B&B

andrea

4 de enero de 2013

1. Branch and Bound

Para la variables binarias que dan soporte a las restricciones relacionadas con los valores absolutos de las distancias y la función objetivo y que la ciudad más cercana corresponda a una ciudad del problema. Para ver mayor información revisar la sección **Variables Binarias del Problema**.

Para el algoritmo *Branch and Bound* (**B&B**) usamos la implementación en la librería *LpSolve*. Dicha librería trae varias heurísticas con las cuales usar el B&B, además de las heurísticas para elegir la rama a tomar, también esta disponible la opción de si iniciar la rama con la función techo o con la función piso.

De las heurísticas que están disponibles en *LpSolve* elegimos 13 para comparar el rendimiento. Además por cada heurística ésta se probó con la función techo y la función piso, para evaluar la mejor combinación. A continuación se explica el proceder de las 13 heurísticas[1]:

1. **NODE_FIRSTSELECT**: Elige la primera columna no entera.
2. **NODE_GAPSELECT**: Selecciona la variable de acuerdo a la distancia de los límites actuales.
3. **NODE_RANGESELECT**: Selecciona la variable de acuerdo al mayor límite actual.
4. **NODE_FRACTIONSELECT**: Selecciona la variable que tenga el valor fraccionario más grande.
5. **NODE_PSEUDOCOSTSELECT**: Selecciona la variable con la estrategia pseudo-costo (búsqueda costo uniforme)
6. **NODE_PSEUDONONINTSELECT**: Es una extensión de la estrategia pseudo-costo basada en minimizar el número de enteros fallidos.
7. **NODE_PSEUDORATIOSELECT**: También es una extensión de la estrategia pseudo-costo basada en maximizar la razón pseudo-cost divido por el número de fallidas. Similar a costo/beneficio.
8. **NODE_WEIGHTREVERSEMODE**: Selecciona la variable por el peor criterio en vez del mejor criterio.
9. **NODE_GREEDYMODE**: Búsqueda informada, con un costo heurístico.
10. **NODE_DEPTHFIRSTMODE**: Búsqueda en profundidad, selecciona el nodo por el que ya venia explorando.
11. **NODE_RANDOMIZEMODE**: Añade un factor random al costo de un nodo candidato.
12. **NODE_BREADTHFIRSTMODE**: Selecciona el nodo que no se halla seleccionado o que se halla seleccionado menos veces (anteriormente).
13. **NODE_AUTOORDER**: Crea una variable de ordenamiento "óptima" para el B&B. (Indexación)

Para realizar la pruebas correspondientes se generaron 20 ejemplos aleatorios. 5 ejemplos con una grilla de tamaño 5 y número de ciudades aleatorio, 5 ejemplos con una grilla de tamaño 10 y número de ciudades aleatorio, 5 ejemplo con una grilla de tamaño 15 y número de ciudades aleatorio y 5 ejemplos con una grilla de tamaño 20 y número de ciudades aleatorio. A cada ejemplo se le aplica el algoritmo con cada heurística una vez

[1] fuente: http://lpsolve.sourceforge.net/5.5/set_bb_rule.htm

con la función techo y otra con la función piso.

Comparamos el desempeño de las heurísticas usando la función techo con los 5 ejemplos de un tamaño de grilla y así con todos los tamaños de la grilla. Se comparaban las heurísticas en 3 tres grupos, por cada grupo se seleccionaba la heurística con el mejor desempeño, y al final se eligió la mejor de las tres mejores. Este fue el procedimiento aplicado para las comparaciones por ejemplos. Finalmente se realizó una comparación entre todos los ejemplos (se eligió el un ejemplo por cada tamaño de grilla, el que tuviera mas ciudades) para elegir las 3 mejores heurísticas, dichas heurísticas usaban la función piso.

1.1. Ejemplos tamaño de grilla 5

Gráficas correspondientes al desempeño de las heurísticas, 3 grupos:
Usando techo:

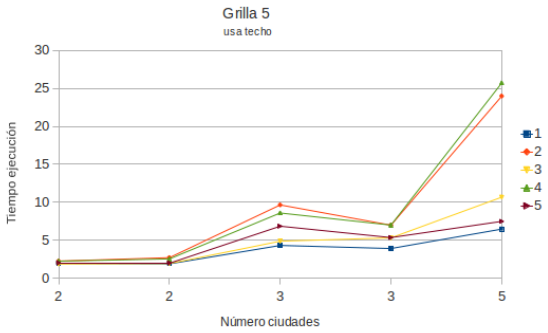

Figura 1: Comparación heurísticas $1 \cdots 5$

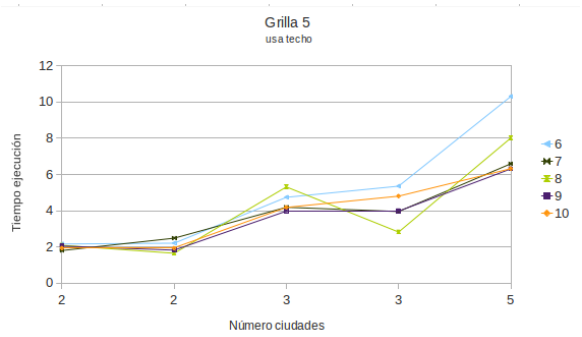

Figura 2: Comparación heurísticas $6 \cdots 10$

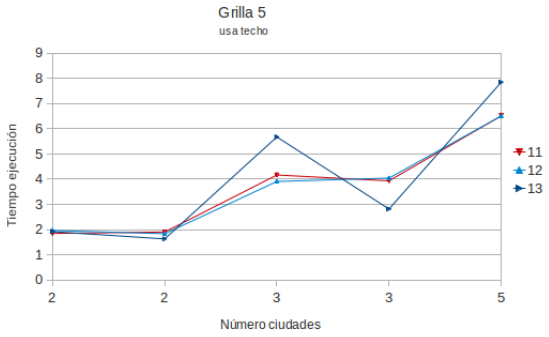

Figura 3: Comparación heurísticas $11 \cdots 13$

De la figura 1 la heurística que presentó mejor desempeño fue la número 1 (NODE_FIRSTSELECT), de la figura 2 fue la número 9 (NODE_GREEDYMODE) y de la figura 3 fue la número 12 (NODE_BREADTHFIRSTMODE). Al comparar éstas tres, la mejor correspondió a la heurística número 9 (NODE_GREEDYMODE).

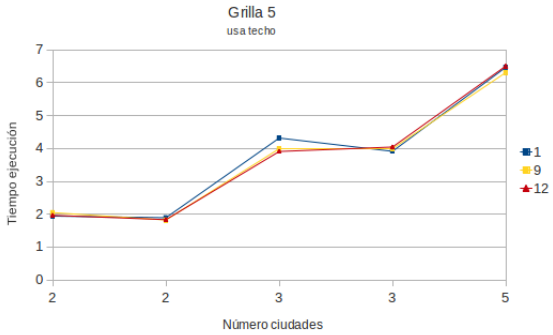

Figura 4: Comparación mejor heurística

Usando piso:

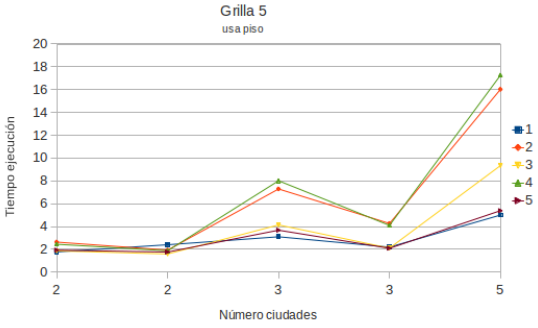

Figura 5: Comparación heurísticas $1 \cdots 5$

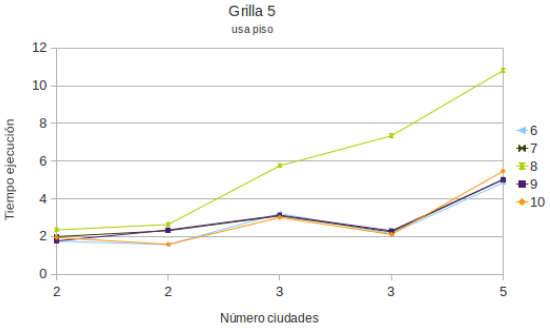

Figura 6: Comparación heurísticas $6 \cdots 10$

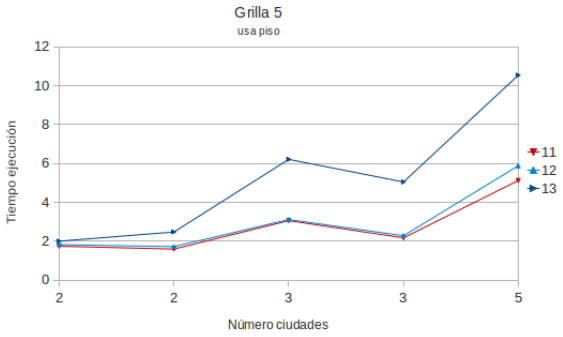

Figura 7: Comparación heurísticas $11 \cdots 13$

De la figura 5 la heurística que presentó mejor desempeño fue la número 1 (NODE_FIRSTSELECT), de la figura 6 fue la número 6 (NODE_PSEUDONONINTSELECT) y de la figura 7 fue la número 11 (NODE_RANDOMIZEMODE) Al comparar éstas tres, la mejor correspondió a la heurística número 6 (NODE_PSEUDONONINTSELECT).

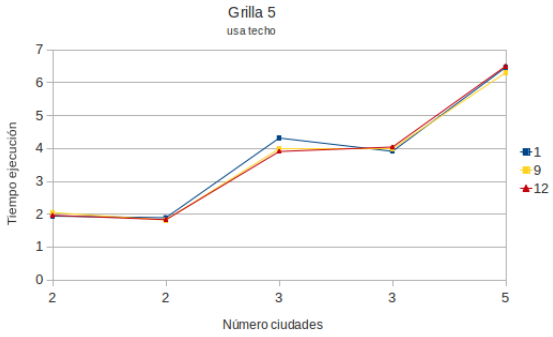

Figura 8: Comparación mejor heurística

6 PSEUDONONINTSELECT GREEDY	9
1,753	2,05
1,573	1,824
3,223	3,993
2,154	3,998
4,858	6,313

Comparación mejor techo-piso

Tomamos la mejor heurística usando techo y la mejor heurística usando piso, comparamos sus resultados con los ejemplos y concluimos que la mejor era la número 6, que usaba piso.

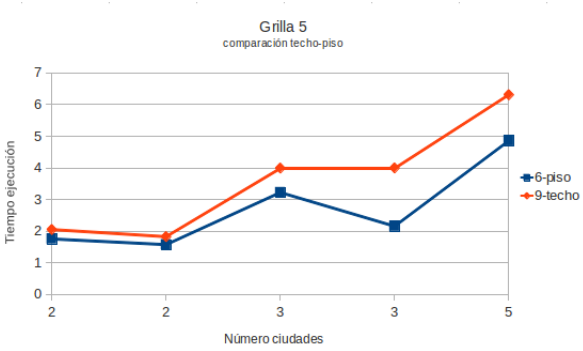

Figura 9: Comparación mejores techo y piso