Problema de Coloración de Gráficas
Algoritmo de Búsqueda Gravitacional

Seminario de Ciencias de la Computación B

Canek Peláez Valdés

Universidad Nacional Autónoma de México

Sánchez Correa Diego Sebastián

1. Introducción

El problema de coloración de gráficas, específicamente la modalidad que involucra la búsqueda del número cromático de la gráfica, es uno de los problemas NP-duros que tienen una gran aplicación en la resolución de problemas del mundo real.

Una de estas es la optimización del uso de los registros disponibles en un procesador para la ejecución de un algoritmo; problema para el que es encontrado rápidamente una aproximación por los compiladores pero del cual, sin embargo, no se puede garantizar una solución óptima, dada la naturaleza del problema.

1.1. El problema

Una gráfica se define como una par ordenado compuesto de un conjunto de vértices V y un conjunto de aristas E, es decir, una gráfica es representada como sigue

$$G = (V, E)$$

donde

$$V = \{v_1, ..., v_n\}$$
$$E \subseteq V \times V$$

El problema es planteado sobre una gráfica no dirigida, es decir, donde se tiene que

$$(v, w) \in E \Rightarrow (w, v) \in E$$

El vecindaro de un vértice se define como todos los vértices para los cuales existe una arista que los une, es decir

$$N(v) = \{w \in V \mid (v, w) \in E\}$$

El problema de coloración de gráficas pregunta cuál es la mejor asignación de colores para cada uno de los vértices contenidos en la gráfica que minimice el número de colores usados, cumpliendo en todo caso que dos colores adyacentes no compartan el mismo color.

El número cromático define, dada una gráfica no dirigida, el número de colores mínimo que debe tener para colorearla por completo. Dadas las condiciones del problema, es posible obtener múltiples soluciones, ninguna de ellas pudiendo llamarse la más óptima (derivado de la naturaleza no determinista del problema).

1.2. La heurística

La heurística de búsqueda gravitacional (GSA) propone un modelo de búsqueda que sigue un patrón de enjambre, es decir, no existe un solo agente que realice la búsqueda; esta se divide entre diversos entes que permiten una mayor exploración y, eventualmente, una mayor explotación.

La inteligencia de enjambre es un modelo donde el comportamiento colectivo emergente is el resultado de un proceso de organización personal, donde los agentes están envueltos, a través sus acciones repetidas e interacción, com su ambiente en evolución. [4]

Modelos de este tipo son bastante populares; entre ellos se encuentran PSO (Eberhart and Kennedy, 1995), ACS (M. Dorigo and V. Maniezzo, 2008), etcétera.

En esta heurística se plantea el uso de entes (soluciones) que se afectan mútuamente a través de fenómenos físicos, siguiendo las leyes de Newton.

En el algoritmo propuesto los agentes que buscan son una colección de masas que interactúan entre sí basándose en la gravedad Newtoniana y las leyes del movimiento. [2]

A pesar de ser esta una de las primeras concepciones de los fenómenos como soluciones a problemas combinatorios, se usará una variación que lo ha tenido como inspiración y es descrito en [4].

Se define un conjunto de agentes

$$B = \{b_1, b_2, ...b_N\}$$

correspondientes a la cada nodo de la gráfica. Cada agente navegará sobre un mundo tórico [1] de acuerdo a un vector $\vec{v_i}$. En cada momento sabemos la posición de cada agente $p_i(t) = (x_i, y_i)$ donde x_i y y_i son coordenadas cartesianas en el espacio. Cuando $t = 0$, tenemos la posición inicial de los agentes $p_i(0) = (x_{0i}, y_{0i})$.

Suponemos que queremos colorear la gráfica con K colores, denotando como $C = \{1, 2, ..., K\}$ al conjunto de colores, donde K no debe ser menor que el número cromático asociado a la gráfica para que el algoritmo converja. Asignamos a estos colores, K puntos en el espacio, los colores objetivo $CG = \{g_1, ..., g_K\}$, dotados de una atracción gravitacional resultando en un componente de velocidad $\vec{v_{gc}}$ afectando a los agentes. La fuerza de atracción disminuirá con la distancia, pero afectará a todos los agenetes en el espacio.

[4] define el sistema como una tupla:

$$F = (B, CG, \{\vec{v_i}\}, K, \{\vec{a_{i,k}}\}, R)$$

donde:

- B es el conjunto de agentes
- $\{\vec{v_i}\}$ es el conjunto de vectores en el instante t

[1] Un toro es *la colección de todos los puntos* $(x, y) \in \mathbb{R}^2$ *bajo la relación de equivalencia* $(x, y) \sim (a, b)$ *cuando* $x - a, y - b \in \mathbb{Z}$ [1]

- K es el número cromático hipotético [2]
- $\{\vec{a_{i,k}}\}$ es el conjunto de fuerzas de atracción de los colores objetivo ejercidas en los agentes.
- R denota las fuerzas de repulsión en el vecindario de los colores objetivo.

1.2.1. Velocidad de un agente

La velocidad de una agente está definida de la siguiente manera:

$$\vec{v_i}(y+1) = \begin{cases} 0 & c_i \in C \ \& \ (\lambda_i = 1) \\ d \cdot \vec{a_{i,k^*}} & c_i \notin C \\ v_r \cdot (p_r - p_i) & (c_i \in C) \ \& \ (\lambda_i = 0) \end{cases}$$

donde

- d es la distancia de la posición p_i del agente a la posición del color objetivo más cercano g_{k^*}
- $\vec{a_{i,k^*}}$ representa la fuerza de atracción que el color objetivo más cercano ejerce sobre el agente
- v_r es la magnitud de un vector aleatorio moviendo al agente hacia una dirección aleatoria p_r cuando este es expulsado de un color objetivo.
- λ_i es un parámetro que representa el efecto del grado de confort del agente. Cuando un agente b_i alcanza el radio de influencia de un color objetivo en un instante t, su velocidad se torna 0.

1.2.2. Dinámica dentro de un color objetivo

La interacción de los agentes y los colores objetivo se define a través de un radio de influencia generado por los colores. Y cuando el agente esté dentro de este, dejará de moverse y el nodo correspondiente en la gráfica será asignado a este color. Se denota al conjunto de agentes cuya posición en el espacio es la región que se encuentra los suficientemente cerca al vecindario de un color como

$$N(g_k) = \{b_i \ \text{ t.q. } ||p_i - g_k|| < r\}$$

Donde r representa el radio de influencia del color.

A pesar de que dentro del radio de influencia del color no hay más atracción gravitacional, puede estar presente cierto grado de repulsión entre agentes que están conectados a través de una arista en la gráfica G. Esta repulsión solo es efectiva para los agentes dentro del vecindario del mismo color objetivo. Este efecto se modela definiendo una función que tiene un valor de 1 si un par de agentes tienen una arista en común, y 0, en otro caso. Las fuerzas de repulsión que experimenta el agente b_i de los agentes en el color objetivo g_k se definen como sigue:

$$R(b_i, g_k) = \sum_{N(g_k)} \textit{repulsión}\ (b_i, b_j)$$

[2] Este se definirá como el número cromático para gráficas creadas artificialmente y como un número aleatorio (lo suficientemente grande) para gráficas, de la misma manera, aleatorias.

1.2.3. Confort

En cada paso del proceso iterativo en el que un agente permanece en un color objetivo sin ser perturbado, su confort aumenta (hasta llegar a un máximo definido). Cuando otro agente b_i, fuera del color objetivo g_{k^*} intenta entrar al vecindario de ese color objetivo, la fuerza de repulsión $R(b_i, g_{k^*})$ es evaluada. Si la repulsión es mayor que cero entonces el agente entrante desafiará la estabilidad del vecindario y al menos un agente tendrá que abandonarlo (el cual puede ser él mismo). Si los valores de confort de los agentes desafiados tienen valores mayores a cero, entonces su confort disminuirá en una unidad. Si su confort llega a cero, entonces el agente es expulsado del color objetivo a una posición p_r aleatoria en el espacio con velocidad v_r.

1.2.4. Función de Costo

La función de costo definida en la configuración espacial del sistema global es:

$$f(B, CG) = |\{b_i \ \ t.q. \ \ c_i \in C \ \& \ R(b_i, g_{ci}) = 0\}|.$$

Esta función de costo es el número de nodos de la gráfica que tienen un color asignado y sin ningún conflicto dentro de un color objetivo. Los agentes fuera del vecindario de un color objetivo no pueden ser evaluados, para que estos sean parte de la solución al problema.

Se destacan a la dimensión del mundo y al radio de influencia de los colores objetivo como factores importantes para la determinación de la velocidad de convergencia del alforitmo. Si el mundo es grande y el radio pequeño el algoritmo convergerá lentamente, monótonamente; si el mundo es reducido y el radio es grande el algoritmo será más rápido pero la convergencia será inestable porque el algoritmo caerá en mínimos locales y necesitará un aumento en la energía transitoria para salir de ellos. La explicación de este comportamiento es que el mundo no está normalizado y la magnitud del vector velocidad puede ser más grande que el radio de influencia del color objetivo y puede cruzar un color sin caer en él.

Cuando todos los agentes se detienen, tenemos que $f(B, CG) = n$ y la asignación de K colores a la gráfica G ha sido concluida exitosamente.

2. Diseño

A pesar de que inicialmente se siguió como guía el algoritmo descrito en [2] (y la versión discreta del mismo, en [3]), opté por implementar la versión dada en [4]; esto porque a pesar de no basarse en la descripción original, donde el autor concebía a las soluciones como agentes que eran modificados y atraídos a otras soluciones a través de un modelo donde aquellas evaluadas positivamente por la función de costo eran capaces de atraer a una gran cantidad de las restantes; mientras que las soluciones que podían mejorar, no tenían tal capacidad. Esto permitía una gran exploración (derivado por su naturaleza de enjambre) que, a su vez, conluía con buenos resultados, dada su buena concepción de la explotación; planteaba una algoritmo similar (siguiendo la misma *meta idea*) pero ya estaba completamente adaptado a los términos que el problema de coloración de gráficas requería.

2.1. La Gráfica

En este nuevo algoritmo se concebía a la gráfica como un mundo tórico donde cada uno de los agentes ya no correspondían a una solución del problema, sino a los nodos de la gráfica.

2.2. Analizador Sintáctico

Siguiendo las guías de diseño de programas similares que buscan encontrar el número cromático de gráficas aleatorias, encontré que existe un estándar de representación de gráficas.

DIMACS consiste en un formato para gráficas no dirigidas e involucra las siguientes reglas de sintaxis.

- Un archivo de entrada contiene toda la información de una gráfica no dirigida.
- Los nodes están numeradas de 1 a n (siendo este el número total de nodos).
- Se asume que los archivos de entrada se encuentran en un estado lógicamente correcto i consistentes: los identificadores de los nodos son válidos, los nodos son únicos y exactamente m aristas son definidas.
- Los comentarios pueden aparecer en cualquier parte del archivo. Cada comentario comienza con el carácter c.
- Debe haber una *línea del problema* por archivo. Esta debe aparecer antes de cualquier nodo o cualquier línea descriptiva. Sigue el siguiente formato:

```
p FORMAT NODES EDGES
```

Donde `p` denota que se trata de la línea del problema. El campo `FORMAT` debería contener la palabra *edge*[1]. El campo `NODES` contiene un entero que define *n* (el número de nodos). Mientras que el campo `EDGES` contiene un entero que define *m* (el número de aristas).

- Habrá un descriptor de arista por cada arista de la gráfica, cada uno con el siguiente formato:

```
e u v
```

Representa la arista (u, v), esta aperecerá solo una vez y no deberá ser repetida como (v, u).

2.3. El problema

Se planteó el problema como uan abstracción de una gráfica que posee un significado geométrico; es decir, esta vivirá en un mundo cartesiano (tórico, específicamente).

La gráfica posee un arreglo de nodos y una matriz de adyacencias. Ambos poseyendo la ventaja de realizar búsquedas en tiempo constante (aprovechando el hecho de que no serán de tamaño variable durante la ejecución del algoritmo); el primero, abstrayendo los elementos como nodos que tienen coordenadas e identificador asociados; mientras que el segundo pretende acceder de manera rápida a las conexiones de la gráfica a partir de los indentificadores y donde dos vértices v_i y v_j estarán conectados si hay un 1 en la coordenada (i, j) de la matriz (evitando abstracciones en estructuras, como en el primer caso).

2.4. Algoritmo de Búsqueda Gravitacional

La heurística se ejecuta a partir de un solo ciclo que verifica, en cada paso iterativo, si la función de costo (evalúa cuántos nodos tienen un color asociado) devuelve el número de nodos total. En adición, contemplando el caso en el cual el número cromático hipotético sea menor al real, se cuenta con un número de iteraciones máximas que previenen ejecución sin fin. Cabe destacar que si el algoritmo termina de esta última forma, la heurística no habrá encontrado una solución para el problema.

[1] Solo se trata de una convención que mantiene consistencia con desafíos previos relacionados al formato.

3. Implementación

La implementación siguió un modelo orientado a objetos (sin la inclusión de herencia/ polimorfismo) que pretende organizar, modularizar y facilitar el desarrollo y el mantenimiento del código en el lenguaje de programación C.

Cada una de los archivos representa una clase y contiene funciones asociadas a la misma, teniendo como parámetro al objeto en cuestión.

Las `structs` (estructuras) se definen dentro de los archivos de código y no dentro de los encabezados. Esto con el fin de restringir el acceso a los elementos privados a los *getters* y *setters*, así como la creación de objetos del mismo tipo, designando el trabajo a los constructores.

3.1. Problemas

La administración de memoria en un lenguaje de programación como C conlleva un orden más severo que permite una menor abstracción en su uso para gozar de una mayor eficiencia.

Fugas de memoria estuvieron presentes, sobre todo en la devolución de apuntadores desde las funciones; se torna complicado administrar su uso y liberación al contar con una cantidad considerable de funciones de este tipo.

3.1.1. Generador de Números Aleatorios

En ciertos casos, en la generación de las coordenadas los colores de la gráfica se contaba con una repetición en las coordenadas x y y, algo singular, ya que hubo una invocación de la función $drand48\_r$[1] para cada una de ellas. Se revisará con más detalle y se espera corregir en un futuro próximo.

3.2. Parámetros

El programa se ejecuta en la línea de comandos, recibiendo los parámetros de la misma a través como parámetros del programa y a la gráfica en un archivo con formato DIMACS en un archivo de texto.

```
./GCP_GSA -f [archivo de entrada] [opciones]
```
Comando de ejecución del algoritmo

Donde las opciones incluyen:

[1] Se trata de una función generadara de números aleatorios que es compatible con la paralelización del sistema; a pesar de que esta no se haya implementado.

- `-s` la semilla
- `-n` el número cromático hipotético
- `-d` la dimensión de la gráfica[2]
- `-i` el número de iteraciones máximo (en caso de no satisfacer la función de costo)
- `-c` el valor de confort máximo
- `-v` la ejecución verbosa del algoritmo[3]
- `-r` el radio de los colores objetivo

[2]Con esto me refiero al tamaño de una lado de la gráfica, es decir, del mundo tórico sobre el que se ejecutará la heurística, dado que se ha decidido que esta sea cuadrada.

[3]Mostrará las coordenadas finales de los agentes y colores.

4. Experimentación

La experimentación se basó en el uso de gráficas predefinidas para el problema de coloración de gráficas. Específicamente, se usó una gráfica de 300 nodos y 21695 aristas para la cual se tiene un número cromático de 28. El algoritmo fue capaz de obtener resultados decentes en cuestión de unos segundos. Para este caso, se corrió con una hipótesis de 80, 100,000 iteraciones y un mundo tórico de dimensión 100, con un radio de 50 para los colores objetivo. El mejor resultado obtenido fue de 36 (todos rondaron las cuatro decenas).

5. Conclusiones

A pesar de que la experimentación no fue exhaustiva, los resultados obtenidos parecen establecer una base sólida para futuros experimentos con una gama más amplia de gráficas no generadas artificialmente y con una implementación de concurrencia que permita una generación más rápida y eficiente de resultados.

Bibliografía

[1] Padraic Bartlett. Toroidal graphs, 2015.

[2] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. Gsa: A gravitational search algorithm. Information Sciences, 2009.

[3] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. Bgsa: Binary gravitational search algorithm. Natural Computing, 2010.

[4] Israel Carlos Rebollo Ruiz. Gravitational swarm intelligence for graph coloring, 2012.