Algoritmos de aproximación codiciosos para encontrar Componentes densos en un gráfico

Moisés Charikar[-]

Universidad de Stanford, Stanford, CA 94305, EE. UU.
moses@cs.stanford.edu

Resumen. Estudiamos el problema de encontrar subgrafos altamente conectados de grafos dirigidos y no dirigidos. Para grafos no dirigidos, la noción de densidad de un subgrafo que usamos es el grado promedio del subgrafo. Para grafos dirigidos, Kannan y Vinay introdujeron recientemente una noción correspondiente de densidad. Esto está diseñado para cuantificar la alta conectividad de las subestructuras en un gráfico dirigido disperso como el gráfico web. Estudiamos los problemas de optimización de encontrar subgrafos maximizando estas nociones de densidad para gráficos dirigidos y no dirigidos. Este artículo proporciona algoritmos de aproximación codiciosos simples para estos problemas de optimización. También respondemos una pregunta abierta sobre la complejidad del problema de optimización para grafos dirigidos.

1. Introducción

El problema de encontrar componentes densos en un gráfico ha sido ampliamente estudiado [1,2,4,5,9]. Los investigadores han explorado diferentes definiciones de densidad y examinado los problemas de optimización correspondientes a la búsqueda de subestructuras que maximicen una determinada noción de densidad. La complejidad de tales problemas de optimización varía ampliamente con la elección específica de una definición. En este artículo, la noción de densidad que nos interesará es, en términos generales, el grado promedio de un subgrafo. En la Sección 1.1 aparecen definiciones precisas para gráficos dirigidos y no dirigidos.

Recientemente, el problema de encontrar subestructuras relativamente altamente conectadas en el gráfico web ha recibido mucha atención [8,10,11,12]. Los experimentos sugieren que tales subestructuras corresponden a comunidades en la web, es decir, colecciones de páginas relacionadas con el mismo tema. Además, la presencia de una gran densidad de enlaces dentro de un conjunto particular de páginas se considera una indicación de la importancia de estas páginas. El algoritmo de Kleinberg [10] identifica *centros*(listas de recursos) y *autoridades*(páginas autorizadas) entre el conjunto de páginas potenciales relevantes para una consulta. los*centros*se caracterizan por la presencia de un gran número de enlaces a la*autoridades*y el*autoridades*se caracterizan por la presencia de un gran número de enlaces desde el*centros*.

[-] Investigación apoyada por Pierre and Christine Lamond Fellowship, ARO MURI Grant DAAH04–96–1–0007 y NSF Grant IIS-9811904. Parte de este trabajo se realizó mientras el autor visitaba el IBM Almaden Research Center.

K. Jansen y S. Khuller (Eds.): APROX. 2000, LNCS 1913, págs. 84–95, 2000.

Kannan y Vinay [9] introducen una noción de densidad para gráficos dirigidos que cuantifica la conexión relativamente alta y es adecuada para gráficos dirigidos dispersos como el gráfico web. Esto está motivado por tratar de formalizar la noción de encontrar conjuntos de centros y autoridades que estén altamente conectados en relación con el resto del gráfico. En este artículo, estudiamos el problema de optimización de encontrar un subgrafo de densidad máxima de acuerdo con esta noción. Ahora procedemos a definir formalmente las nociones de densidad que utilizaremos en este trabajo. Estas son idénticas a las definiciones en [9].

1.1 Definiciones y Notación

Dejar $GRAMO(V,E)$ sea un grafo no dirigido y $S \subseteq V$. Definimos $mi(S)$ para ser los bordes inducidos por S, es decir

$$mi(S) = \{yo \in mi : i \in s j \in S\}$$

Definición 1. *Dejar $S \subseteq V$. Definimos la densidad $F(S)$ del subconjunto S ser - estar*

$$F(S) = \frac{|E(S)|}{|S|}$$

Definimos la densidad $F(GRAMO)$ del grafo no dirigido $GRAMO(V,E)$ ser - estar

$$F(GRAMO) = \text{máx.}_{S \subseteq V} \{F(S)\}$$

Tenga en cuenta que $2F(S)$ es simplemente el grado promedio del subgrafo inducido por S y $2F(GRAMO)$ es el grado promedio máximo sobre todos los subgrafos inducidos. El problema de la computación $F(GRAMO)$ también se conoce como el *Subgrafo más denso* problema y se puede resolver utilizando técnicas de flujo. (Consulte el Capítulo 4 en el libro de Lawler [13]. El algoritmo, debido a Gallo, Grigoriadis y Tarjan [7], utiliza un flujo máximo paramétrico que se puede realizar en el tiempo requerido para realizar un cálculo de flujo máximo único mediante el algoritmo push-relabel) .

Un problema relacionado que ha sido ampliamente estudiado es el *más denso k-Problema de subgrafo*, donde el objetivo es encontrar un subgrafo inducido de k vértices de máximo grado medio [1,2,4,5]. Se sabe relativamente poco sobre la aproximabilidad de este problema y resolverlo sigue siendo una pregunta abierta muy interesante.

Ahora definimos la densidad para grafos dirigidos. Dejar $GRAMO(V,E)$ sea un grafo dirigido y $S\ T \subseteq V$. Definimos $mi(S\ T)$ para ser el conjunto de aristas que van desde S a T, es decir

$$mi(S\ T) = \{yo \in mi : i \in s j \in T\}.$$

Definición 2. *Dejar $S\ T \subseteq V$. Definimos la densidad $d(S\ T)$ del par de conjuntos $S\ T$ ser - estar*

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

Definimos la densidad $d(GRAMO)$ del grafo dirigido $GRAMO(V,E)$ ser - estar

$$d(GRAMO) = \text{máx.}_{S\ T \subseteq V} \{d(S\ T)\}$$

La noción anterior de densidad para grafos dirigidos fue introducida por Kannan y Vinay [9]. El conjunto S corresponde a los bujes y al conjunto T corresponde a las autoridades en [10]. Tenga en cuenta que para $|S|=|T|$, $d(S\ T)$ es simplemente el número promedio de aristas que van desde un vértice en S a T (o el número promedio de aristas que van a un vértice en T de S). Kannan y Vinay explican por qué esta definición de densidad tiene sentido en el contexto de gráficos dirigidos dispersos como el gráfico web. Tenga en cuenta que en la definición anterior, los conjuntos S y T no están obligados a ser disjuntos.

El problema de la computación d(*GRAMO*) fue considerado en [9]. obtienen un O(Iniciar sesión *norte*) aproximación relacionando d(*GRAMO*) al valor singular de la matriz de adyacencia de *GRAMO* y usando el algoritmo Monte Carlo recientemente desarrollado para la Descomposición en Valores Singulares de una matriz [3,6]. También muestran cómo se pueden utilizar las técnicas SVD para obtener una O(Iniciar sesión *norte*) aproximación para F(*GRAMO*). Dejan abierta la cuestión de resolver la complejidad de la informática d(*GRAMO*) exactamente.

En este trabajo demostramos que la cantidad d(*GRAMO*) se puede calcular exactamente usando técnicas de programación lineal. También damos un algoritmo simple de 2 aproximaciones codiciosos para este problema. Como calentamiento, primero explicamos cómo F(*GRAMO*) se puede calcular exactamente usando técnicas de programación lineal. Luego presentamos un algoritmo simple de 2 aproximaciones codiciosos para este problema. Esto procede eliminando repetidamente el vértice de grado más bajo. Nuestro algoritmo y análisis para calcular la densidad d(*GRAMO*) para gráficos dirigidos se basa en las técnicas de computación F(*GRAMO*) para grafos no dirigidos.

2 Algoritmo exacto para F(*GRAMO*)

Mostramos que el problema de la computación F(*GRAMO*) se puede expresar como un programa lineal. Mostraremos que la solución óptima de este PL es una combinación convexa de soluciones integrales. Este resultado en sí mismo probablemente no sea tan interesante dado el algoritmo exacto basado en el flujo para calcular F(*GRAMO*)[7]. Sin embargo, la técnica de prueba sentará las bases para las pruebas más complicadas en el algoritmo para calcular d(*GRAMO*) luego.

Utilizamos el siguiente LP:

$$\text{máximo} \sum_{yo} X_{yo} \tag{1}$$

$$\forall yo \in mi \quad X_{yo} \leq y_i \tag{2}$$

$$\forall yo \in mi \quad X_{yo} \leq y_j \tag{3}$$

$$\sum_i y_i \leq 1 \tag{4}$$

$$X_{yo}, y_i \geq 0 \tag{5}$$

Lema 1. *Para cualquier* $S \subseteq V$ *, el valor del LP (1)-(5) es al menos* $F(S)$.

Prueba. Daremos una solución factible para el PL con valor $F(S)$. Dejar $X = \frac{1}{|S|}$. Para cada $i \in S$, establecer $\bar{y}_i = X$. Para cada $yo \in mi(S)$, establecer $X_{yo} = X$. todo el resto las variables se establecen en 0. Ahora, $\sum_i \bar{y}_i = |S| \cdot X = 1$. Así, $(\bar{x}, \bar{y})$ es una solución factible al LP. El valor de esta solución es

$$|E(S)| \cdot X = \frac{|E(S)|}{|S|} = F(S)$$

Esto prueba el lema.

Lema 2. *Dada una solución factible del LP (1)-(5) con valor v podemos construir $S \subseteq V$ tal que $F(S) \geq v$.*

Prueba. Considere una solución factible $(\bar{x}, \bar{y})$ al LP (1)-(5). Sin pérdida de generalidad, podemos suponer que para todo yo, $X_{yo} = \min(\bar{y}_i, \bar{y}_j)$.

Definimos una colección de conjuntos S indexado por un parámetro $r \geq 0$. Deja $S(r) = \{i : \bar{y}_i \geq r\}$ y $mi(r) = \{yo : X_{yo} \geq r\}$. Ya que $X_{yo} \leq \bar{y}_i$ y $X_{yo} \leq \bar{y}_j$, $yo \in mi(r) \Rightarrow i \in S(r), j \in S(r)$. También, desde $X_{yo} = \min(\bar{y}_i, \bar{y}_j)$, $i \in S(r), j \in S(r) \Rightarrow yo \in mi(r)$. De este modo $mi(r)$ es precisamente el El conjunto de aristas induce por $S(r)$.

Ahora, $\int_0^\infty |S(r)|\, dr = \sum_i \bar{y}_i \leq 1$. Tenga en cuenta que $\int_0^\infty |E(r)|\, dr = \sum_{yo} \bar{X}_{yo}$. Este es el valor de la función objetivo de la solución de PL. Sea este valor v.

Decimos que existe r tal que $|mi(r)|/|S(r)| \geq v$. Supongamos que no existieran tales r. Después

$$\int_0^\infty |E(r)|\, dr < v \int_0^\infty |S(r)|\, dr \leq v.$$

Esto da una contradicción. Para encontrar tal r, observe que podemos verificar todos los conjuntos combinatoriamente distintos $S(r)$ simplemente comprobando los conjuntos $S(r)$ obtenido al establecer $r = \bar{y}_i$ para cada $i \in V$.

Juntando los Lemas 1 y 2, obtenemos el siguiente teorema.

Teorema 1.

$$\text{máximo}_{S \subseteq V} \{F(S)\} = OPTAR(LP) \tag{6}$$

dónde $OPTAR(LP)$ denota el valor de la solución óptima para el LP (1)-(5). Además, un conjunto S maximizando $F(S)$ se puede calcular a partir de la solución óptima de la LP.

Prueba. Primero establecemos la igualdad (6). Del Lema 1, la RHS $\geq$ el LHS. (Considera el S que maximiza $F(S)$). Del Lema 2, el LHS $\geq$ el RHS. La demostración del Lema 6 da una construcción de un conjunto S que maximiza $F(S)$ de la solución PL óptima.

3 Codicioso 2-Aproximación para $F(GRAMO)$

Queremos producir un subgrafo de $GRAMO$ de grado medio grande. Intuitivamente, deberíamos desechar los vértices de bajo grado para producir dicho subgrafo. Esto sugiere un algoritmo codicioso bastante natural. De hecho, el desempeño de tal un algoritmo ha sido analizado por Asahiro, Iwama, Tamaki y Tokuyama [2] para un problema ligeramente diferente, el de obtener un subgrafo de grado promedio grande en un número dado k de vértices.

El algoritmo mantiene un subconjunto S de vértices. Inicialmente $S \leftarrow V$. En cada iteración, el algoritmo identifica i_{min}, el vértice de grado mínimo en el subgrafo inducido por S. El algoritmo elimina i_{min} del conjunto S y pasa a la siguiente iteración. El algoritmo se detiene cuando el conjunto S esta vacio. De todos los conjuntos S construido durante la ejecución del algoritmo, el conjunto S maximizando $F(S)$ (es decir, el conjunto de grado medio máximo) se devuelve como salida del algoritmo.

Probaremos que el algoritmo produce una aproximación 2 para $F(GRAMO)$. Hay varias formas de probar esto. Presentamos una prueba que puede parecer complicada al principio. Esto preparará el escenario para el algoritmo de $d(GRAMO)$ luego. Además, creemos que la prueba es interesante porque establece conexiones entre el algoritmo voraz y el dual de la formulación LP que usamos en la sección anterior.

Para analizar el algoritmo, producimos un límite superior en la solución óptima. El límite superior tiene la siguiente forma: Asignamos cada borde yo a cualquiera i o j. por un vértice i, $d(i)$ es el número de aristas yo o Ji asignado a i. Dejar $d_{máximo} = máximo_i\{d(i)\}$. (Otra forma de ver esto es que orientaremos los bordes del gráfico y $d_{máximo}$ es el número máximo de aristas orientadas hacia cualquier vértice). El siguiente lema muestra que $F(S)$ está delimitado por $d_{máximo}$.

Lema 3.

$$\underset{S \subseteq V}{\text{máximo}}\{F(S)\} \leq d_{\text{máximo}}$$

Prueba. Considere el conjunto S que maximiza $F(S)$. Ahora, cada borde en $mi(S)$ debe asignarse a un vértice en S. De este modo

$$|E(S)| \leq |S| \cdot d_{\text{máximo}}$$

$$F(S) = \frac{|E(S)|}{|S|} \leq d_{\text{máximo}}$$

Esto concluye la prueba.

Ahora, la asignación de bordes a uno de los puntos finales se construye a medida que se ejecuta el algoritmo. Inicialmente, todos los bordes están sin asignar. Cuando se elimina el vértice de grado mínimo de S, al vértice se le asignan todas las aristas que van desde el vértice al resto de los vértices en S. Mantenemos la invariante de que todas las aristas entre dos vértices en el conjunto actual S no están asignados; todos los demás bordes están asignados. Al final de la ejecución del algoritmo, se asignan todos los bordes.

Dejar $d_{máximo}$ definirse como antes para la asignación específica construida correspondiente a la ejecución del algoritmo codicioso. El siguiente lema relaciona el valor de la solución construida por el algoritmo voraz con $d_{máximo}$.

Lema 4. *Dejar v ser el valor máximo de $F(S)$ para todos los conjuntos S obtenidos durante la ejecución del algoritmo voraz. Después $d_{máximo} \leq 2v$.*

*Prueba.*Considere una sola iteración del algoritmo codicioso. Ya que i_{min}se selecciona para ser el vértice de grado mínimo en$S$, su grado es como mucho $2|E(S)|/|S|\leq 2v$. Tenga en cuenta que a un vértice en particular se le asignan bordes solo en el punto en que se elimina de$S$. Esto prueba que$d_{\text{máximo}}\leq 2v$.

Juntando los Lemas 3 y 4, obtenemos lo siguiente.

Teorema 2.*El algoritmo codicioso da una*2*aproximación para*$F(GRAMO)$*.*

*Tiempo de ejecución.*Es fácil ver que el algoritmo voraz se puede implementar para ejecutarse en$O(norte^2)$ tiempo para un gráfico con*norte*vértices y*metro*bordes Podemos mantener los grados de los vértices en el subgrafo inducido por$S$. Cada iteración implica identificar y eliminar el vértice de grado mínimo, así como actualizar los grados de los vértices restantes, lo cual se puede hacer en$O(norte)$ tiempo. Usando montones de Fibonacci, podemos obtener un tiempo de ejecución de$O(metro+norte$Iniciar sesión$norte)$ que es mejor para gráficos dispersos.

3.1 Intuición detrás del límite superior

El lector puede preguntarse sobre el origen del límite superior de la solución óptima utilizada en la sección anterior. De hecho, no hay nada mágico en esto. Está estrechamente relacionado con el dual de la formulación LP utilizada en la Sección 2. De hecho, el dual de LP (1)-(5) es el siguiente:

$$\min y \tag{7}$$

$$\forall yo \in mi \quad \alpha_{yo} + \beta_{yo} \geq 1 \tag{8}$$

$$\forall yo \quad y \geq \sum_j \alpha_{yo} + \sum_j \beta_{ji} \tag{9}$$

$$\alpha_{yo}, y \geq 0 \tag{10}$$

El límite superior construido corresponde a una solución dual dondeα_{yo},β_{yo}son variables 0-1.$\alpha_{yo}=1$ corresponde al borde*yo*siendo asignado aiy$\beta_{yo}=1$ corresponde al borde*yo*siendo asignado aj. Después$y$corresponde a$d_{\text{máximo}}$. En efecto, nuestra prueba construye una solución dual a medida que se ejecuta el algoritmo codicioso. El valor de la solución dual es$d_{\text{máximo}}$, el límite superior en el Lema 3.

Pasamos ahora al problema de calcular$d(GRAMO)$ para grafos dirigidos$GRAMO$. Aquí, las ideas desarrolladas en los algoritmos para$F(GRAMO)$ para grafos no dirigidos $GRAMO$ resultará muy útil.

4 Algoritmo exacto para$d(GRAMO)$

Recordar que$d(GRAMO)$ es el valor máximo de$d(S\ T)$ sobre todos los subconjuntos$S\ T$de vértices. Primero presentamos una relajación de programación lineal para$d(GRAMO)$. Nuestra relajación LP depende del valor de $|S|/|T|$para la pareja$S\ T$que maximiza$d(S\ T)$. De

Por supuesto, no conocemos esta relación a priori, por lo que escribimos un LP separado para cada valor posible de esta relación. Tenga en cuenta que hay $O(norte^2)$ valores posibles. Para $|S|/|T|=C$, usamos la siguiente relajación LP $LP(C)$.

$$\text{máximo} \sum_{yo} X_{yo} \tag{11}$$

$$\forall yo \quad X_{yo} \leq s_i \tag{12}$$

$$\forall yo \quad X_{yo} \leq t_j \tag{13}$$

$$\sum_i s_i \leq \sqrt{C} \tag{14}$$

$$\sum_j t_j \leq \frac{1}{\sqrt{C}} \tag{15}$$

$$X_{yo}, s_i, t_j \geq 0 \tag{dieciséis}$$

Ahora demostramos el análogo del Lema 1 anterior.

Lema 5. *Considerar $S, T \subseteq V$. Dejar $C = |S|/|T|$. entonces el valor óptimo de $LP(C)$ Por lo menos $d(S, T)$.*

Prueba. Daremos una solución factible (x, s, t) por $LP(C)$ (11)-(16) con valor $d(S, T)$. Dejar $X = \frac{\sqrt{C}}{|S|} = \frac{1}{\sqrt{C} \cdot |T|}$. Para cada $i \in S$, establecer $s_i = X$. Para cada $j \in T$, establecer $t_j = X$. Para cada uno $yo \in mi(S, T)$, establecer $X_{yo} = X$. todo el resto Las variables se establecen en 0. Ahora, $\sum_i s_i = |S| \cdot X = \sqrt{C}$ y $\sum_j t_j = |T| \cdot X = 1/\sqrt{C}$. Por lo tanto, este es un factible solución a $LP(C)$. El valor de esta solución es

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

Esto prueba el lema.

El siguiente lema es el análogo del Lema 2.

Lema 6. *Dada una solución factible de $LP(C)$ con valor v podemos construir $S, T \subseteq V$ tal que $d(S, T) \geq v$.*

Prueba. Considere una solución factible (x, s, t) a $LP(C)$ (11)-(16). Sin pérdida de generalidad, podemos suponer que para todo yo, $X_{yo} = \min(s_i, t_j)$.

Definimos una colección de conjuntos S, T indexado por un parámetro $r \geq 0$. Deja $S(r) = \{i : s_i \geq r\}$, $T(r) = \{j : t_j \geq r\}$ y $mi(r) = \{yo : X_{yo} \geq r\}$. Ya que $X_{yo} \leq s_i$ y $X_{yo} \leq t_j$, $yo \in mi(r) \Rightarrow i \in S(r), j \in T(r)$. También desde $X_{yo} = \min(s_i, t_j)$. De este modo $mi(r)$ es preciso y el conjunto de borde es que go de $S(r)$ a $T(r)$.

Ahora, $\int_0^\infty |S(r)| dr = \sum_i s_i \leq \sqrt{C}$. También, $\int_0^\infty |T(r)| dr = \sum_j t_j \leq \frac{1}{\sqrt{C}}$. Por el Desigualdad de Schwarz,

$$\int_0^\infty \sqrt{|S(r)||T(r)|}\, dr \leq \sqrt{\left(\int_0^\infty |S(r)|\, dr\right)\left(\int_0^\infty |T(r)|\, dr\right)} \leq 1$$

Tenga en cuenta que $\int_0^\infty |E(r)|dr = \sum_{yo} \bar{X}_{yo}$. Este es el valor de la función objetivo de la solución. Sea este valor v.

Decimos que existe r tal que $|mi(r)|/$ no había $\sqrt{|S(r)||T(r)|} \geq v$. Suponer tales r. Después

$$\int_0^\infty |E(r)|dr < v \int_0^\infty \sqrt{|S(r)||T(r)|}dr \leq v.$$

Esto da una contradicción. Para encontrar tal r, observe que podemos verificar todos los conjuntos combinatoriamente distintos $S(r), T(r)$ simplemente comprobando $S(r), T(r)$ obtenido al establecer $r = s_i$ y $r = t_j$ para cada $i \in V, j \in V$.

Tenga en cuenta que el par de conjuntos $S\ T$ garantizada por la prueba anterior no necesita satisfacer $|S|/|T| = C$. Juntando los Lemas 5 y 6, obtenemos el siguiente teorema.

Teorema 3.

$$\text{máximo}_{S\ T \subseteq V} \{d(S\ T)\} = \text{máximo}_C \{OPTAR(LP(C))\} \quad (17)$$

dónde $OPTAR(LP(C))$ denota el valor de la solución óptima para $LP(C)$. Además, conjuntos $S\ T$ maximizando $d(S\ T)$ se puede calcular a partir de las soluciones óptimas al conjunto de programas lineales $LP(C)$.

Prueba. Primero establecemos la igualdad (17). Del Lema 5, la RHS $\geq$ el LHS. (Considera el $S\ T$ que maximicen $d(S\ T)$). Del Lema 6, el LHS $\geq$ el RHS. (Establecer C ser el valor que maximiza $OPTAR(LP(C))$ y considere la solución óptima para $LP(C)$.) La demostración del Lema 6 da una construcción de conjuntos $S\ T$ maximizando $d(S\ T)$ de la solución PL que maximiza $OPTAR(LP(C))$.

Observación 1. Tenga en cuenta que el algoritmo propuesto implica resolver $O(norte^2)$ LPs, uno para cada posible valor de la relación $C = |S|/|T|$. De hecho, esta relación se puede adivinar dentro de un $(1+\varepsilon)$ factor utilizando sólo $O(\text{Iniciar sesión}\, norte/\varepsilon)$ valores. no es muy difícil de demuestre que esto produciría un $(1+\varepsilon)$ aproximación. El lema 5 se puede modificar para incorporar el $(1+\varepsilon)$ factor.

5 Algoritmo de aproximación para *d*(*GRAMO*)

5.1 Intuiciónbdetrás del algoritmo

A partir de los conocimientos adquiridos al analizar el algoritmo codicioso para aproximar F(*GRAMO*), examinando el dual de la formulación LP para d(*GRAMO*) debería darnos algunos consejos sobre cómo un algoritmo codicioso para d(*GRAMO*) deben ser construidos y analizados.

el dual de $LP(C)$ es el siguiente programa lineal:

$$\min \sqrt{C} \cdot \gamma + \sqrt{\frac{d}{C}} \tag{18}$$

$$\forall yo \quad \alpha_{yo} + \beta_{yo} \geq 1 \tag{19}$$

$$\forall i \quad \gamma \geq \sum_{j} \alpha_{yo} \tag{20}$$

$$\forall j \quad d \geq \sum_{i} \alpha_{yo} \tag{21}$$

$$\alpha_{yo}, \gamma, \delta \geq 0 \tag{22}$$

Cualquier solución factible del dual es un límite superior de la solución integral. Esto sugiere naturalmente un límite superior correspondiente a una solución dual donde α_{yo}, β_{yo} son variables 0-1. $\alpha_{yo}=1$ corresponde al borde yo siendo asignado a i y $\beta_{yo}=1$ corresponde al borde yo siendo asignado a j. Después γ es el número máximo de aristas yo asignado a cualquier vértice i (grado de salida máximo). d es el número máximo de aristas yo $\sqrt{}$ asignado a un vértice j (máximo en grados). Entonces el valor de la solución dual $\sqrt{C} \cdot \gamma + \frac{\sqrt{\delta}}{\sqrt{C}}$ es un límite superior en el $d(S\ T)$ para todos los pares de conjuntos $S\ T$ tal que $|S|/|T|=C$.

5.2 Algoritmo de aproximación codicioso

Ahora usaremos los conocimientos adquiridos al examinar el dual de $LP(C)$ para construir y analizar un algoritmo de aproximación codicioso. Como en el algoritmo exacto, necesitamos adivinar el valor de $C=|S|/|T|$. Para cada valor de C, ejecutamos un algoritmo codicioso. la mejor pareja $S\ T$ (es decir, uno que maximiza $d(S\ T)$) producido por todos esos algoritmos codiciosos es el resultado de nuestro algoritmo.

Ahora describimos el algoritmo codicioso para un valor específico de C. El algoritmo mantiene dos conjuntos. S y T y en cada etapa elimina el vértice de grado mínimo en S o el vértice de grado mínimo en T según una determinada regla. (Aquí el grado de un vértice i en S es el número de aristas de i a T. El grado de un vértice j en T se define de manera similar).

1. Inicialmente, $S \leftarrow V,\ T \leftarrow V$.
2. Deja $i_{\min}$ ser el vértice $i \in S$ que minimiza $|mi(\{eso\})|$. Dejar $d_S \leftarrow |mi(\{i_{\min}\}, T)|$.
3. Le $\sqrt{t}$ $j_{\min}$ ser el vértice $j \in T$ que minimiza $|mi(S,\{j\})|$. Dejar $d_T \leftarrow |mi(S,\{j_{\min}\})|$.
4. Si $\sqrt{C} \cdot d_S \leq \frac{1}{\sqrt{C}} \cdot d_T$ gallina

   establecer $S \leftarrow S - \{yo_{\min}\}$ más

   establecido $T \leftarrow T - \{j_{\min}\}$.
5. Si ambos S y T no están vacíos, vuelva al Paso 2.

De todos los conjuntos $S\ T$ producido durante la ejecución del algoritmo anterior, el par que maximiza $d(S\ T)$ se devuelve como la salida del algoritmo.

Para analizar el algoritmo, producimos un límite superior en la solución óptima. El límite superior tiene la siguiente forma sugerida por el dual de $LP(C)$: Asignamos cada borde (dirigido) yo a cualquiera i o j. por un vértice i, $d_{afuera}(i)$ es el numero de aristas *yo* asignado a i. por un vértice j, $d_{en}(j)$ es el número de aristas $i \cdot j$ asignado a j. Dejar $d^{máximo}_{afuera} = máximo_i \{d_{afuera}(i)\}$ y $d^{máximo}_{en} = máximo_j \{d_{en}(j)\}$. los siguiente lema da el límite superior en $d(S\ T)$ para todos los pares $S\ T$ tal que $|S|/|T| = C$ en términos de $d^{máximo}_{afuera}$ y $d^{máximo}_{en}$.

Lema 7.

$$\underset{|S|/|T|=C}{máximo}\{d(S\ T)\} \leq \sqrt{C} \cdot d^{máximo}_{afuera} + \frac{1}{\sqrt{C}} \cdot d^{máximo}_{en}$$

Prueba. Esto se sigue directamente del hecho de que la asignación de aristas a los vértices corresponde a una solución 0-1 de la d $\sqrt{}$ igual a $LP(C)$. Tenga en cuenta que el valor de la la solución dual correspondiente es exactamente $\sqrt{C} \cdot d^{máximo}_{afuera} + \frac{1}{\sqrt{C}} \cdot d^{mx}_{en}$. Nosotros dar una alternativa, prueba combinatoria de este hecho.

Considere el par de conjuntos $S\ T$ que maximiza $d(S\ T)$ sobre todos los pares $S\ T$ tal que $|S|/|T| = C$. Ahora, cada borde en $mi(S\ T)$ debe asignarse a un vértice en S o un vértice en T. De este modo

$$|E(S\ T)| \leq |S| \cdot d^{máximo}_{afuera} + |T| \cdot d^{máximo}_{en}$$

$$d(S\ T) = \frac{|E(S\ T)|}{\sqrt{|S||T|}} \leq \sqrt{\frac{|S|}{|T|}} d^{máximo}_{afuera} + \sqrt{\frac{|T|}{|S|}} d^{máximo}_{en} = \sqrt{C} \cdot d^{máximo}_{afuera} + \frac{1}{\sqrt{C}} \cdot d^{máximo}_{en}$$

Ahora, la asignación de bordes a uno de los puntos finales se construye a medida que se ejecuta el algoritmo. Tenga en cuenta que se obtiene una asignación separada para cada valor diferente de C. Inicialmente, todos los bordes están sin asignar. Cuando se elimina un vértice de cualquiera S o T en el Paso 4, al vértice se le asignan todas las aristas que van desde el vértice al otro conjunto (es decir, si i_{min} se elimina, se le asignan todos los bordes de i_{min} a T y del mismo modo si j_{min} esta borrado). Mantenemos la invariante de que todas las aristas que van desde el conjunto actual S al conjunto actual T no están asignados; todos los demás bordes están asignados. Al final de la ejecución del algoritmo, se asignan todos los bordes.

Dejar $d^{máximo}_{afuera}$ y $d^{máximo}_{en}$ definirse como antes para la asignación específica construida correspondiente a la ejecución del algoritmo voraz. El siguiente lema relaciona el valor de la solución construida por el algoritmo voraz con $d^{máximo}_{afuera}$ y $d^{máximo}_{en}$.

Lema 8. *Dejar v ser el valor máximo de $d(S\ T)$ para todos los pares de conjuntos $S\ T$ obtenerd durante la ejecución del algoritmo codicioso para un valor particular de C. Después* $\sqrt{C} \cdot d^{máximo}_{afuera} \leq v$ y $\frac{1}{\sqrt{C}} \cdot d^{máximo}_{en} \leq v$.

Prueba. Considere una ejecución de los pasos 2 a 5 en cualquier punto del algoritmo. Ya que i_{min} se selecciona para ser el vértice de grado mínimo en S, su grado es como máximo $|mi(S\ T)|/|S|$, es decir $d_S \leq |mi(S\ T)|/|S|$. Similarmente $d_T \leq |mi(S\ T)|/|T|$. Ahora,

$$\min\left(\sqrt{C} \cdot d_S, \frac{1}{\sqrt{C}} d_T\right) \leq \sqrt{d_S d_T} \leq \frac{|E(S\ T)|}{\sqrt{|S||T|}} \leq v.$$

Si $\sqrt{C} \cdot d_S \leq \frac{1}{\sqrt{C}} d_T$, después i_{min} se elimina y se asignan los bordes que van desde i_{min} a T. En este caso, $\sqrt{C} \cdot d_S \leq v$. Si este no fuera el caso, j_{min} se elimina y se le asignan aristas que van desde S a j_{min}. En este caso, $\frac{1}{\sqrt{C}} d_T \leq v$. Tenga en cuenta que un vértice en particular se le asigna v bordes a él sólo si se quita de cualquiera S o T en el Paso 4. Esto prueba que $\sqrt{C} \cdot d_{\text{máximo}}^{\text{afuera}} \leq v$ y $\frac{1}{\sqrt{C}} d_{\text{máximo}}^{en} \leq v$.

Juntando los Lemas 7 y 8, obtenemos lo siguiente.

Lema 9. *Dejar v ser el valor máximo de $d(S,T)$ para todos los pares de conjuntos S,T obtenido durante la ejecución del algoritmo voraz para un valor particular de C. Después,*

$$v \geq \frac{1}{2} \max_{|S|/|T|=C} \{d(S,T)\}.$$

Observe que el par maximizador S,T en el lema anterior no es necesario satisfacer $|S|/|T|=C$.

La salida del algoritmo es el mejor par. S,T producido por el algoritmo codicioso sobre todas las ejecuciones (para diferentes valores de C). Dejar S^*, T^* ser los conjuntos que maximizan $d(S,T)$ sobre todos los pares S,T. Aplicando el lema anterior para el valor específico $C=|S^*|/|T^*|$, obtenemos el siguiente límite en la relación de aproximación del algoritmo.

Teorema 4. *El algoritmo codicioso da una 2 aproximación para $d(GRAMO)$.*

Observación 2. Como en el algoritmo basado en LP exacto, en lugar de ejecutar el algoritmo para todos $\Omega(norte^2)$ valores de C, podemos adivinar el valor de C en la solución óptima de dentro de un $(1+\varepsilon)$ factor utilizando sólo $O(\text{Iniciar sesión}\, norte/\varepsilon)$ valores. no es muy difícil de demuestre que esto perdería sólo un $(1+\varepsilon)$ factor en la relación de aproximación. Necesitamos modificar el Lema 7 para incorporar el $(1+\varepsilon)$ factor.

Tiempo de ejecución. Similar a la implementación del algoritmo voraz para $F(GRAMO)$, el algoritmo codicioso para $d(GRAMO)$ *por un valor particular de C* se puede implementar ingenuamente para ejecutarse en $O(norte^2)$ tiempo o en $O(metro+norte\ \text{Iniciar sesión}\ norte)$ tiempo usando montones de Fibonacci. Por el comentario anterior, necesitamos ejecutar el algoritmo codicioso para $O(\text{Iniciar sesión}\, norte/\varepsilon)$ valores de C para obtener un $2+\varepsilon$ aproximación.

6. Conclusión

Todos los algoritmos presentados en este documento se generalizan a la configuración donde los bordes tienen pesos. En conclusión, mencionamos algunas direcciones interesantes para el trabajo futuro. En la definición de densidad $d(GRAMO)$ para grafos dirigidos, los conjuntos S,T no estaban obligados a ser disjuntos. ¿Cuál es la complejidad de calcular una noción de densidad ligeramente modificada? $d'(GRAMO)$ donde maximizamos $d(S,T)$ sobre conjuntos disjuntos S,T? Tenga en cuenta que cualquier α-algoritmo de aproximación para $d(GRAMO)$ se puede utilizar para obtener una $O(\alpha)$- aproximación para $d'(GRAMO)$. Finalmente, sería interesante obtener un algoritmo basado en flujo para computar $d(GRAMO)$ exactamente, en la misma línea que el algoritmo basado en flujo para calcular $F(GRAMO)$.

Expresiones de gratitud

Me gustaría agradecer a Ravi Kannan por presentarme el problema y darme una versión preliminar de [9]. También me gustaría agradecer a Baruch Schieber por sugerir una mejora al algoritmo en la Sección 5. La versión anterior poco elegante tenía una garantía de aproximación peor.

Referencias

1. Y. Asahiro y K. Iwama. Encontrar subgrafos densos.*proc. 6º Simposio Internacional de Algoritmos y Computación (ISAAC)*, LNCS 1004, 102–111 (1995).
2. Y. Asahiro, K. Iwama, H. Tamaki y T. Tokuyama. Encontrar con avidez un subgrafo denso.*Diario de Algoritmos*, 34(2):203–221 (2000).
3. P. Drineas, A. Frieze, R. Kannan, S. Vempala y V. Vinay. Agrupación en Grandes Gráficos y Matrices.*proc. 10º Simposio Anual ACM-SIAM sobre Algoritmos Discretos*, 291–299 (1999).
4. U. Feige, G. Kortsarz y D. Peleg. el denso*k*-Problema de subgrafo.*algorítmica*, a aparecer. Versión preliminar en*proc. 34º Simposio Anual IEEE sobre Fundamentos de Ciencias de la Computación*, 692–701 (1993).
5. U. Feige y M. Seltser. en el más denso*k*-Problema de subgrafo. Informe técnico del Instituto Weizmann CS 97-16 (1997).
6. A. Frieze, R. Kannan y S. Vempala. Algoritmos rápidos de Monte-Carlo para encontrar aproximaciones de bajo rango.*proc. 39º Simposio Anual IEEE sobre Fundamentos de Ciencias de la Computación*, 370–378 (1998).
7. G. Gallo, MD Grigoriadis y R. Tarjan. Algoritmo de flujo máximo paramétrico rápido y aplicaciones.*SIAM J. en Informática.*, 18:30–55 (1989).
8. D. Gibson, J. Kleinberg y P. Raghavan. Inferir comunidades web a partir de la topología web.*proc. HIPERTEXTO*, 225–234 (1998).
9. R. Kannan y V. Vinay. Análisis de la estructura de gráficos grandes.*manuscrito*, agosto de 1999.
10. J. Kleinberg. Fuentes autorizadas en entornos vinculados por hipertexto.*proc. 9º Simposio Anual ACM-SIAM sobre Algoritmos Discretos*, 668–677 (1998).
11. J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan y A. Tomkins. La web como gráfico: medidas, modelos y métodos.*proc. 5ª Conferencia Internacional Anual sobre Computación y Combinatoria (COCOON)*, 1–17 (1999).
12. SR Kumar, P. Raghavan, S. Rajagopalan y A. Tomkins. Rastreo automático de comunidades cibernéticas emergentes.*proc. 8va Conferencia WWW*, Redes informáticas, 31(11–16):1481–1493, (1999).
13. EL Lawler.*Optimización Combinatoria: Redes y Matroides*. Holt, Rinehart y Winston (1976).