is an edge in a semantic graph, et_k denotes the type for edge k.
- s, d, and c, respectively, represent the source, destination, and current vertices. The current vertex refers to the vertex being examined by the search algorithm.

We define a semantic graph as $G = (V, E, vt, et)$ and its ontology graph as $T = (T_V, T_E)$. It is important to note that a semantic graph does not have vertices and edges with types that are not present in its associated ontology graph. In other words, T_V and T_E are, respectively, supersets of the vertex and edge types that occur in the semantic graph G.

Related Work

To our knowledge, no one has developed heuristics for path-finding in semantic graphs. For social networks, Faloutsos *et al* (2004) have developed an algorithm for detecting "connection subgraphs." Their approach regards a social network with weighted undirected edges as an electrical circuit with a network of resistors and a "connection" between two vertices as a path with the most units of electrical current. Their algorithm does not apply to semantic graphs, which are semantically much richer than social networks.

There is a small community machine learning researchers that have used semantic graphs with heterogeneous types of vertices and edges (Getoor 2003; McGovern et al. 2003; Neville, Adler, and Jensen 2003). Their algorithms are typically designed for learning probabilistic models on vertices and/or edges for subsequent inference. For example, McGovern and Jensen (2003) learn models that identify predictive structures in semantic graphs (such as learning relational structures that predict academy award nominees).

Cost and Heuristic Functions

Most heuristic search algorithms contain a variant of the following cost function: $f(s,d) = g(s,c) + h(c,d)$, where $f(s,d)$ is the total cost of exploring a path from the source vertex s to the destination vertex d, $g(s,c)$ is the cost encountered so far, and $h(c,d)$ is the estimated cost to reach the destination vertex d from the current vertex c. The precise definitions of $g(s,c)$ and $h(c,d)$ depend on the notion of what constitutes an "optimal" path for the task at hand.

In this paper, our notion of an optimal path between two vertices is the path with the fewest number of edges (a.k.a. a shortest path). Thus, the cost of a path is the number of edges on that path. In particular, we define the following cost function:

$$f(s,c,d) = g(s,c) + h(vt_c, vt_d) \quad (1)$$

where $g(s,c)$ is the cost of the shortest path from s to c in the semantic graph and $h(vt_c, vt_d)$ is the cost of the shortest path from c's vertex type to d's vertex type in the ontology graph. The function $h(vt_c, vt_d)$ is our simple non-probabilistic heuristic.

Not all heuristic functions are created equal. In particular, heuristic functions that are *admissible* are desirable because they guarantee that a search algorithm will return an optimal solution whenever one exists. Namely, a heuristic function is *admissible* if it never *overestimates* the cost of reaching the destination vertex.

Our standard non-probabilistic heuristic, $h(vt_c, vt_d)$, is admissible. The reasoning is as follows. Since the ontology graph is an abstraction of the semantic graph (with respect to its types), the minimum length of the shortest path between vt_c and vt_d in the ontology graph is always less than

or equal to the minimum length of the shortest path between c and d in the semantic graph. This condition is true for any c and d.

Our probabilistic heuristics use a variant of *dynamic weighting* (Pohl 1973). The main idea behind dynamic weighting is that as the search nears the destination vertex, the heuristic function h should have less of an influence on the value of f. Keeping this in mind, we define the following cost function for path-finding in semantic graphs:

$$f'(s,c,d) = g(s,c) + h'(vt_c, vt_d) = \quad (2)$$

$$g(s,c) + h(vt_c, vt_d) + \left(w(vt_c, vt_d) \times \frac{h(vt_c, vt_d)}{h_{min}(vt_c, vt_d)} \right)$$

where $h_{min}(vt_c, vt_d)$ is the minimum heuristic value to vt_d among all the neighbors of vertex type vt_c and $w(vt_c, vt_d)$ is a weight function measuring the uncertainty of $h(vt_c, vt_d)$. The uncertainty associated with $h(vt_c, vt_d)$ stems from the fact that the ontology graph (on which $h(vt_c, vt_d)$ is computed) is an abstraction of the semantic graph. That is, a permissible path according to the ontology graph may not exist in the semantic graph.

Our two probabilistic approaches define different choices for w. As long as w is between 0 (i.e., no uncertainty) and 1 (i.e., complete uncertainty), our new probabilistic heuristic function $h'(vt_c, vt_d)$ is *1-admissible* (Pearl 1984). In other words, $h'(vt_c, vt_d)$ never overestimates the cost of reaching the destination vertex by more than a factor of two. The proof is as follows. The upper bound for $h'(vt_c, vt_d)$ is $h(vt_c, vt_d) + h(vt_c, vt_d) = 2 \times h(vt_c, vt_d)$, which occurs when $w(vt_c, vt_d)$ is one (i.e., there is complete uncertainty) and $h_{min}(vt_c, vt_d)$ is one (i.e., minimum distance to the destination vertex is determined to be one).[1]

For the dynamic weighting heuristic, we did not choose the following heuristic:

$$h'(vt_c, vt_d) = w(vt_c, vt_d) \times h(vt_c, vt_d) \quad (3)$$

because it is not admissible. Here is an example that illustrates this point. Suppose we have the following:

- Path 1: $w_1(vt_c, vt_d) \times h_1(vt_c, vt_d) = 0.5 \times 5 = 2.5$. So, $f_1(s,c,d) = g(s,c) + 2.5$.
- Path 2: $w_2(vt_c, vt_d) \times h_2(vt_c, vt_d) = 0.2 \times 9 = 1.8$. So, $f_2(s,c,d) = g(s,c) + 1.8$.

Since $g(s,c) \geq 0$ and $f_2(s,c,d) \leq f_1(s,c,d)$, path 2 will be examined before path 1. But, $h_2(vt_c, vt_d) \geq h1(vt_c, vt_d)$, which means that an optimal answer may not be found!

Setting the Uncertainty Weight, *w*

The weight $w(vt_c, vt_d)$ is a normalized range function, which represents the uncertainty of the path from c to d in the semantic graph. In particular, this weight function is defined in terms of minimizing the cost associated with visiting neighbors of c in the semantic graph that either do not lead to d or lead to long path(s). We define this uncertainty cost to be one minus the certainty of an edge type's occurrence. In other words, an edge type with high certainty in the ontology is expected to have a low uncertainty (or cost) in the semantic graph. Specifically, $w(vt_c, vt_d)$ equals:[2]

$$\frac{\max(PathUncertainty) - \min(PathUncertainty)}{\max(PathUncertainty))} \quad (4)$$

The terms $\max(PathUncertainty)$ and $\min(PathUncertainty)$, respectively, refer to the maximum and minimum uncertainty of edges on the path from vt_c to vt_d in the ontology graph.[3]

We measure uncertainty with a probabilistic function that is based on the frequencies of vertex and edge types.[4] In particular, $uncertainty(vt_c, vt_d) = 1 - certainty(vt_c, vt_d)$. To calculate the certainty of a path, we need to define the certainty of an edge. In particular, the function $certainty_{et_k}(vt_i, vt_j)$ measures the *probability of a given vertex, i, of type vt_i having at least one edge of type $et_k = (vt_i, vt_j)$*. We define this probability to be as follows:

$$Pr(et_k|vt_i) = Pr(vt_i - [et_k] \rightarrow vt_j|vt_i) = \quad (5)$$

$$Pr(\exists \textit{ at least one edge } et_k : vt_i - [et_k] \rightarrow vt_j|vt_i) =$$

$$1 - \frac{\binom{(|vt_i|-1)\times|vt_j|}{|et_k|}}{\binom{|vt_i|\times|vt_j|}{|et_k|}} = 1 - \frac{\prod_{l=0}^{|vt_j|-1}(|vt_i||vt_j| - |et_k| - l)}{\prod_{l=0}^{|vt_j|-1}(|vt_i||vt_j| - l)}$$

The number of elements in the sample space is represented by the denominator, which is the total number of all possible graph structures with $|vt_i|$ vertices of type vt_i linked to $|vt_j|$ vertices of type vt_j by $|et_k|$ edges of type et_k. There are two events within the sample space: (*i*) a given vertex vt_i has at least one outgoing edge of type et_k or (*ii*) a given vertex vt_i has no outgoing edge of type et_k. The numerator of Eq. 5 is the number of graphs in which a given vertex of type $|vt_i|$ has at least one outgoing edge of type et_k. Two noteworthy observations about Eq. 5 are (*i*) large values for $|vt_j|$ make the product term smaller since each factor is less than 1; consequently the certainty value increases and (*ii*) additional factors become important when $|et_k|$ is small compared to $|vt_i| \times |vt_j|$.

The function $certainty_{et_k}(vt_i, vt_j)$ assumes that there are no redundant edges in the semantic graph. That is, we depend on the following axiom:

Axiom 1: Let there be no redundant edges in the semantic graph. Then, for any edge type $et_k = (vt_i, vt_j) \in T_E$ and $vt_i, vt_j \in T_V$, $\mid et_k \mid \leq \mid vt_i \mid \times \mid vt_j \mid$, where $\mid vt_i \mid$, $\mid vt_j \mid$, and $\mid et_k \mid$ are the number of times vt_i, vt_j, and et_k occur in the semantic graph, respectively.

Axiom 1 states that the number of occurrences of an edge type in a semantic graph cannot be greater than the product of the number of occurrences of the vertex types to which it

[1] The lower bound for $h'(vt_c, vt_d)$ is $h(vt_c, vt_d)$, which occurs when $w(vt_c, vt_d)$ is zero, *i.e.*, there is no uncertainty.

[2] If $\max(PathUncertainty)$ is zero, $w(vt_c, vt_d)$ is set to one and not infinity.

[3] For paths from vt_c to vt_d, $\max(PathUncertainty) = \max(\{\forall(i,j) \in path(vt_c, vt_d) : (1 - certainty(vt_i, vt_j))\})$ and $\min(PathUncertainty) = \min(\{\forall(i,j) \in path(vt_c, vt_d) : (1 - certainty(vt_i, vt_j))\})$.

[4] The statistics relating to the number of occurrences of each vertex and edge types are collected when the semantic graph is being constructed.

connects. Figure 3 illustrates this point. If a semantic graph has three instances of the vertex type vt_i (namely, A, B, and C) and two instances of the vertex type vt_j (namely, D and E), then their connecting edge type et_k cannot occur more than six times since redundant edges are prohibited (*e.g.*, two instances of edge type et_k cannot connect A to D).

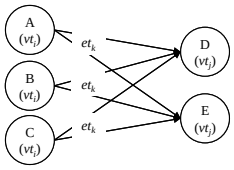

Figure 3: Portion of a semantic graph, which illustrates Axiom 1. Since no redundant edges are allowed, there cannot be more than six instances of et_k between the three instances of vt_i and the two instances of vt_j.

Axiom 1 provides us with a lower-bound for our edge certainty measure as defined in Eq. 5. In particular, the certainty of an edge of type et_k existing in a semantic graph is never less than

$$\frac{\mid et_k \mid}{\mid vt_i \mid \times \mid vt_j \mid} \tag{6}$$

The certainty measure (as defined in Eq. 5) and its estimate (as defined in Eq. 6) allow us to make conclusions about the semantic graph without having to traverse the large graph itself. For instance, here are two simple and extreme scenarios. Suppose we have an ontology graph T containing vertices vt_i and vt_j that are connected via an edge $et_k = (vt_i, vt_j)$. In the first scenario, the frequency statistics indicate that the semantic graph G contains only one instance of et_k, vt_i, and vt_j each. Therefore, we can infer with 100% certainty that a path of length one exists in G from the only vertex of type vt_i to the only vertex of type vt_j via the edge of type et_k. In the second scenario, the frequency statistics indicate that no instance of et_k exists in the semantic graph G even though multiple instances of vt_i and vt_j are present in G. Here we can conclude that a path of length one does not exist in G from a vertex of type vt_i to a vertex of type vt_j.

With Axiom 1 and Eq. 5 in mind, we present our two probabilistic certainty functions in the next subsections.

Certainty as Likelihood Our likelihood certainty measure, $certainty_L(vt_c, vt_d)$, estimates the likelihood of each edge on a path from vt_c to vt_d and assigns the maximum (edge) likelihood as the certainty value for the path. Using Eq. 5, we compute $certainty_L(vt_c, vt_d)$ according to the following rules. If $c \equiv d$, then $certainty_L(vt_c, vt_d)$=1. Otherwise, $certainty_L(vt_c, vt_d) = \max(\{\forall vt_{c'} \in Neighbors(vt_c) : (1 - Pr(vt_c \rightarrow vt_{c'} | vt_c)), certainty_L(vt_{c'}, vt_d)\})$.

The function $certainty_L(vt_c, vt_d)$ is biased towards edge types with high certainties because such edge types are expected to map back to edges with low costs (or uncertainties) in the semantic graph (recall that $uncertainty = 1 - certainty$). In particular, among the neighbors of vt_c (in the ontology), we choose the neighbor with the highest certainty of having an edge in the semantic graph.

Certainty as Posterior Probability Our posterior probability certainty measure, $certainty_P(vt_c, vt_d)$, computes the posterior probability of each edge on a path from vt_c to vt_d and assigns the maximum (edge) posterior probability as the certainty value for the path. We use Bayes' Theorem and Eq. 5 to calculate $certainty_P(vt_c, vt_d)$ according to the following rules. If $c \equiv d$, then $certainty_P(vt_c, vt_d)$=1. Otherwise, $certainty_P(vt_c, vt_d) = \max(\{\forall vt_{c'} \in Neighbors(vt_c) : (\frac{likelihood(vt_c, vt_{c'}) \times prior(vt_c, vt_{c'})}{marginal(vt_{c'})}), certainty_P(vt_{c'}, vt_d)\})$. We compute the likelihood of an edge type $(vt_c, vt_{c'})$ occurring at least once in the semantic graph with $likelihood(vt_c, vt_{c'}) = 1 - Pr(vt_c \rightarrow vt_{c'} | vt_c)$. To calculate the unconditional prior probability of an edge existing between vt_c and $vt_{c'}$, we use $prior(vt_c, vt_{c'}) = \frac{|(vt_c, vt_{c'})|}{\sum_{i=1}^{K} |(vt_c, vt_{c'})|}$, where $K = \mid Neighbors(vt_c) \mid$. The marginal normalization computes the unconditional probability of a vertex of type $vt_{c'}$ occurring in the semantic graph with $marginal(vt_{c'}) = \frac{|vt_{c'}|}{\sum_{i=1}^{K} vt_i}$, where $K = \mid Neighbors(vt_c) \mid$.

The function $certainty_P(vt_c, vt_d)$ is biased towards edge types with high posterior probability because such edge types are expected to map back to edges with low costs in the semantic graph. Specifically, among the neighbors of vt_c (in the ontology), we select the one with the highest posterior probability of having an edge in the semantic graph.

Section Summary

To recap our two probabilistic heuristics utilize the following heuristic function:

$$h'(vt_c, vt_d) = h(vt_c, vt_d) + w(vt_c, vt_d) \times \frac{h(vt_c, vt_d)}{h_{min}(vt_c, vt_d)}$$

The weight $w(vt_c, vt_d)$ measures the uncertainty of an edge type's occurrence in the semantic graph with the following formula:

$$\frac{\max(1 - certainty(vt_c, vt_d)) - \min(1 - certainty(vt_c, vt_d))}{\max(1 - certainty(vt_c, vt_d))}$$

In the Experiments Section, we embed our heuristics into unidirectional and bidirectional A* (Pearl 1984) to measure their performances.

Experiments

In this section, we embed our probabilistic heuristics and the simple non-probabilistic heuristic into A* and compare their performances to breadth-first search on two large real-world semantic graphs and their ontologies. Our experiments include both unidirectional and bidirectional searches. We utilize standard implementations of A* (with a heap) and breadth-first search. In the bidirectional searches, we expand the most promising direction first. Specifically, the order of expansion is forward search followed by backward search if $g(s, c)$ in the forward direction is less than $g(d, c)$ in the backward direction.

Performance Metrics

Our performance metrics are (*i*) the number of vertices visited, (*ii*) the work factor, (*iii*) the stretch factor, and (*iv*) the relative branching factor. The work factor is the ratio of the number of vertices visited with heuristic h versus the number of vertices visited with breadth-first search. Conceptually, the work factor is an estimate of how much faster the search algorithm with our heuristic is compared to the breadth-first search algorithm. On the other hand, the stretch factor is a quality metric and is the ratio of the path length of the subgraph returned by heuristic h versus the path length of the subgraph returned by breadth-first search.[5] The relative branching factor is is another measure of how much work an algorithm is performing and equals the ratio of the effective branching factor with heuristic h versus the effective branching factor with breadth-first search.

We will use the following notation in reporting our results:

- C_{T_E}: edge connectivity[6] in an ontology T.
- C_{G_E}: edge connectivity in a semantic graph G.
- *BST*: # of vertices visited by breadth-first search.
- *STD*: # of vertices visited by A* search with the standard cost function (*i.e.*, the heuristic function is not weighted).
- *LKL*: # of vertices visited by A* search with our likelihood-based heuristic.
- *PST*: # of vertices visited by A* search with our posterior-probability-based heuristic.
- *N*: # of vertices visited, *WF*: work factor, *SF*: stretch factor, and *BF*: branching factor.

Description of Real Data

We use two real-world semantic graphs to test our heuristics. The first semantic graph is a terrorism graph generated from data available at the Anti-Defamation League Web site. The ontology for this graph was generated from the information available at `http://www.teknowledge.com`. The second semantic graph is a movie graph generated from the data available at the IMDB Web site and listed in the UCI KDD Repository (`http://kdd.ics.uci.edu/`). See Barthélemy, Chow, and Eliassi-Rad (2005) for details on the ontology for the movies graph. Table 1 summarizes the information in these two semantic graph and their corresponding ontology.

Results on Real-World Semantic Graphs

For our experiments, we randomly select 100 pairs of source and destination vertices and average the results. Since the relative branching factor depends on path length, we report results on the most common path length for each graph. Tables 2 and 3, respectively, show the results for Q_1 (shortest path) and Q_2 (shortest subgraph) with unidirectional search in the terrorism and movies semantic graphs.

[5]The path length returned by breadth-first search is the ground truth when the optimal path is defined to be shortest path.

[6]The edge connectivity on a graph $G(V, E)$ is defined as $\frac{|E|}{|E'|}$, where $\mid E \mid$ is the number of edges in G and $\mid E' \mid$ is the number of edges in a fully connected graph with $\mid V \mid$ vertices (*i.e.*, $\mid E' \mid = \mid V \mid \times \mid V \mid$).

Table 1: Information on Two Real-World Semantic Graphs and Their Ontologies (TRSM: Terrorism Data and IMDB: Movies Data

Data	$\mid V \mid$	$\mid E \mid$	$\mid T_V \mid$	$\mid T_E \mid$	C_{G_E}	C_{T_E}
TRSM	2436	25234	59	522	0.0043	0.15
IMDB	42026	528756	8	30	0.0003	0.47

Table 2: Average Performance Over 100 Runs for Finding the Shortest Path with Unidirectional Search

Data	BFS	STD/BFS	LKL/BFS	PST/BFS
TRSM	N=1902	WF=0.781	WF=0.697	WF=0.666
		SF=1.000	SF=1.008	SF=1.004
	Path Length=3	BF=0.901	BF=0.816	BF=0.777
IMDB	N=32918	WF=0.769	WF=0.720	WF=0.669
		SF=1.000	SF=1.017	SF=1.028
	Path Length=4	BF=0.908	BF=0.838	BF=0.796

Table 3: Average Performance Over 100 Runs for Finding the Shortest Subgraph with Unidirectional Search

Data	BFS	STD/BFS	LKL/BFS	PST/BFS
TRSM	N=2240	WF=1.000	WF=1.000	WF=1.000
		SF=1.000	SF=1.000	SF=1.000
	Path Length=3	BF=1.000	BF=1.000	BF=1.000
IMDB	N=36724	WF=1.000	WF=1.000	WF=0.998
		SF=1.000	SF=1.005	SF=1.002
	Path Length=4	BF=1.000	BF=0.997	BF=0.998

Tables 4 and 5, respectively, depict the results for Q_1 (shortest path) and Q_2 (shortest subgraph) with bidirectional search in the terrorism and movies semantic graphs.

Table 4: Average Performance Over 100 Runs for Finding the Shortest Path with bidirectional Search

Data	BFS	STD/BFS	LKL/BFS	PST/BFS
TRSM	N=1777	WF=0.765	WF=0.472	WF=0.531
		SF=1.000	SF=1.003	SF=1.004
	Path Length=3	BF=0.924	BF=0.567	BF=0.625
IMDB	N=31585	WF=0.537	WF=0.376	WF=0.375
		SF=1.000	SF=1.000	SF=1.000
	Path Length=4	BF=0.790	BF=0.615	BF=0.686

Table 5: Average Performance Over 100 Runs for Finding the Shortest Subgraph with bidirectional Search

Data	BFS	STD/BFS	LKL/BFS	PST/BFS
TRSM	N=2188	WF=0.980	WF=0.965	WF=0.953
		SF=1.000	SF=1.000	SF=1.000
	Path Length=3	BF=0.983	BF=0.984	BF=0.981
IMDB	N=36078	WF=0.974	WF=0.674	WF=0.743
		SF=1.000	SF=1.000	SF=1.000
	Path Length=4	BF=0.978	BF=0.845	BF=0.898

Discussions

Here are some notable observations about our experiments:

- The performances of our heuristics depend on the properties of the ontology. For example, if the ontology is a fully connected graph (i.e., a clique) then no additional information is provided by the heuristic term of the simple A* cost function. The same issue applies to our probabilistic heuristics when the ontology is a clique and all vertex types and all edge types appear with the same frequencies in the semantic graph. On the other hand, the heuristic term of a cost function becomes much more useful when the ontology has a large average path length. For instance, our heuristics perform better on the movies domain than on the terrorism data since the average path length in the movies ontology is larger than that of the terrorism domain. In particular, the average path length for the movies ontology is 1.5 and for the terrorism ontology is 0.964. Very small average path lengths in the ontology graph (especially values less than 1 which indicate disconnected components in the graph) reduce the effectiveness of our heuristics for finding shortest subgraphs more than for finding shortest paths. This is not surprising since the former task is harder than the latter.
- Bidirectional search consistently outperforms unidirectional search on both domains because of the small average path lengths in semantic graphs (which leads to finding an intersection between the forward and backward frontiers quicker than finding the destination vertex). For example, in the terrorism domain the average path length of the semantic graph is 2.837. In the movies domain the average path length of the semantic graph is 3.385.
- When finding single shortest paths with either unidirectional or bidirectional search algorithms, A* with our probabilistic heuristics outperforms (in work factor and relative branching factor) both A* with the simple non-probabilistic heuristic and breadth-first search. The posterior probability-based heuristic performs slightly better then the likelihood-based heuristic since it takes into account the current vertex's local neighborhood information when making choices about vertex exploration.
- For our probabilistic heuristics in A*, all stretch factors satisfy the 1-admissibility bound. Specifically in our experiments, whenever the stretch factor was not 1, A* with the probabilistic heuristic missed the shortest path length by one edge. However, in these cases, the work and branching factors of the heuristic search were considerably lower than those of breadth-first search or A* with the simple non-probabilistic heuristic. The stretch factors were closer to 1 with bidirectional search than with unidirectional search. This is not surprising since semantic graphs typically have small average path lengths.

Conclusion

Many real-world graphs fall under the category of semantic graphs. Such graphs encode human-understandable entities and relationships. As the sizes of such real-world graphs increase, the need for fast path-finding algorithms grows. In this paper, we present and examine two probabilistic heuristics for searching large real-world semantic graphs to retrieve shortest path or subgraphs between two given vertices. In particular, we utilize both the statistical and the connectivity information encoded in the ontology graph to calculate the probability of a path's existence in the semantic graph. We embed our probabilistic heuristics into A* and compare their performances to breadth-first search and A* search with a non-probabilistic heuristic. Our heuristics are based on simple probability calculations and can be efficiently computed and inserted into any informed search algorithm. Our experimental results with both unidirectional and bidirectional search algorithms demonstrate the advantages of our approach on real-world semantic graphs.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.1 UCRL-CONF-202002.

References

Barthélemy, M.; Chow, E.; and Eliassi-Rad, T. 2005. Knowledge representation issues in semantic graphs for relationship detection. In *Papers from the 2005 AAAI Spring Symposium on AI Technologies for Homeland Security*.

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

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

Getoor, L. 2003. Link mining: A new data mining challenge. *SIGKDD Explorations* 5(1):84–89.

McGovern, A., and Jensen, D. 2003. Identifying predictive structures in relational data using multiple instance learning. In *Proc. of the 20th Int'l Conf. on Machine Learning*, 528–535. Washington, DC, USA: AAAI Press.

McGovern, A.; Friedland, L.; Hay, M.; Gallagher, B.; Fast, A.; Neville, J.; and Jensen, D. 2003. Exploiting relational structure to understand publication patterns in high-energy physics. *SIGKDD Explorations* 5(2):165–173.

Neville, J.; Adler, M.; and Jensen, D. 2003. Clustering relational data using attribute and link information. In *Proc. of the Text Mining and Link Analysis Workshop, 18th Int'l Joint Conf. on Artificial Intelligence*.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, Massachusetts: Addison-Wesley.

Pohl, I. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proc. of the 3rd Int'l Joint Conf. on Artificial Intelligence*, 20–23.