Algorithms for Finding Dense Components in a Graph

Richard Hoshino

In this talk, we will study the problem of finding highly connected subgraphs of some large graph G.

To do this, we will define the *density* of a graph, and discuss algorithms that will determine the exact value of this density.

We are interested in measuring the density of a graph so that we can quantify this notion of "highly connectedness". This is especially useful when we apply this concept to analyze sparse graphs such as the Web Graph and the Citation Graph.

In this talk, we will only examine undirected graphs. However, the same ideas and techniques can be used to find the density of any directed graph. (Note: both the Web Graph and the Citation Graph are directed graphs). The following paper discusses this in great detail.

Reference: *Greedy Approximation Algorithms for Finding Dense Components in a Graph*, by Moses Charikar of Stanford University.

Let $G(V, E)$ be an undirected graph, where V are the vertices of G, and E are the edges of G.

Let $S \subseteq V$. We define $E(S)$ to be the edges induced by S, namely

$$E(S) = \{ij \in E : i \in S, j \in S\}$$

We define the *density* of the subset S to be $f(S) = \dfrac{|E(S)|}{|S|}$.

And we define the *density* of the undirected graph $G(V, E)$ to be $f(G) = \max_{S \subseteq V} \{f(S)\}$.

Note that the average degree of the subgraph induced by S is just $2f(S)$.

As a warmup, let's find the density of each of the following graphs:

How can we devise an algorithm that will calculate $f(G)$, or at least return a value close to $f(G)$?

It makes sense to try to remove vertices of low degree.

So here is a greedy algorithm based on this idea:

(1) Let $S := V$. (i.e., let S be all the vertices of G).

(2) Calculate the value of $f(S)$.

(3) Identify i_{min}, the vertex of minimum degree in the subgraph induced by S.

(4) Remove i_{min} from the set S, and go back to step 2. Continue until S is empty.

Let's use Maple to see how this algorithm works on the following graph:

It turns out that this algorithm is reasonably efficient, that it gives a *two-approximation* for $f(G)$.

In other words, if v is the highest value of $f(S)$ obtained during the algorithm, then $v \leq f(G) \leq 2v$.

Theorem: this greedy algorithm is a 2-approximation for $f(G)$.

Proof: The left side of the inequality follows immediately by writing the problem as a linear program, and considering its dual. But before we do that, we need to discuss what a linear program is.

Once we do that, we will prove that the *exact value* of $f(G)$ can always be computed, using linear programming.

Here is an example of a linear program.

$$\begin{aligned} \text{maximize} \;\; 4x_1 + 10x_2 + 6x_3 \\ 2x_1 + 4x_2 + x_3 \leq 12 \\ 6x_1 + 2x_2 + x_3 \leq 26 \\ 5x_1 + x_2 + 2x_3 \leq 80 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{aligned}$$

There are well-known techniques for solving linear programming problems, using a technique known as the simplex method.

Thus, let's attempt to turn our problem of finding $f(G)$ into a linear programming problem.

Let the vertices of G be $v_1, v_2, \ldots, v_n$.

Pick an arbitrary subset S. Here is another way to express $f(S)$. We do the following:

For each vertex v_i, we let

$y_1 = 1$ if $v_i \in S$ and $y_1 = 0$ if $v_i \notin S$.

And for each edge e_{ij}, we let

$x_{ij} = 1$ if $e_{ij} \in E(S)$ and $x_{ij} = 0$ if $e_{ij} \notin E(S)$.

Since $|E(S)| = \sum\limits_{ij \in E} x_{ij}$, and $|S| = \sum\limits_{i \in V} y_i$, we conclude that

$$f(S) = \frac{\sum x_{ij}}{\sum y_i}.$$

Here is an example.

Note that if:

If $y_i = 1$ and $y_j = 1$, then $x_{ij} = 1$.

If $y_i = 1$ and $y_j = 0$, then $x_{ij} = 0$.

If $y_i = 0$ and $y_j = 1$, then $x_{ij} = 0$.

If $y_i = 0$ and $y_j = 0$, then $x_{ij} = 0$.

So in all cases, notice that $x_{ij} \leq y_i$ and $x_{ij} \leq y_j$.

Note: any pair of $0-1$ vectors x and y satisfying these inequalities corresponds to a possible pair of sets, S and $E(S)$.

Thus, we wish to find the maximum value of

$$f(S) = \frac{\Sigma\, x_{ij}}{\Sigma\, y_i}.$$

Given that

1. Each x_{ij} and each y_i is either 0 or 1.
2. $x_{ij} \leq y_i$ and $x_{ij} \leq y_j$ for each pair of adjacent vertices i and j.

Note that we can calculate $f(S)$ for every set S. And so by our definition, the density of our graph, $f(G)$, is the maximum value of $f(S)$, over all possible subsets $S \subseteq V$.

Okay, now we're getting closer. However, having the $\sum_{i \in V} y_i$ in the denominator looks terrible, so let's *normalize* this optimization function.

So we set $\sum_{i \in S} y_i = 1$.

By doing this, we must now change the x_{ij}'s, because we have $x_{ij} \leq y_i$ and $x_{ij} \leq y_j$ for each edge e_{ij}. So now the vectors x and y are no longer $0 - 1$ vectors.

However, we have now turned our problem of finding $f(G)$ to a nice linear program:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{ij \in E} x_{ij} \\
\forall e_{ij} \quad & x_{ij} \leq y_i \\
\forall e_{ij} \quad & x_{ij} \leq y_j \\
& \sum_{i \in V} y_i \leq 1 \\
\forall e_{ij}, v_i \quad & x_{ij}, y_i \geq 0
\end{aligned}
$$

Note: in order for us to use the simplex method, we require $\sum_{i \in V} y_i \leq 1$. However, clearly the optimal value of $\sum_{ij \in E} x_{ij}$ when $\sum_{i \in V} y_i = 1$.

Recall that when we applied our greedy algorithm on the following graph, Maple returned the incorrect value of $\frac{14}{11}$, rather than the correct value of $\frac{9}{7}$.

Let's see what happens when we try to find $f(G)$ using this linear programming method.

Now we prove that the linear program does in fact give an output of $f(G)$, the density of G.

Lemma 1: For any $S \subseteq V$, the value of this linear program is at least $f(S)$.

Lemma 2: Given a feasible solution of this linear program, with value v, we can construct a set $S \subseteq V$ such that $f(S) \geq v$.

Putting Lemmas 1 and 2 together, we have proven that the optimal value of the linear program is equal to $f(G)$, the density of the graph G.

How can we extend this to directed graphs?

Let $G(V, E)$ be an directed graph, where V are the vertices of G, and E are the edges of G.

Let $S, T \subseteq V$. We define $E(S, T)$ to be the set of edges going from S to T, namely

$$E(S, T) = \{ij \in E : i \in S, j \in T\}$$

In the case of the web graph, S corresponds to the *hubs* and T corresponds to the *authorities*.

We define the *density* of the pair of sets S, T to be

$$d(S, T) = \frac{|E(S, T)|}{\sqrt{|S||T|}}$$

And we define the *density* of the directed graph $G(V, E)$ to be $d(G) = \max_{S,T \subseteq V} \{d(S, T)\}$.

(Note: the sets S and T do not have to be disjoint).

We can compute an exact algorithm for determining $d(G)$ using the same linear programming methods we discussed for finding $f(G)$.

The Dual Problem

There is a beautiful method of solving a linear programming problem: one can solve it by examining the *dual problem*.

Let's go back to this example.

For this example, we wished to

$$\begin{aligned}
\text{maximize} \quad x_{12} + x_{23} + x_{34} + x_{14} + x_{24} \\
x_{12} \leq y_1, x_{12} \leq y_2 \\
x_{23} \leq y_2, x_{23} \leq y_3 \\
x_{34} \leq y_3, x_{34} \leq y_4 \\
x_{14} \leq y_1, x_{14} \leq y_4 \\
x_{24} \leq y_2, x_{24} \leq y_4 \\
y_1 + y_2 + y_3 + y_4 \leq 1 \\
x_{12}, x_{23}, x_{34}, x_{14}, x_{24} \geq 0 \\
y_1, y_2, y_3, y_4 \geq 0
\end{aligned}$$

Let's write this as a matrix.

We wish to maximize

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_{12} \\ x_{23} \\ x_{34} \\ x_{14} \\ x_{24} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

$$\text{such that} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{12} \\ x_{23} \\ x_{34} \\ x_{14} \\ x_{24} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

with $x_{12}, x_{23}, x_{34}, x_{14}, x_{24}, y_1, y_2, y_3, y_4 \geq 0$

Primal Problem:

$$\begin{aligned} \text{maximize} \quad z &= c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

Dual Problem:

$$\begin{aligned} \text{minimize} \quad z &= b^T w \\ A^T w &\geq c \\ x &\geq 0 \end{aligned}$$

So in our example, we have,

$$c = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad x = \begin{pmatrix} x_{12} \\ x_{23} \\ x_{34} \\ x_{14} \\ x_{24} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \qquad w = \begin{pmatrix} p_{12} \\ p_{23} \\ p_{34} \\ p_{14} \\ p_{24} \\ q_{12} \\ q_{23} \\ q_{34} \\ q_{14} \\ q_{24} \\ t \end{pmatrix}.$$

So the corresponding *dual problem* is:

Minimize $z = b^T w = t$ such that

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_{12} \\ p_{23} \\ p_{34} \\ p_{14} \\ p_{24} \\ q_{12} \\ q_{23} \\ q_{34} \\ q_{14} \\ q_{24} \\ t \end{pmatrix} \geq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

And this is equivalent to:

$$\begin{aligned} \text{minimize} \quad \text{t} \\ p_{12} + q_{12} \geq 1 \\ p_{23} + q_{23} \geq 1 \\ p_{34} + q_{34} \geq 1 \\ p_{14} + q_{14} \geq 1 \\ p_{24} + q_{24} \geq 1 \\ t \geq p_{12} + p_{14} \\ t \geq p_{23} + p_{24} + q_{12} \\ t \geq p_{34} + q_{23} \\ t \geq q_{34} + q_{14} + q_{24} \\ p_{12}, p_{23}, p_{34}, p_{14}, p_{24}, q_{12}, q_{23}, q_{34}, q_{14}, q_{24}, t \geq 0. \end{aligned}$$

Solving this in Maple, we get the minimum value being $t = \frac{5}{4}$.

And this is clearly the density of the graph G.

We know that the optimal value of the dual problem is the same as the optimal value of the primal problem, and this is confirmed in this example.

Future Work:

In the definition of density $d(G)$ for directed graphs, the sets S and T were not required to be disjoint. So let's ask this question:

We can devise an algorithm to find the value of $d'(G)$, where we maximize $d(S,T)$ over *disjoint sets* S and T. What would be complexity of this algorithm?

Using flow techniques, we can find an algorithm for computing $f(G)$ exactly. However, no one knows of a flow-based algorithm for computing the value of $d(G)$. It would be interesting to see if this can indeed be accomplished.