Searching for a k-Clique in Unknown Graphs

(Extended Abstract)

Roni Stern
Ben-Gurion University
Beer-Sheva, 85104, Israel
roni.stern@gmail.com

Meir Kalech
Ben-Gurion University
Beer-Sheva, 85104, Israel
kalech@bgu.ac.il

Ariel Felner
Ben-Gurion University
Beer-Sheva, 85104, Israel
felner@bgu.ac.il

ABSTRACT

Agents that solve problems in *unknown graphs* are usually required to iteratively explore parts of the graph. In this paper we research the problem of finding a k-clique in an *unknown graph* while minimizing the number of required exploration actions. Two novel heuristics ($KnownDegree$ and $Clique^*$) are proposed to reduce the required exploration cost by carefully choosing which part of the environment to explore. We further investigate the problem by adding probabilistic knowledge of the graph and propose an MDP and a Monte Carlo based heuristic ($RClique^*$) that uses knowledge of edges probabilities to reduce the required exploration cost. The efficiency of the proposed approaches is demonstrated on simulated random and scale free graphs.

1. INTRODUCTION

Most classic algorithms that solve graph problems assume that the entire structure of the graph is given as input in data structures (e.g. adjacency list or adjacency matrix). We refer to such problems as problems on **known graphs**. By contrast, there are real-life problems which can be modeled as problems on graphs where the graph is not given as input and is not necessarily known in advance. For example, a robot navigating in an unknown terrain where vertices and edges are physical locations and roads respectively. Another example is an agent searching the World Wide Web, where the vertices are the web sites and the edges represent hypertext links. A possible application is a web crawler searching an online database such as CiteSeer or Google Scholar (many of which can not be completely downloaded). We refer to such problems as problems on **unknown graphs**.

Solving problems in such **unknown graphs** requires exploring some part of the graph. We define an exploration action, that when applied to a vertex, discovers all its neighboring vertices and edges. In the web graph domain, for example, the exploration corresponds to sending an HTTP request, retrieving an HTML page and parsing all the hypertext links in it. The hypertext links are the outgoing edges, and the connected web sites are the neighboring vertices. For a physical domain, the exploration is applied by using sensors at a location to discover the near area, e.g., the outgoing edges and the neighboring vertices on the map.

Cite as: Searching for a k-Clique in Unknown Graphs (Extended Abstract), Roni Stern, Meir Kalech and Ariel Felner, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

Such exploration actions are associated with a cost. The exploration cost often requires a different resource than the traditional computational effort. For the web graph scenario, the exploration cost includes network I/O of sending and receiving IP packets. For physical world environment, exploration cost is associated with the cost of activating sensors at a vertex. In such cases, an important task will be to solve the problem while minimizing the exploration cost. Computational CPU cost can be therefore omitted as long as it is running a polynomial algorithm.

We focus on the problem of searching for a k-clique in unknown graphs, starting from a single known vertex. A k-*clique* is a set of k vertices that are pairwise connected. The task is to find a k-clique with minimum exploration cost. Beyond the value of the investigation of such basic problem in unknown graphs, k-clique can be practical in real-world unknown graph domains. For example, finding a set of physical locations forming a clique suggests the existence of a metropolitan. Another example is an agent searching for a set of web sites forming a clique. A clique of web sites suggests resemblance in content. Consider, for example, a searching process of an online academic database such as Google Scholar and CiteSeer. Each paper is referenced by a hypertext link, and citations of a paper are also available as links. Assume a clique of papers is found, i.e. a group of papers where each paper cites or is cited by all the other papers in the group. It is very likely that all papers in the group discuss the same subject. Thus finding a clique in such a web site allows a focused content search.[1]

2. BEST-FIRST SEARCH APPROACH

Searching for a k-clique in unknown graphs is inherently an iterative process, in which the graph is explored vertex by vertex. Therefore, we propose a best-first search approach. At the beginning of every iteration a simple search is performed checking if there is a k-clique in the known part of the graph. Note that since this step operates on the known part of the graph, it requires no exploration cost. If a k-clique has not been found yet and there are vertices that have not been explored yet, the agent chooses which vertex to explore next. This is equivalent to a best-first search where applying an exploration action to a vertex corresponds to **expanding** that vertex and **generating** its neighbors.

In the worst case, all the vertices must be explored. However we propose several intelligent heuristics that significantly reduce the exploration cost in practice. The first

[1]This structural approach can of course be complemented by text data mining approach to prune irrelevant papers.

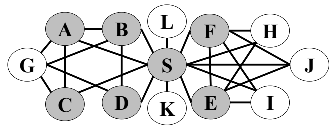

Figure 1: Example of the different heuristics.

heuristic, which we call *Known Degree*, is inspired by a common heuristic used for k-clique problem in known graphs, of searching first vertices with high degree [2]. Since the real degree of a vertex that has not been explored yet is not known, we consider its *known degree* - the number of **expanded** vertices that are adjacent to it. The vertex with the highest *known degree* is selected to be explored.

The second heuristic we developed relates to the size of the *potential k-clique* that a vertex is connected to. A set of vertices is a potential k-clique if they have already been explored and they may become a part of a k-clique. This occurs only when a set of explored vertices is a clique itself (albeit smaller than k), and if there exist enough unexplored vertices that are neighbors of all the vertices in that set. The second heuristic we proposed chooses to explore the vertices that are connected to the largest potential k-clique. An important property of this heuristic is that k minus the size of the largest potential k-clique is a tight lower bound to the cost of exploring a k-clique. In other words, **no admissible heuristic can be better**.

Figure 1 presents an example of both heuristics. The size of the desired clique is 6. Gray and white nodes mark expanded and generated nodes respectively. The next node chosen by *KnownDegree* will be G as its known degree is 4. On the other hand, $Clique^*$ will choose either H,I or J, as they are connected to a potential k-clique of size three (nodes S,E and F). Note that (S, E, F) is a potential 6-clique because nodes H,J and I may become connected when they will be explored.

In many domains, although the exact searched graph is unknown, there is some knowledge on the probability of the edges. For example, if the searched graph is the World Wide Web then it is well-known that it behaves like a scale free graph in which the degree distribution of the graph vertices follows a power law (i.e. the probability of a vertex having a degree x is x^{β} for some exponent β) [1]. Another example that is common in robotics is a navigator robot with imperfect vision of the environment. This can be represented as an error probability on the existence of an edge in the graph of the environment.

By adding probabilistic knowledge, the problem of searching for a k-clique in an unknown graph can be modeled as a finite-horizon MDP. The *states* are all possible exploration outcomes and the *actions* are the exploration actions applied to nodes in V_{gen}. Unfortunately, the size of the MDP state space grows double exponential with the number of vertices in the graph, prohibiting solvers that requires enumerating all of the states such as Value or Policy Iteration.

We propose $RClique^*$, a Monte-Carlo based sampling heuristic that combines $Clique^*$ and probabilistic knowledge of the edges in the graph. $RClique^*$ approximates the expected exploration cost by sampling states of the MDP state space mentioned above. This is done by using $Clique^*$ (with random tie braking) to choose which vertex to explore and simulate the exploration outcome using the probabilistic knowledge available. This simulated exploration is repeated until either a k-clique is found or a maximum exploration depth is reached (the maximum depth is a predefined parameter to the search). This whole process is repeated a predefined number of times. The average exploration cost of finding a k-clique is then used as a heuristic. It is important to note that during the simulated exploration no real exploration is preformed, thus it does not incur any exploration cost.

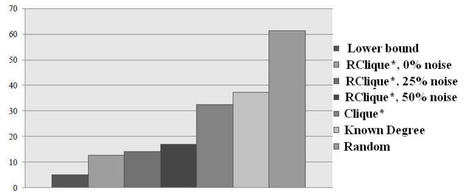

Figure 2: Random graphs, various levels of noise.

3. EXPERIMENTAL RESULTS

We experimented with simple random graphs and scale free graphs. Probabilistic knowledge was simulated by defining a prior probability for every possible edge between generated nodes. We defined a variable *noise* that determines the **maximum** uncertainty of a single edge. For every possible edge e the agent is given $P(e \in E)$=1-$rand(0, noise)$ if $e \in E$ or $P(e \in E) = rand(0, noise)$ otherwise, where $rand(a, b)$ is a random number uniformly distributed between a and b. Clearly if $noise = 1$ the agent has random knowledge about the existence of edges in G.

We run experiments on 100 random graphs with 100 vertices and 800 edges. For $RClique^*$ we used 250 samples and a depth of 3 with various levels of noise. Figure 2 shows the average exploration cost of searching for a 5-clique. The y-axis shows the average exploration cost. The bars denotes the different heuristics. *Random* is a baseline heuristic where vertices are explored randomly. *LowerBound* is the shortest path from the root to the closest k-clique in the graph plus $k - 1$. No algorithm can do better than that.

Interestingly, even with a setting of 50% noise $RClique^*$ outoperfroms $Clique^*$ by a factor of almost 2. We have also evaluated the effect of different sampling depths on a $RClique^*$ heuristic using 250 samples and 50% noise. Depth 1,2,3,4 and 5 yielded an exploration cost that was 5.34, 4.15, 2.65, 2.52, 2.32 times the lowerbound respectively. Thus it seems that while deeper sampling reduces the exploration cost of the search, there is no substantial reduction beyond depth 3. Similar results were obtained for scale-free graphs except for two phenomenons. No significant difference between $Clique^*$ and *KnownDegree* was found. Additionally, the reduction in exploration cost of the the non-probabilistic heuristics over *Random* grew to more than 80%. This is because scale-free graph usually contain few "hub" nodes that are connected to many other nodes, yielding better results for *KnownDegree* and worse results for *Random*.

4. REFERENCES

[1] A. Bonato and W. Laurier. A survey of models of the web graph. In *CAAN*, pages 159–172, 2004.

[2] E. Tomita and T. Kameda. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. of Global Optimization*, 37(1):95–111, 2007.