Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Industrial

Algoritmos modernos para producción

Pauta para el proyecto

Profesores: Lorena Pradenas – Rosa Medina – Carlos Contreras

Fecha: 27 de Septiembre de 2022

1. Enunciado del proyecto

El proyecto consiste en escoger un paper de la lista adjunta de papers (revisar TEAMS del curso y además inscribirse). Se debe trabajar con el problema del paper escogido e implementar una metaheurística (de las vistas en clases) diferente a la propuesta en el paper (en caso de que el paper escogido ya presente una metaheurística). Al final del semestre se debe realizar una presentación, exponiendo el problema escogido, su modelamiento con la metaheurística, resultados de la metaheurística propuesta con las instancias disponibles para el problema escogido y finalmente, las conclusiones. Adicionalmente, se debe preparar un mini informe.

2. Formato para el informe

Se debe usar el formato presentado en el Anexo 5 y debe ser presentando en formato `pdf`. El tipo de fuente es Times New Roman, tamaño 12 con espaciado de 1,5. Los pseudo-códigos del documento son opcionales. Las secciones dependerán de la metaheurística utilizada. Extensión máxima 5 páginas sin contar bibliografía. En caso de usar LaTeX, usar el tipo de fuente estándar y documento tipo `article`.

3. Evaluación de la exposición

Tiempo de presenytación son 10 minutos, se descuenta por uso adicional de tiempo. Se requiere usar lenguaje técnico y apropiado para la ocasión y en la exposición se avalúa:

- Introducción: presentación breve y clara del trabajo original (1,0 punto).
- Diseño: Exposición clara y precisa del diseño de la metaheurística para el problema (2,5 puntos).
- Resultados: Exposición clara y precisa de los resultados obtenidos (1,0 punto).
- Conclusiones (0,5 ptos.).
- Respuestas claras y precisas a las preguntas de los profesores. (1,0 punto).

Notar que la nota está sujeta a la complejidad del problema abordado.

4. Instrucciones

- El proyecto es individual.

- El tema debe ser inscrito en un archivo word en TEAMS. Un paper solo puede estar asociado a solo un estudiante.
- Para la implementación de la metaheurística se recomienda el uso de Python 3.x. En caso de utilizar otro lenguaje, pedir la autorización de los profesores.
- Se enviar un correo un archivo comprimido con el código fuente del proyecto y un informe a TEAMS hasta el día antes del día de la presentación, por syllabus 22 de noviembre de 2022. El correo debe tener el siguiente asunto: Metaheurísticas-2022-02: Proyecto. Además, debe tener copia a los tres profesores: lpradena@udec.cl, rosmedina@udec.cl, carlos.contreras.b@udec.cl.
- El archivo comprimido final enviado debe ser un archivo comprimido en zip o rar que contenga una carpeta con el siguiente formato:

  `ApellidoPaterno_ApellidoMaterno`
  Ejemplo de carpeta: Juan Pérez Valdivia → `Pérez_Valdivia`
  Archivo comprimido final: `Pérez_Valdivia.pdf`

- En caso de detectarse copia o plagio tendrán la nota mínima sin apelación.
- En caso de usar códigos o ideas de otros trabajos, agregar las referencias respectivas.
- En caso de nos seguir alguna de las condiciones o instrucciones será penalizado con 1.0 punto por cada infracción.

5. Anexo

Proyecto

Jorge Dantzig
Departamento de Ingeniería Industrial, Universidad de Concepción

1. Metaheurística Propuesta

Se propone un Iterated Local Search (ILS) que se presenta en el Algoritmo 1 que contiene cuatro componentes: solución inicial (línea 1); una estrategia de perturbación (línea 5); dos estrategias de búsqueda local (línea 2 y en las líneas 6–10) y finalmente un criterio de aceptación (línea 11-12). El algoritmo contempla una solución actual y la mejor solución encontrada, s y s^*, respectivamente. En la siguientes subsecciones se presentan con más detalles la representación, función objetivo, solución inicial, estrategia de perturbación, estrategias de búsqueda local, criterio de aceptación y parámetros.

Algoritmo 1 Iterated Local Search

1: $s \leftarrow$ `heurística-vecino-más-cercano()`
2: $s \leftarrow$ `2-opt(`s`)`
3: $s^* \leftarrow s$
4: **repeat**
5: $s \leftarrow$ `intercambio-aleatorio(`s`)`
6: **if** `rand`$(0,1) > \alpha$ **then**
7: $s \leftarrow$ `mejora-intercambio-aleatorio(`s`)`
8: **else**
9: $s \leftarrow$ `2-opt(`s`)`
10: **end if**
11: $s^* \leftarrow$ si $f(s) < f(s^*)$
12: $s \leftarrow$ `criterio-aceptación(`$f(s) \leq f(s^*) \times 1{,}05 : s, s^*$`)`
13: **until** número de iteraciones

1.1. Representación

La represetanción usada es la de permutación, donde cada índice corresponde a una ciudad única, $s = \{1, \ldots, n\}$.

1.2. Función objetivo

Se utiliza la suma de las distancias euclidianas en cada ciudad como en la ecuación (1).

$$f(s) = \sum_{i=1}^{n-1} \texttt{distancia}(s_i, s_{i+1}) + \texttt{distancia}(s_n, s_1) \tag{1}$$

1.3. Solución inicial

Es utilizada la heurística del vecino más cercano aleatoria (Flood, 1956). Esta consiste en escoger de manera aleatoria una ciudad i y luego escoger la ciudad más cercana a la ciudad i, luego i pasa a ser la nueva ciudad escogida. Repetir el procedimiento de las ciudadades más cercanas hasta que no hayan más ciudades disponibles. El procedimiento anterior se presenta en el Algoritmo 2.

Algoritmo 2 heurística-vecino-más-cercano

Entrada: V: ciudades, c_{ij}: pesos de las aristas.
Salida: Ruta

1: $S \leftarrow \emptyset$
2: $i \leftarrow$ escoger aleatoriamente una ciudad.
3: $S \leftarrow S \cup \{i\}$
4: $V \leftarrow V \setminus \{i\}$
5: **while** $V \neq \emptyset$ **do**
6: $\quad k \leftarrow \text{mín}\{c_{ij} : j \in V\}$
7: $\quad S \leftarrow S \cup \{k\}$
8: $\quad V \leftarrow V \setminus \{k\}$
9: $\quad i \leftarrow k$
10: **end while**

1.4. Estrategia de Perturbación

Se determinan dos enfoques para la estrategia de perturbación. El primero consiste en escoger una posición i aleatoriamente e intercambiar la ciudad de esa posición con la ciudad de la posición siguiente $(i+1)$ de la ruta como se presenta en el Algoritmo 3.

Algoritmo 3 intercambio-aleatorio

Entrada: $s = \{v_1, v_2, \ldots, v_n\}$ y c_{ij}.
Salida: s'

1: escoger aleatoriamente una posición i.
2: $s' \leftarrow$ nueva ruta considerando el intercambio las ciudades de las posiciones i y $i+1$.

1.5. Estrategia de Búsqueda local

En esta estrategia se utilizan dos enfoques. La primera es la búsqueda local 2-opt (Morton and Land, 1955; Croes, 1958), la cual consiste en recorrer iterativamenete pares de arista e intercambiarlos. En caso de que con el intercambio la ruta mejore su costo, entonces, se mantiene, caso contrario se deshace. En particular, en esta implementación se utiliza la versión que si se encuentra una mejora el algoritmo se detiene. El algoritmo antes descrito se presentan en el Algoritmo 4.

Algoritmo 4 2-opt

Entrada: $s = \{v_1, v_2, \ldots, v_n\}$ y c_{ij}.
Salida: s'

1: **for** $i \leftarrow 1$ hasta $n-2$ **do**
2: $\quad$ **for** $j \leftarrow i+1$ hasta $n-1$ **do**
3: $\quad\quad$ **if** $c_{v_i v_{i+1}} + c_{v_j v_{j+1}} > c_{v_i v_{i+1}} + c_{v_{i+1} v_{j+1}}$ **then**
4: $\quad\quad\quad$ Se desconecta (v_i, v_{i+1}) y (v_j, v_{j+1}).
5: $\quad\quad\quad$ Se remplaza por (v_i, v_j) y (v_{i+1}, v_{j+1}).
6: $\quad\quad\quad$ Se actualiza el costo de la solución actual.
7: $\quad\quad\quad$ Se termina el algoritmo.
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**

El segundo enfoque utilizado es una heurística de mejora que consiste en intercambiar ciudades aleatoriamente 30 veces y cuando exista mejora se mantiene. Este enfoque se detalla en e Algoritmo 5.

Algoritmo 5 búsqueda local con un intercambio-aleatorio

Entrada: $s = \{v_1, v_2, \ldots, v_n\}$.
Salida: s'

1: **repeat**
2: escoger aleatoriamente una posición i de s.
3: escoger aleatoriamente una posición j de s.
4: $s \leftarrow$ si al intercambiar las ciudades de las posiciones i y j mejora.
5: **until** 30 veces

1.6. Criterio de aceptación

Si la solución es menor o igual a un 5 % desviada de la mejor solución encontrada, entonces aceptar como solución actual, caso contrario usar la mejor encontrada.

1.7. Parámetros

El número de iteraciones se define en 500 iteraciones y además α es 0,8.

2. Resultados computacionales

El algoritmo propuesto es implementado en Python 3.7. Las pruebas s realizaron en un Intel(R) Core(TM) i7-7500U con 2.7 GHz y 16 GB de RAM (ejecutado en un solo un hilo), en el sistema operativo GNU/Lunux Ubuntu 20.04. El algoritmo propuesto se ejecutó 10 veces con cada instancia. El código de la implementación se encuentra disponible en `https://bit.ly/37ooz4a`. En la Tabla 1 se presentan los resultados con cuatro instancias con un tamaño de entre 17 a 52 vértices en la segunda columna. En la tercera columna, el tamaño de las aristas está entre 136 a 1326. En la cuarta columna los costos óptimos. La quinta columa es el costo mínimo obtenido de las 10 ejecuciones por el algoritmo propuesto. En la sexta columna se encuentra el error relativo mínimo (ERP), que es calculado como: ((costo mínimo obtenido - costo óptimo) / costo óptimo) $\times 100\,\%$. La séptima columna corresponde al costo promedio de las 10 ejecuciones. En la octava columna se encuentra el error relativo promedio (ERM) análogo al ERP pero con el costo promedio. Finalmente en la última columna se encuentra el tiempo promedio en segundos.

Tabla 1: Resultados de ILS en las cuatro instancias.

instancia	$\|V\|$	$\|A\|$	costo óptimo	mínimo	ERP	promedio	ERM	tiempo
`gr17`	17	136	5047	5047	0,00	5047,00	0,00	0,32
`gr24`	24	276	1272	1272	0,00	1274,36	0,19	0,39
`dantzig42`	24	8614	699	701	0,29	705,10	0,87	7,23
`berlin52`	52	1326	7542	7620	1,03	7627,99	1,14	10,2
promedio	29,25	2588,00	3640,00	3660,00	0,33	3663,61	0,55	4,54

En la Tabla 1 se observa que la instancia `berlin52` es la que más tiempo computacional necesita, 10,2 segundos y en la que peor rendimiento tiene el algortimo propuesto. El algoritmo encuentra dos óptimos cuando se considera el mínimo costo entre las 10 ejecuciones. Solo un óptimo cuando se considera el promedio de las 10 ejecuciones. El rendimiento general del algoritmo propuesto es de 0,33 y $0,55$, el error relativo promedio y mínimo respectivamente. Mientras el tiempo promedio es 4,54 segundos.

Referencias

Croes, G. A. (1958). A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6(6):791–812.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, 4(1):61–75.

Morton, G. and Land, A. H. (1955). A contribution to the 'travelling-salesman'problem. *Journal of the Royal Statistical Society: Series B*, 17(2):185–194.