Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI-5651 Diseño de algoritmos I
enero-abril 2010, Profesor: Oscar Meza

Proyecto 2

Se quiere que implemente una modificación del algoritmo de "backtracking" denominado "Brown's modified algorithm with look-ahead rule" para hallar una coloración mínima de un grafo, presentado en el artículo de Peemöller [1].

Las modificaciones son las siguientes:

a) No se utilizará el procedimiento "label" para el "backtracking", sino que se procederá a hacer el "backtracking" según la rutina "BACKWARDS" y con los criterios de la sección 3.1 para definir el conjunto CP, mencionados en el artículo de Kubale&al[2]. Pero sí, debe considerar el "backtracking" no cronológico propuesto en la penúltima línea de la segunda columna de la página 2 de [2].
b) Estando en un nodo del árbol de "backtracking", el próximo vértice a colorear será el de máximo grado de saturación rompiendo los empates con el de mayor grado en el grafo inducido por los vértices no coloreados. Este criterio es el que utiliza Korman[3] en la sección 8.2.1.2 (ó la sección 3.4.(ii) de [2])
c) Los vecinos de un nodo válido del árbol de "backtracking" deberán ser considerados en el siguiente orden: asignar primero al vértice v a colorear, el color, aún no considerado, que tenga más vértices coloreados de ese color.

El algoritmo, que llamaremos DSATUR+, procederá como sigue:

1) Se obtiene la coloración dada por la heurística BreLaz+interchangeProcedure, para obtener una cota superior (COTA_SUP) del número cromático.
2) Se obtendrá una cota inferior (COTA_INF) del número cromático obteniendo la máxima clique que se obtiene de aplicar BreLaz+interchangeProcedure N veces (donde N es el orden del grafo) partiendo de cada uno de los vértices del grafo, en lugar del vértice de mayor grado (ver sección 1 de Brelaz [5]). Note que si COTA_SUP es igual a COTA_INF, no tiene sentido entrar al "backtracking" pues ya habremos conseguido la coloración mínima.
3) El parámetro q en [1], será inicializado con COTA_SUP.
4) Sean $\{v_1, v_2, \ldots, v_p\}$ los vértices de la clique máxima encontrada en (2). El "backtracking" se comienza desde el nodo $\{\{v_1\}, \{v_2\}, \ldots, \{v_p\}\}$
5) Luego se procede como indica el algoritmo de "backtracking" antes descrito.

Si el algoritmo de "backtracking" (punto (5)) tarda más de 5 minutos en una instancia dada, se aborta el programa. Se utilizarán las siguientes instancias para hacer las pruebas:

1) Las instancias del proyecto 1 con cotas iniciales distintas.
2) Las instancias con hasta 500 vértices de la página:
http://mat.gsia.cmu.edu/COLOR/instances.html

Luego hará un nuevo algoritmo, que llamaremos DSATUR*, que resulta de una pequeña modificación al criterio (c) de DSATUR+, que es la siguiente:

> (c´) Los vecinos de un nodo válido del árbol de "backtracking" deberán ser considerados en el siguiente orden: asignar primero al vértice v a colorear, el color, aún no considerado, que tenga menos vértices coloreados de ese color.

RESULTADOS:

En el informe del proyecto deberá incluir, aparte de lo señalado en las normas de presentación del informe:

- Demostrar formalmente que en efecto el algoritmo implícito dado determina una coloración mínima.
- Los resultados de los tres algoritmos (Brelaz+interchangeProcedure, DSATUR+ y DSATUR*) los darán como en las tablas 2 y 3 del artículo de Sewell [4]. Si procede, es decir, si el algoritmo de enumeración implícita termina, se deberá indicar por cada grupo (n,d) para cuántos grafos en el grupo el algoritmo de Brelaz+interchange da la solución óptima.
- Las normas de presentación del informe y las reglas de documentación del código están en http://www.ldc.usb.ve/~meza/ci-5651/e-a2010/ . Es obligatorio documentar bien el código.
- Debe programar en C en ambiente UNIX.

AYUDA: utilice la implementación iterativa del "backtracking" dada en la sección 2 de [2].

Comentarios finales:

- Para que tomen el tiempo con la mayor precisión posible, está una función "gettimeofday " de C que obtiene el tiempo, aquí está un ejemplo de su uso:

```
#include <sys/time.h>
struct timeval t_p;
…..
  if (!gettimeofday(&t_p,NULL))
     TIEMPOINICIAL = (double) t_p.tv_sec + ((double) t_p.tv_usec)/1000000.0;
  else printf("\n mal tiempo \n");

.....llamada al programa.....
```

```
o     if (!gettimeofday(&t_p,NULL))
o        TIEMPOFINAL = (double) t_p.tv_sec + ((double) t_p.tv_usec)/1000000.0;
o     else printf("\n mal tiempo \n");
o  printf("tiempo EN SEGUNDOS de ejecución del programa: #1.4f",
     TIEMPOINICIAL – TIEMPOFINAL);
```

- **IMPORTANTE:** EL INFORME DEL PROYECTO DEBE SER ENTREGADO EN UN ARCHIVO .PDF ESCRITO CON UN PROCESADOR DE TEXTO (ES DECIR, NO A MANO). ME DEBEN ENTREGAR EL INFORME EN PAPEL Y ENVIARME POR E-MAIL EL ARCHIVO PDF EL DÍA DE LA ENTREGA, AL IGUAL QUE UN ARCHIVO CON EL CODIGO FUENTE, EL MAKEFILE Y LAS INSTANCIAS DE GRAFOS DE MANERA DE YO COMPILARLO Y EJECUTARLO.

- FECHA DE ENTREGA: 31 de marzo

Bibliografía (que podrá encontrar en la página web):

[1] Jürgen Peemöller. “A correction to the Brelaz’s modification of Brown’s coloring algorithm” communication of the ACM. Vol. 26. Número 8, agosto 1983, pp. 595-597.

[2] Kubale, Jackowski. “A generalized implicit enumeration algorithm for graph coloring”. Communications of the ACM. Vol. 28. Número 4. Abril, 1985, pp. 167-176.

[3] Korman. “The graph-coloring problema”. En Combinatorial Optimization. N. Christofides, A. Mingozzi, P. Toth, y C. Sandi. Eds., Wiley, New York, 1979, pp. 211-235.

[4] Sewell. “An improved algorithm for exact graph coloring”. DIMACS series in Discrte Mathematics and Theoretical computer sciences. Vol. 26. 1996. Pp. 359-372

[5] Brélaz. “New methods to color the vértices of a graph”. Communications of the ACM 22, 4 (abril. 1979), pp. 251-256