Trabajo de Investigación

k partición de un grafo

Introducción

En el ámbito de la solución a situaciones reales por medio de algoritmos basados en técnicas de Inteligencia Artificial, para todos es claro que nadie a dicho la última palabra en materia de eficacia y robustez de algoritmos. Lo anterior se debe a la peculiaridad que cada situación representa en la realidad, porque cada una de ellas tiene sus propias variables las cuales determinan lo que podría o no modificarse, y además, cada una de estas variables tiene su propio rango de acción; su propio dominio definido.

Existen innumerables situaciones que eventualmente se podrían implementar usando un algoritmo de los descritos anteriormente. En el informe anterior, explicamos una situación característica de los grafos, un típico problema de resolución en tiempo NO polinomial. Este consistía en que dado un grafo determinado con arcos conectados, hacer una bipartición de este con en dos subgrafos con el mismo número de nodos cada uno con el propósito de minimizar la cantidad de arcos que relacionarían ambos subgrafos. No cabe duda que, como se ejemplificó en el informe anterior, esto tiene un sinnúmero de posibles usos. En vista de lo estudiado anteriormente, discutimos esto con el profesor guía del tema y hemos llegado a la conclusión de que esta aplicación es un caso específico de otro aun más general.

Por lo anteriormente estipulado las aplicaciones de este nuevo caso, aun más general, serían mucho más grandes sin perder de vista la especificación de las variables mencionadas en el inciso anterior.

A continuación el informe de la nueva aplicación.

Explicación del problema

En esta oportunidad tengo en el tapete el mismo grafo solicitado en el informe anterior, solo que esta vez lo dividí no en 2 subgrafos, sino que en k subgrafos diferentes. Además de esto los arcos tienen un peso, una ponderación, que les da un cierto orden entre los de su especie.

Ahora bien, a manera de recordatorio, el objetivo principal propuesto en e el informe anterior para ese ejemplo específico, era particionar un grafo en 2 (con una cantidad de nodos cada uno) y minimizar el número de arcos de conexión entre ambos subgrafos(ver figura).

Como lo mencioné en la introducción, este problema básicamente fue descartado debido a que nos dimos cuenta de que su aplicación era bastante limitada. Optamos mejor por expandir las fronteras y observar hasta donde podríamos llegar en la realidad y sin duda que existen otras aplicaciones con una capacidad de abarcar un rango más amplio de situaciones reales.

Partición de grafos en k subgrafos

Si yo decido dividir el grafo en k subgrafos con un numero determinado de nodos cada uno, y definimos un valor de k=2, entonces tenemos el caso anterior, siempre y cuando el objetivo final sea minimizar el número de arcos que conectan ambos subgrafos.

Ahora modificare más aun el problema dedicándonos al objetivo en sí. Según preví, la situación da para muchas interpretaciones desde el punto de vista de que es lo que se desea. Por esto a continuación doy unos cuantos puntos de vista de que es lo que se desea hacer con un grafo determinado después de haberlo

particionado en k subgrafos con el mismo número de nodos cada uno.

- Nodos con un peso específico y los arcos sin peso: si se desea buscar un objetivo claro a este planteamiento, podríamos decir que queremos un parcelamiento de los k subgrafos con el fin de obtener subgrafos con el mismo peso, quizás con la intención de generar una especie de equilibrio del grafo e general. No me convence esta visión del problema porque aparte de ser muy difícil de encontrarle una utilidad real, no presenta mayor dificultad de resolución.
- Nodos con un peso específico al igual que los arcos del grafo: esta situación es sin duda más compleja que la anterior. Lo principal es fijarse en que es lo que realmente nos interesa, o bien un objetivo basado directamente en los nodos(como el caso anterior) o bien basado en los arcos del grafo. Si nos situamos en los nodos como referencia no tenemos mucho que portar. Pero si no situamos en los arcos podríamos darnos cuenta que el peso de los nodos no hacen más que entorpecer la funcionalidad de un objetivo sobre los arcos, por esto se eligió solo pesos de arcos, ya que pesos sobre nodos sería una caso específico del caso con arcos con peso.
- Nodos sin peso y arcos con un peso específico: este es el caso que más adecuado me pareció debido a que tiene una mayor cantidad de aplicaciones prácticas, y tiene un objetivo claro y de óptimas condiciones para aplicar una técnica de algoritmo inteligente.

Descripción del problema en detalle(nodos sin peso y arcos con un peso específico)

Hasta el momento tenemos claro ciertos aspectos del problema. Retomando, dije que tenemos un grafo con n nodos. Este es un grafo completamente conexo en el cual los arcos tienen un peso específico.

Dadas las anteriores características podemos formalizar aun más la situación y expondré que el principal objetivo es dividir el grafo en k subgrafos con una cantidad de nodos diferentes, y la condición principal es que los arcos entre los subgrafos.(ver figura)

¿Porque IA para este tipo de problemas?

Este tipo de problemas tal y como se describió acá, es un tipo de problemas que se resuelve en tiempo NO polinomial(problema de tipo NP). A medida que crece el número de variables, el tiempo de resolución total aumenta en forma exponencial. Generalmente esto ocurre en gran medida para problemas de optimización. En cambio, si este fuese un problema de satisfacción de restricciones solamente, a pesar de que se trata con heurísticas parecidas, es más probable que se llegue a la solución más rápidamente que con el otro método que además de pedir una solución, pide la óptima.

Si se intentara resolver esto con métodos tradicionales existentes, el problema radicaría en como representar las restricciones adecuadamente. Las restricciones varían de acuerdo a la especificación del problema, entonces, si yo quisiese implementar una solución a esta situación debo basarme necesariamente en su especificidad: sus restricciones.

Por otro lado, como abordé esta situación con una técnica de IA tuve que adaptar la metodología y así generar un algoritmo robusto SOLO para problemas como los míos. Lo que hice fue hacer una mezcla de Hill Climbing con restart incluido y algo Algoritmo voraz todo esto para darle más eficacia a el método resolutivo.

Definición formal del problema

A continuación el Objetivo, variables, dominios, restricciones y función objetivo del problema planteado.

Objetivo

Dado un grafo de n nodos completamente conexo y sus arcos con un peso determinado. El objetivo es dividir

el grafo K subgrafos. Todos estos subgrafos deben tener incorporados a lo menos 2 nodos y a lo más el doble de los nodos que habría de existir en cada subgrafo si se divide el grafo total en K subgrafos de igual cantidad de nodos. La principal idea y final es que estos subgrafos al quedar conectados entre sí por arcos, tengan el menor grado de cohesión posible, es decir, que los arcos que unen los subgrafos pesen lo menos posible y de esa manera poder evitar una cierta dependencia trascendental entre ellos.

Variables

En cuanto a la información de los nodos

Xif :representa al nodo i que pertenece al subgrafo f

i = {1,2,3,... ,n}

f = {1,2,3,... ,l}

l : # total de grafos

n: # total de nodos

En cuanto a la información de los arcos

pij :representa al peso o ponderación del arco que sustentan los arcos i y j

i = {1,2,3,... ,n}

j = {1,2,3,... ,n}

Dominios

Estos son pesos discretos entre 1 y 100

,si el nodo i NO pertenece al subgrafo j

,si el nodo i SI pertenece al subgrafo j

Restricciones

Básicamente la única restricción trascendental es que hay un número mínimo y máximo de nodos pertenecientes a cada subgrafo, a continuación las definiciones formales.

Vemos que la cantidad máxima de nodos, sumadas todas las variables, es n

Donde l es el número promedio de nodos por subgrafo si se desea una partición de k subgrafos sobre un total de n nodos.

Para un determinado subgrafo f la cantidad de nodos esta limitada por abajo con 2 y arriba con 2l. Esto para no disparar la cantidad de nodos agrupada.

La función objetivo

Esta función objetivo como ya se señaló en una oportunidad pretende minimizar los arcos de conexión entre

subgrafos. Lo anterior es equivalente a analizar su complemento también, o sea, intentar maximizar el peso de los arcos que pertenecen a un subgrafo. Por lo tanto la función objetivo sería la siguiente

Ahora, a esta función objetivo le vamos a incorporar una penalización, relativa a la restricción de que tiene un número mínimo y máximo de nodos por subgrafo. Esta referencia no la habíamos mencionado anteriormente porque que se basa en un subjetivismo personal de la solución final. Acá pienso, y que sería muy cómodo(aunque todo depende del contexto donde se este desarrollando la aplicación), que la mayoría de los nodos por subgrafo se mantenga constante en a lo largo del grafo en general, digo esto porque en la realidad lo lógico de una aplicación de este tipo es que se tienda a esto.

Representación del problema

La forma de representar el grafo será por medio de una matriz de conectividad, donde cada casillero indicará el peso de los arcos que sustentan los nodos señalados en la fila y la columna en particular(ver figura).

	1	2	3			n
1	−	0	12			5
2	0	−	67			0
3	12	67	−			99
n	5	0	99			−

En la figura anterior los casilleros que indican 0 implica que entre el nodo i(perteneciente a la fila) y el nodo j(perteneciente a la columna) no existe un arco que los sustenta. Por el contrario aquellos casilleros distintos de 0 con un valor X implica que SI existe un arco entre el nodo i(fila) y j(columna) y cuyo peso es X(valor que fluctúa entre 1 y 100).

Para llevar a cabo un camino hacia una solución debemos definir la representación de una solución provisoria. Vamos a definir un individuo.

Donde para llenar cada una de las casillas multiplicaremos la variable por f(# de subgrafo)

X11*1=1, X23*3=3, X34*4=4, X4k*k=k,....., Xn2*2=2

Otra cosa importante es como vamos a ir llenando(método) esta solución para generar un primer individuo y así poder ir formando poblaciones, ¿porque esto?, tan simple porque si elegimos términos aleatorios, estos seguramente van a costar caro debido a que no van a respetar las restricciones, luego las verificaciones que se vendrán a posteriori por cada instanciación de una situación inicial, tendremos el mismo proceso una y otra vez.

Por lo anterior, pienso que un algoritmo voraz es lo adecuado para obtener una primera situación inicial.

Pienso que posibilidades de resolución hay muchas, es más, todas las vistas en clases sirven para encontrar una solución sin necesariamente perder de vista lo del algoritmo voraz. Debido a la experiencia que tengo del tema podría señalar que una combinación de algoritmo voraz y Hill Climbing con restart daría buenos resultados.

Hill Climbing

Hilando un poco más fino en lo que respecta a la solución del problema, señalaremos que acá existen dos instancias de iteración:

- La respecta a la cantidad de restart que se van a hacer, lo cual le da el grado de exploración del problema.
- Por otra parte, esta la cantidad de iteraciones que se harán para realizar mutaciones sobre una determinada variable con el fin de ir siempre aumentando el valor de la función de objetivo.

Solución propuesta

En este momento la cancha ya esta rayada, por lo tanto, a continuación asiento el problema planteando el algoritmo resolutivo de la k partición de un grafo.

Paso 1: definición constantes

rmax: = # máximo de iteraciones relativas a restart

emax = # máximo de iteraciones relativas a mejores soluciones(explotación).

mejor_solucion = vacío

Paso 2: si rmax > 0

Generar una solución inicial con un algoritmo voraz

sino

ir al **Paso 4**

Paso 3: si emax > 0

Seleccionar una variable Xi

Buscar un valor del dominio que mejore la función objetivo según una heurística

si existe un valor

mejor_solucion = sol. con nuevo valor

emax = emax − 1

ir al **Paso 3**

sino

emax = 0

sino

rmax = rmax −− 1

ir al **Paso 2**

Paso 4: Retornar mejor_solucion

Terminar

Respecto a temas de investigación relacionados

Para ser sincero, averigüé con profesores del departamento y recibí algunas ideas que me sirvieron para interpretar el problema. Además, busque mucho material con el fin de poder encontrar información de nuevas técnicas evolutivas más originales y novedosas. Existen muchas pero en el ámbito de multiparticionamiento de grafos encontré un solo paper.

Me use en contacto con Rina Dechter por medio de correo electrónico con el propósito de darme algunos materiales. Felizmente ella no vaciló en prestarme ayuda y recibí algunos papers donde mi trabajo tenía alguna relación. No sé si sea el momento adecuado para señalarlo pero recibí un paper(de ella) donde explicaba todos los métodos de recorrido de grafos que nosotros estudiamos en clases.

En uno de los paper que ella me mandó encontré a un multiparticionamiento de grafos fue el particionamiento de grafos en DOS subgrafos con el principal objetivo es minimizar el peso del total de arcos que unen ambos subgrafos y también el multiparticionamiento con el mismo objetivo que yo tengo planteado.

Vi que acá se aplica un algoritmo resolutivo denominado Bottleneck algorithm(algoritmo cuello de botella). Este algoritmo es capaz de resolver los problemas en un tiempo polinomial debido a una función de decisión adecuada denominada test, PERO esta función test, que es una decisión que realiza el algoritmo en tiempo NP.

Veremos la definición de la generalización del problema de particionamiento del grafo bottleneck.

Las instancias son:

G=(V,E), grafo con V vértices y E arcos

k=1... k, numero de subgrafos

Wj= subconjunto de nodos que forman el subgrafo

fj(n),gj(n) especifican los máximos y mínimos peso por subgrafo j

V1,... ,Vk subparticiones del grafo.

Deben satisfacer que " j: gi(n)" |Vj| "fj(n), y −−>Wj [Author:DP]" Vj.

Acá tenemos un función de Test que diría:

Test(Gi): ¿es acaso Gi una partición de componentes conexas que resulte una partición legal?

Y su algoritmo sería:

1.− Sea C un set de componentes conexas de Gi

2.− Inicializa Vj = , j=1,... ,k

3.− for cada c " C

- si existe j1"j2 tal que (C"W1") y (C"W2") entonces

retornar NO

STOP

- if (C"W1")

C = C − {c}

fj(n) = fj(n) − {c}

gj(n) = gj(n) − {c}

Vj = Vj " {c}

4.− llamar a k−subgrafo−suma con Aj=fj(n) y Bj=gj(n), j=1,...,k

si k−subgrafo−suma retorna NO entonces

retorna NO

sino

retorna SI (Vj = Vj " Sj, donde Sj contiene las componentes elegidas

para es el subgrafo j)

Como ya señalamos anteriormente, en general el problema de la partición del grafo bottleneck es un problema que se resuelve en tiempo polinomial, a pesar de esto, este es un problema que se soluciona en tiempo NP−duro. En nuestro caso el algoritmo bottleneck descrito lo resuelve en un tiempo pseudo polinomial, del orden de O(n2log(n)), donde n es el número de vértices del grafo. Si se trata de grafos densos este tiempo de resolución es asintóticamente óptimo a cualquier algoritmo que requiera un ordenamiento del peso de los arcos con un orden mas o menos de O(mlog(m)), por supuesto que este no es nuestro caso.

Conclusiones

Debido a mis problemas de organización no pude investigar más con detalle la existencia de trabajos respecto al tema presentado en este informe, además me hubiese gustado también generar un programa en un lenguaje de programación.

Puedo inferir que, por algo que leí en algún paper relativo al porque no mas dedicación de los investigadores a este tema, el futuro de este punto específico ya terminó, se habla mas de una bipartición como base de futuros proyectos pero no sé, en lo personal, de futuros proyectos basados en la k partición de un grafo.

Creo que de todas las técnicas que yo conozco, la combinación que yo elegí acá fue la mas adecuada porque es un método relativamente robusto que no requiere de muchos cálculos, no así como lo hubiese sido alguna implementación basada en algoritmos genéticos.

Mínimo número de arcos

Subgrafo 1

Subgrafo 2

$$p_{ij} = \{1,2,3,\ldots,100\}$$

$$X_{if} = \begin{matrix} 0 \\ 1 \end{matrix}$$

$$l = \frac{n}{k}$$

$$2 \quad \sum_{i=1}^{n} X_{if} \quad 2l$$

$$Max \qquad X_{if} \quad p_{ij} \quad X_{jf} - \sum_{f=1}^{k} \sum_{i=1}^{n} X_{if} - l \quad ^2$$

$$Max \qquad X_{if} \quad p_{ij} \quad X_{jf}$$

$$\sum_{f=1}^{k} \sum_{i=1}^{n} X_{ij} = n$$