BIOINFORMATICS APPLICATIONS NOTE Vol. 26 no. 21 2010, pages 2788–2789
doi:10.1093/bioinformatics/btq514

Gene expression

Advance Access publication September 9, 2010

ODES: an overlapping dense sub-graph algorithm

James Long[1,*] and Chris Hartman[2]

[1]International Arctic Research Center and [2]Department of Computer Science, University of Alaska Fairbanks, Fairbanks, AK 99775, USA

Associate Editor: Trey Ideker

ABSTRACT

Summary: Enumeration of the dense sub-graphs of a graph is of interest in community discovery and membership problems, including dense sub-graphs that overlap each other. Described herein is ODES (Overlapping DEnse Sub-graphs), pthreads parallelized software to extract all overlapping maximal sub-graphs whose densities are greater than or equal to a specified cutoff density of at least 1/2.

Availability and Implementation: http://dense.sf.net

Contact: jlong@alaska.edu

Received on July 6, 2010; revised on August 23, 2010; accepted on August 31, 2010

1 INTRODUCTION

Algorithms that infer community membership are important in a number of network problems (Adamcsek *et al.*, 2006; Hu *et al.*, 2005; Palla *et al.*, 2005). In an algorithm like CODENSE (Hu *et al.*, 2005), mRNA interaction maps are generated from microarray data and represented as graphs, where vertices are mRNAs and edges are expression correlations. One is interested in enumerating those groups of mRNAs that have high-correlated expression over different conditions, suggesting that proteins produced from such mRNAs interact with one another in biological modules. Modules have high connectivity between members, and thus would form dense sub-graphs within a map. mRNAs belonging to more than one module lie in the intersection of dense sub-graphs, and algorithms that delineate the modules must be able to distinguish overlapping dense sub-graphs.

We describe herein an exact algorithm to find, within some large graph, overlapping maximal dense sub-graphs whose densities are greater than or equal to a specified cutoff density of at least 1/2, where 'density' of a graph with e edges and n vertices is defined as the number of edges divided by the total number of possible edges, i.e. $2e/n(n-1)$, and where 'maximal' denotes that the inclusion of an additional vertex would lower the density below the cutoff. The algorithm constrains the brute force search-space, yielding acceptable performance on dense subgraphs of 20 vertices.

2 METHODS

2.1 Theory

We prove a theorem that any connected graph with three or more vertices and density at least 1/2 contains a vertex whose removal neither disconnects the graph, nor diminishes its density. Since the resulting graph will be connected and have density at least as large, it qualifies for inductive rounds of vertex removal consistent with the theorem, which may continue until there is a single-edge graph remaining, which has density 1. The algorithm to discover dense sub-graphs in some larger graph is based upon reversing this procedure to build up the dense subgraphs from single-edge sub-graphs of the larger graph.

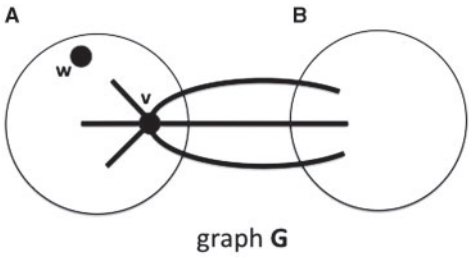

Fig. 1. Cut vertex **v**.

THEOREM A connected graph G, with $\text{den}(G) \geq 1/2$, and $n \geq 3$, has at least one non-cut vertex w, with degree $\text{d}(w) \leq d_{\text{ave}}$, the average degree. Removal of w from G does not decrease the density of G.

PROOF. Given a connected graph G, with e edges, n vertices and $\text{den}(G) \geq 1/2$, pick some vertex v where $\text{d}(v) \leq d_{\text{ave}}$. If v is not a cut vertex, then we are done, and $w = v$. If v is a cut vertex, then let S be the smallest component of $G - v$, and let $A = S + v$ and $B = G \backslash A$. Setting $x = |V(S)|$ and $y = |V(B)|$, we have $x \leq y$, $n = x + y + 1$, and $|V(A)| = x + 1$ (Fig. 1).

It is well known that there exist at least two non-cut vertices in any connected graph with more than one vertex, and thus there must be at least one such vertex w in A that is different from v. Since w is not a cut vertex in A, it is not a cut vertex in G. To show that $\text{d}(w) \leq d_{\text{ave}} = 2e/n$, we note that $\text{den}(G) \geq 1/2 \Rightarrow 2e/n(n-1) \geq 1/2$, or $d_{\text{ave}} \geq (n-1)/2 = (x+y)/2$.

Now

$$\text{d}(w) \leq |V(A)| - 1 = x = \frac{x+x}{2} \leq \frac{x+y}{2} \leq d_{\text{ave}}.$$

Removal of w from G does not decrease the density of G:

$$\text{d}(w) \leq \tfrac{2e}{n} \Rightarrow nd(w) \leq 2e \Rightarrow nd(w) - ne \leq 2e - ne$$
$$\Rightarrow n(e - \text{d}(w)) \geq e(n-2) \Rightarrow \tfrac{(e-\text{d}(w))}{(n-2)} \geq \tfrac{e}{n}$$
$$\Rightarrow \tfrac{2(e-\text{d}(w))}{(n-1)(n-2)} \geq \tfrac{2e}{n(n-1)} \Rightarrow \text{den}(G-w) \geq \text{den}(G).$$

Q.E.D.

2.2 Algorithm

Within a large graph G, any maximal sub-graph with density above a specified minimum $D \geq 1/2$ may be found in the following manner.

Initialize a list S with all single-edge sub-graphs S_i of G. Then, for each S_i in S, successively examine every vertex v adjacent to S_i whose degree in $S_i + v$ is less than or equal to the average degree of $S_i + v$, and whose addition to S_i maintains the density of $S_i + v$ above D, i.e. search for every v that satisfy $\text{den}(S_i) \geq \text{den}(S_i + v) \geq D \geq 1/2$.

For each v that qualifies, insert $S_i + v$ into a sorted list L, provided that $S_i + v$ is not already in L, determined by a binary search. When no other

*To whom correspondence should be addressed.

entries can be added to L, the contents of L become the new S, and another iteration with an initially empty L commences. If no entries have been made into L, the process terminates.

The list of dense sub-graphs to return is built by appending any S_i that is not extended within an iteration, assuming its order (number of vertices) exceeds some specified minimum.

Implementing the algorithm in the above manner, however, has the drawback that the current density of each S_i must be tracked in order to ensure that $\text{den}(S_i) \geq \text{den}(S_i + v)$. But any $S_i + v$ that increases the density of S_i, and is not already in L, is really a pre-computation of some other sub-graph S_k with a different adjacent vertex w added consistent with the theorem, i.e. $S_i + v$ can be reordered as $S_k + w$, $v \neq w$, by our theorem, and may thus be added to L early. Thus it is sufficient to insure only that $\text{den}(S_i + v) \geq D \geq 1/2$, eliminating the need to track current densities.

Each round of extending candidate sub-graphs S_i by one vertex is independent of previous rounds, and can thus be handled by a separate thread of execution. Candidate sub-graphs from one thread are passed to the next thread 'mod NUM_THREADS', and handled asynchronously. In the place of L, each thread maintains a sorted list of sub-graphs it has passed to the next thread, which is cleared when the order of sub-graphs being handled is incremented.

2.3 Complexity

Worst-case performance is when G is a clique. For a clique of n vertices, a vertex is either in a sub-graph, or it is not, resulting in $O(2^n)$ sub-graphs to examine. This worst case is ameliorated when G is not a clique by the ordering property of the theorem, which forces initial seed graphs to be of high density, e.g. searching for sub-graphs of density 0.7 forces all initial seed graphs with three vertices to be a clique, and subsequent four vertex seed graphs to have five of six possible edges, etc. In practice, this works well for biological networks that typically exhibit a power law distribution of vertex degree.

2.4 Implementation

ODES is implemented with pthreads in the C programming language as a function call, returning a list of maximal dense sub-graphs. The function signature is composed of the input graph, the minimum desired density of a subgraph, and the minimum number of vertices that a dense sub-graph must contain.

3 RESULTS

Figure 2 records timings using four threads on graphs $G_i$of 20 vertices each, whose density varies over values that range from a minimum cutoff density D to 1, and where G_i is not embedded in any larger graph. Six plots are shown, corresponding to each of six values for D, and the time shown is how long it takes for the algorithm to return G_i.

We observe that the ordering property of the theorem has the greatest effect on reducing the search time when the actual density of G_i is close to the cutoff D, rather than being appreciably denser than D, showing a 3-fold speedup for densities close to D and exhibiting roughly constant running time for densities of G_i close to D as D increases.

Embedding G_i within a 10 000-vertex graph, where the average degree of a vertex is 4, increases the runtime $\sim$10%.

4 FUTURE WORK

(i) Density bins: Figure 2 suggests that as the number of threads available on modern multi-core processors increases, the algorithm should be restructured to run over a collection of density bins, where a set of threads is dedicated to each density bin that handles only those sub-graphs whose density has decreased below the cutoff for the bin above it, thus finding all maximal dense sub-graphs of the same order in roughly constant running time. All single-edge sub-graphs would start off in the bin that handles the highest density range.

(ii) Hash function: profiling reveals a large percentage of time spent in binary searches of the sorted list of sub-graphs passed to the next thread; a suitable hash function may increase performance.

(iii) Overlap indication: some indication of which graphs overlap, and where, is desired.

(iv) Edge exclusion from the initial single-edge sub-graph list: due to its high complexity, ODES does not scale well to large-dense sub-graphs. The guarantee of finding all overlapping dense sub-graphs, however, makes its use attractive in conjunction with heuristic methods, where heuristically determined edges from large dense sub-graphs would be excluded from the initial single-edge sub-graph list for ODES, but still retained in the subsequent search space, thus allowing overlaps with the large dense sub-graphs.

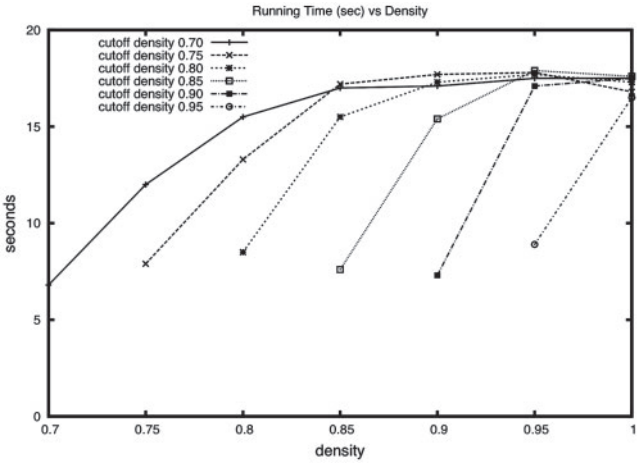

Fig. 2. Running times for various cutoff densities.

ACKNOWLEDGEMENTS

The authors would like to thank Dr Glenn Chappell and Dr Jill Faudree for suggestions, and Dr Thomas Marr for direction.

Conflict of Interest: none declared.

REFERENCES

Adamcsek,B. *et al.* (2006) CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, **22**, 1021–1023.

Hu,H. *et al.* (2005) Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics (ISMB 2005)*, **21** (Suppl. 1), 213–221

Palla,G. *et al.* (2005) Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, **435**, 814–418