UNIVERSIDAD SIMÓN BOLÍVAR

Departamento de Computación y Tecnología de la Información

CI-5651 - Diseño de Algoritmos I

Profesor: Guillermo Palma

PROYECTO 2

Elaborado Por:

Salvador Di Renzo 10-10296
Stefani Castellanos 11-11394

Sartenejas, Marzo 2017

CORRECCIONES EN EL ALGORITMO *GREEDY*

Antes de realizar el algoritmo *Branch and Bound* fue necesario corregir la solución ofrecida por el algoritmo *greedy* del proyecto 1, ya que el beneficio obtenía valores mejor que el óptimo lo cual no es posible. Dicho error consistía en la falta de una condición que evitara sumar el beneficio cuando no se cruza por una arista determinada, es decir, cuando las veces que atravesaba una arista era 0.

Adicionalmente se decidió incorporar la creación de los ciclos eulerianos lo cual permitió mejorar algunas soluciones, sin embargo, se obtendrían muchos mejores resultados si se eliminaran los ciclos negativos de estas soluciones. A continuación se muestran los nuevos resultados obtenidos:

ALBAIDA

Instancia	Vo	%dHeurPlanos	%dHeur	tHeur (seg)
ALBAIDAA	6266	0.303224	109.783	0
ALBAIDAB	4372	0	62.8317	0

CHRISTOFIDES

Instancia	Vo	%dHeurPlanos	%dHeur	tHeur (seg)
P1	3	0	666.667	0
P2	66	0	157.576	0
P3	56	0	89.2857	0
P4	45	0	108.889	0
P5	35	0	45.7143	0
P6	60	0	65	0
P7	89	0	71.9101	0

P8	90	0	44.4444	0
P9	46	2.17391	86.9565	0
P10	41	0	87.8049	0
P11	9	0	44.4444	0
P12	10	0	20	0
P13	5	0	120	0
P14	128	0	60.1562	0
P15	43	9.30233	41.8605	0
P16	113	6.19469	107965	0
P17	42	7.14286	126.19	0
P18	21	19.0476	109.524	0
P19	90	0	26.6667	0
P20	246	0	75.2033	0
P21	258	2.32558	73.6434	0
P22	474	0	93.2489	0
P23	360	0	147.222	0
P24	237	2.95359	91.5612	0

DEGREE

Instancia	Vo	%dHeurPlanos	%dHeur	tHeur (seg)
D0	109	0	1025.69	0
D1	115	0	1700	0
D2	274	0	261.679	0

D3	172	0	491.279	0
D4	210	0	392.857	0
D5	313	0	161.342	0
D6	166	0	47.5904	0
D7	260	0	397.692	0
D8	457	0	7.43983	0
D9	160	0	962.5	0
D10	0	0	0	0
D11	398	0	140.704	0
D12	280	0	650	0
D13	717	7.67085	142.817	0
D14	810	0	53.3333	0
D15	702	0	174.501	0
D16	980	1.53061	82.551	0
D17	1115	4.03587	137.22	0
D18	515	0	80.9709	0
D19	509	0	67.9764	0
D20	457	0	396.937	0
D21	1000	2.4	90.4	0
D22	989	0	101.011	0
D23	968	0	150.62	0
D24	1463	0	45.2495	0
D25	1317	0	74.5634	0

D26	1625	0.923077	77.7846	0
D27	549	8.74317	102.368	0
D28	815	3.55828	352.761	0
D29	55	-840	3089.09	0
D30	1378	0	116.473	0
D31	1503	0	158.55	0
D32	1066	3.56473	138.368	0
D33	2074	0	82.6905	0
D34	1806	0	94.2414	0
D35	1901	4.26092	64.9658	0

GRID

Instancia	Vo	%dHeurPlanos	%dHeur	tHeur (seg)
G0	0	0	0	0
G1	0	0	0	0
G2	0	0	0	0
G3	2	0	200	0
G4	0	0	0	0
G5	4	0	75	0
G6	9	0	88.8889	0
G7	1	0	600	0
G8	4	0	225	0

G9	2	0	1550	0
G10	0	0	0	0
G11	4	0	425	0
G12	15	0	213.333	0
G13	11	0	472.727	0
G14	14	0	407.143	0
G15	26	0	69.2308	0
G16	20	0	230	0
G17	24	0	75	0
G18	8	0	950	0
G19	6	0	1400	0
G20	9	0	977.778	0
G21	32	0	309.375	0
G22	32	3125	84375	0
G23	33	0	130.303	0
G24	57	0	94.7368	0
G25	46	0	80.4348	0
G26	57	0	82.4561	1
G27	14	0	550	0
G28	23	4.34783	804348	0
G29	14	7.14286	271429	0
G30	50	0	134	0
G31	54	0	142.593	0

G32	57	3.50877	273.684	0
G33	92	0	130.435	0
G34	86	0	61.6279	0
G35	88	1.13636	86.3636	0

RANDOM

Instancia	Vo	%dHeurPlanos	%dHeur	tHeur (seg)
R0	1742	0	1603.73	0
R1	4253	0	172.49	0
R2	5638	0	470.539	0
R3	18453	0	160.51	0
R4	17316	0	305.244	0
R5	398	0	3014.32	0
R6	12478	0	83.1463	0
R7	9405	0	189.282	0
R8	14847	0	352.367	0
R9	17405	0	474.479	0
R10	7125	0	563.916	0
R11	1493	0	1112.73	0
R12	32453	6.12886	170.739	0
R13	30732	0	363.735	0
R14	27791	0	96.4557	0
R15	10533	0	86.3856	0

R16	4276	6.43124	245.229	0
R17	28462	0	195.376	0
R18	26873	0	250.776	0
R19	0	0	0	0

ALGORITMO BRANCH AND BOUND

El algoritmo fue implementado siguiendo el pseudocódigo presentado en el enunciado del proyecto, exceptuando la función "repite-ciclo", sin embargo, no implementar esta función no afecta los resultados finales sino el tiempo que toma encontrar una solución.

El grafo fue modelado como uno dirigido, dado que se necesitaba hacer *Depth First Search* para buscar la solución, y es más costoso y complicado de implementar en un grafo no dirigido. Cada arista en la lista de adyacencias está conformada por el vértice al que llega, el beneficio y el costo.

- obtener-lista-adyacencias: esta función se realizó obteniendo todas las aristas que salen de v, repitiendo las aristas con beneficio 0 y el original de la arista.

- agregar-lado: dada una arista e(v, v') se agrega la arista v' al final de la lista de adyacencias de v.

- eliminar-lado: se recorre el camino actual y se determina cual es el último vértice de la lista y se procede a eliminarlo.

- esta-lado: permite determinar si una arista (v, v') está o no en la solución. Se recorre la lista de adyacencias del vértice v y se busca la arista que se desea agregar, si la encuentra se incrementa el contador. Este procedimiento se realiza de igual manera para v', y se busca por el vértice original v. Si el contador resulta en 1 y el beneficio de la arista a agregar es 0 se puede agregar, de lo contrario no se puede.

- beneficio: el cálculo del beneficio siguiendo el camino y según lo establecido por la definición del problema.

- cumple-acotamiento: según lo especificado en el enunciado del problema tomando como beneficio disponible la solución óptima conocida.

- ciclo-negativo: se recorre la solución parcial, si el vértice al que llega la arista que se desea agregar forma parte de la solución parcial, se determina el beneficio y si es negativo la función devuelve verdadero.

- repite-ciclo: como fue explicado anteriormente dicha función no fue implementada, sin embargo podemos establecer el bosquejo del algoritmo que lo calcularía: dado un vértice v que pertenece a la arista que se desea agregar se busca en la solución parcial y entre sus adyacentes cual es la arista que comienza el ciclo que se pretende cerrar. Si cumple la condición especificada en el enunciado se devuelve verdadero.

Desafortunadamente, el código presenta un error en el que se poda alguna rama que podría mejorar la solución. En algunos casos se logra la solución óptima pero en otros no.

Las corridas fueron realizadas en una computadora portátil con procesador Intel Core 2 Duo y 4GB de memoria RAM con 7 años de antigüedad. No se lograron obtener todos los resultados, dado que dicha computadora presenta fallas y si el algoritmo tarda mucho tiempo, se reinicia. Por esta razón las corridas se realizaron por un máximo de 30 minutos.

CHRISTOFIDES

Nombre de la Instancia	Valor óptimo	Valor de la heurística (*greedy)*	% de desviación (*greedy)*	Tiempo del algoritmo *Branch and Bound*
P01	3	-17	666.667	0
P02	66	-38	157.576	0
P03	56	6	89.2857	-T-
P04	45	-4	108.889	-T-
P05	35	19	45.7143	1
P06	60	21	65	-T-
P07	89	25	71.9101	557

P08	90	50	44.4444	274
P09	46	6	86.9565	147
P10	41	5	87.8049	0
P11	9	5	44.4444	0
P12	10	8	20	0
P13	5	-1	120	0
P14	128	51	60.1562	-T-
P15	43	25	41.8605	1
P16	113	-9	107965	-T-
P17	42	-11	126.19	103
P18	17	-2	111765	1
P19	90	66	26.6667	39
P20	246	61	75.2033	-T-

DEGREE

Nombre de la Instancia	Valor óptimo	Valor de la heurística (*greedy)*	% de desviación	Tiempo del algoritmo *Branch and Bound*
D0	109	-1009	1025.69	0
D1	115	-1840	1700	0
D2	274	-443	261.679	0
D3	172	-673	491.279	0
D4	210	-615	392.857	1
D5	313	-192	161.342	0

D6	166	87	47.5904	0
D7	260	-774	397.692	2
D8	457	423	7.43983	0
D9	160	-1380	962.5	35
D10	0	-114	0	0
D11	398	-162	140.704	2
D12	280	-1540	650	72
D13	662	-307	146.375	-T-
D14	810	378	53.3333	121

GRID

Nombre de la Instancia	Valor óptimo	Valor de la heurística (*greedy)*	% de desviación (*greedy*)	Tiempo del algoritmo *Branch and Bound*
G0	0	-6	-	0
G1	0	-3	-	0
G2	0	0	0	0
G3	2	-2	200	0
G4	0	-2	-	0
G5	4	1	75	0
G6	9	1	88.8889	0
G7	1	-5	600	0
G8	4	-5	225	0

G10	0	-74	-	0
G12	15	-17	213.333	-T-
G13	11	-41	472.727	-T-
G14	14	-43	407.143	-T-
G15	26	8	69.2308	-T-

RANDOM

Nombre de la Instancia	Valor óptimo	Valor de la heurística (*greedy)*	% de desviación (*greedy*)	Tiempo del algoritmo *Branch and Bound*
R1	4253	-3083	180.016	0
R2	5638	-20891	470.539	0
R3	18453	-11166	160.51	7
R4	17316	-35540	305.244	0
R6	12478	2103	83.1463	24
R7	9405	-8397	189.282	0
R8	14847	-37469	352.367	-T-
R9	17523	-65178	471.957	-T-
R10	17405	-33054	289.911	-T-

Generalmente los casos con mayor cantidad de vértices son difíciles de resolver en el tiempo estipulado para cualquiera de los tipos de problemas, sin embargos algunos si obtienen las solución. Es importante destacar que, dado que el algoritmo Greedy podría obtener beneficios negativos, al inicializar el *Branch and Bound* prácticamente se ignora y se toma como mejor solución no salir del vértice d.

Los resultados que no se encuentran en las tablas indican que el beneficio del mejor camino se encontraban muy lejos del óptimo.