3. Técnicas heurísticas constructivas

- Son métodos iterativos deterministas
- En cada iteración, manejan una solución parcial al problema
- La solución final se va construyendo, añadiendo en cada iteración un elemento a la solución parcial
- La decisión tomada en cada iteración sobre qué elemento añadir, es determinante en el resultado final
- El resultado final también suele depender de la solución parcial de partida o semilla
- El método acaba cuando la solución al problema se ha completado
- Suelen ser rápidos, por ello en la práctica es habitual aplicarlos distintas veces a un mismo problema y obtener diferentes soluciones, para finalmente, quedarnos con la mejor.

38

38

- Ejemplos
  - Vecino más cercano - TSP
  - Inserción - TSP
  - Clarke y Wright - TSP, VRP
  - Reglas de prioridad, Serie, Paralelo - PPRL
  - …

39

39

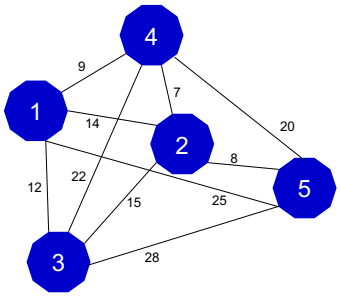
Algoritmo del vecino más cercano para el TSP

- Se trata de uno de los algoritmos más sencillos para el TSP
- La idea es visitar, a partir de una ciudad, siempre la ciudad más cercana
- Su complejidad es $O(n^2)$, siendo n el número de ciudades
- Pasos
  1. Seleccionar una ciudad aleatoriamente como ciudad origen en la que nos encontramos
  2. Encontrar, de entre las ciudades no visitadas, la más cercana a la que nos encontramos y moverse a ella
  3. ¿Se han visitado todas las ciudades? Si no es así, volver a 2
  4. Volver a la ciudad elegida en el paso 1
- El resultado cambia en función del nodo elegido en el paso 1, por lo que se puede repetir el algoritmo eligiendo cada vez un nodo diferente de partida y quedándonos al final con la mejor ruta

40

■40

- Ejercicio



Dado el grafo anterior, donde los valores representan el coste de transporte entre ambos nodos, obtener:

1.- Diferentes rutas mediante el algoritmo del V+C

41

■41

Algoritmo de inserción para el TSP

- Se trata de un tipo de algoritmo del que existen diferentes variantes
- La idea es partir de un ciclo que no visita todas las ciudades e ir añadiendo de forma iterativa ciudades a dicho ciclo hasta que se forma un ciclo que las visita todas
- El ciclo inicial puede estar formado por un único nodo
- Existen diferentes variantes, dependiendo de la forma en la que se elige la ciudad a añadir al ciclo y cómo se conecta ésta con las demás para formar el ciclo
- La complejidad de este algoritmo es de $O(n^3)$

42

■42

0) Variante Inserción del más cercano

- Pasos
  1. Establecer una ciudad i como origen
  2. Encontrar el nodo k más cercano al origen y establecer el subciclo i-k-i
  3. Paso selección: Encontrar el nodo k, no perteneciente al subciclo actual, más cercano a cualquiera de los nodos del subciclo actual
  4. Paso inserción: Encontrar el arco (i,j) en el subciclo actual que minimiza $d_{ik}+d_{kj}-d_{ij}$. Insertar k entre i y j.
  5. Si no todas las ciudades están incluidas en el ciclo actual ir a 3.

43

■43

1) Variante Inserción del más lejano

- Pasos
  1. Establecer una ciudad i como origen
  2. Encontrar el nodo k más lejano al origen y establecer el subciclo i-k-i
  3. Paso selección: Encontrar el nodo k, no perteneciente al subciclo actual, más lejano a cualquiera de los nodos del subciclo actual
  4. Paso inserción: Encontrar el arco (i,j) en el subciclo actual que minimiza $d_{ik}+d_{kj}-d_{ij}$. Insertar k entre i y j.
  5. Si no todas las ciudades están incluidas en el ciclo actual ir a 3.

44

■44

2) Variante Inserción aleatoria

- Pasos
  1. Establecer una ciudad i como origen
  2. Encontrar el nodo k más cercano al origen y establecer el subciclo i-k-i
  3. Paso selección: Seleccionar aleatoriamente un nodo k no perteneciente al subciclo actual
  4. Paso inserción: Encontrar el arco (i,j) en el subciclo actual que minimiza $d_{ik}+d_{kj}-d_{ij}$. Insertar k entre i y j.
  5. Si no todas las ciudades están incluidas en el ciclo actual ir a 3.

45

■45

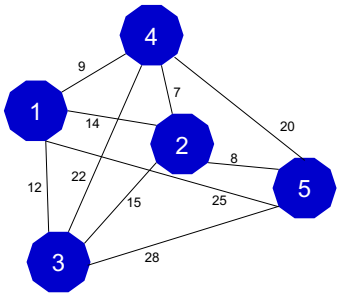
3) Variante Inserción más barata

- Pasos
  1. Establecer una ciudad i como origen
  2. Encontrar el nodo k más cercano al origen y establecer el subciclo i-k-i
  3. Paso inserción: Encontrar el arco (i,j) en el subciclo actual y k no perteneciente al mismo que minimiza $d_{ik}+d_{kj}-d_{ij}$. Insertar k entre i y j.
  4. Si no todas las ciudades están incluidas en el ciclo actual ir a 3.

46

■46

- Ejercicio



Dado el grafo anterior, donde los valores representan el coste de transporte entre ambos nodos, obtener:

1.- Una ruta mediante el algoritmo de inserción del más cercano

2.- Una ruta mediante el algoritmo de inserción del más lejano

¿Cuál es mejor?

47

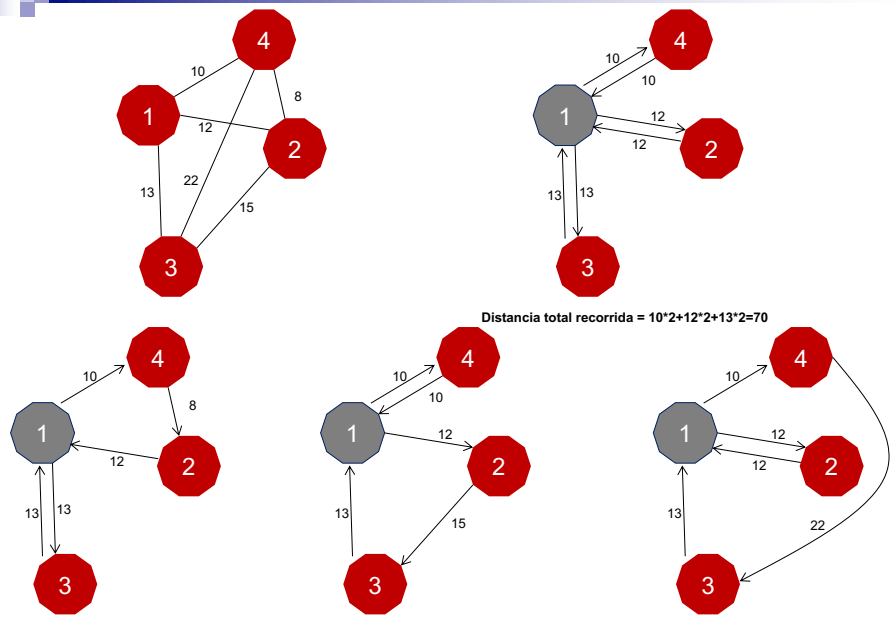
■47

Algoritmo de Clarke y Wright para el TSP

- Se trata de una de las técnicas más conocidas y más usadas para resolver el problema del TSP (y VRP)
- Parte de la idea de que, para visitar todas las ciudades, el viajante parte de una ciudad cualquiera que califica como "origen" y hace tantos viajes de ida y vuelta como ciudades ha de visitar. Evidentemente, esta solución no representa un ciclo y no sirve como solución al problema
- La clave del algoritmo consiste en calcular el ahorro que supondría, frente a la solución actual, unir dos de las ciudades a visitar. Es decir, visitar una tras la otra sin necesidad de volver al origen
- La complejidad de este algoritmo es de $O(n^3)$

48

■48



49

■49

4. Técnicas heurísticas de mejora

- Parten de una solución factible al problema
- En algunos problemas, obtener una solución factible inicial ya puede requerir mucho esfuerzo computacional
- Se trata de un proceso iterativo que trata de mejorar la solución actual mediante algún tipo de cambio en la misma, deteniendo el proceso cuando dicha solución no puede mejorarse mediante este tipo de cambios
- La decisión tomada en cada iteración sobre cómo mejorar la solución, puede ser determinante en el resultado final
- El resultado final también puede depender de la solución factible inicial
- Suelen ser rápidos, por ello en la práctica es habitual aplicarlos distintas veces a distintas soluciones de partida, para finalmente, quedarnos con la mejor.

50

■50

- Ejemplos
  - N-Opt - TSP, VRP
  - Palmer - FLOWSHOP

51

■51

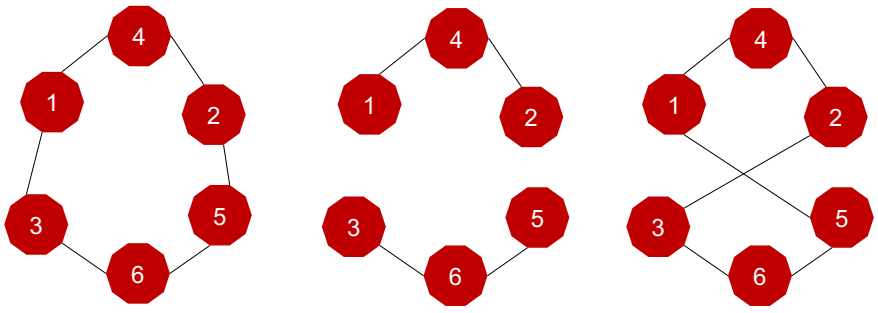
Algoritmo 2-opt para el TSP

- Parte de una solución factible, como puede ser un ciclo obtenido con uno de los algoritmos constructivos estudiados en los apartados anteriores
- En cada iteración se trata de mejorar la solución actual mediante algún tipo de cambio en la misma,
- En este caso se prueba el cambio de dos arcos del ciclo que tenemos por dos arcos diferentes, comprobando si esto supone una mejora, es decir, una reducción de la distancia total recorrida
- Si dicho cambio es beneficioso, se lleva a cabo y se prueba otro. Si no lo es, se prueba otro y el proceso se repite hasta que ninguno de estos cambios consiga mejorar la solución
- La complejidad de este algoritmo es de $O(n^2)$

52

■52

- Si se eliminan dos enlaces de un ciclo, existe una única forma de añadir dos enlaces nuevos para “arreglar” el ciclo



53

■53

- En un ciclo de n ciudades, existen $\binom{n}{2} - n$ formas diferentes de elegir los dos enlaces a eliminar
- Cuando ya no es posible mejorar la solución cambiando dos enlaces a la vez, se dice que la solución es 2-óptima (lo que no significa que sea la solución óptima al problema)
- El algoritmo 2-opt se puede generalizar al caso 3-opt, 4-opt, …, k-opt. Una solución k-óptima es también (k-1)-óptima.
- Una opción para mejorar soluciones podría consistir en aplicar el algoritmo 2-opt y a la solución obtenida el 3-opt.
- A continuación se presentan los pasos del algoritmo

54

■54

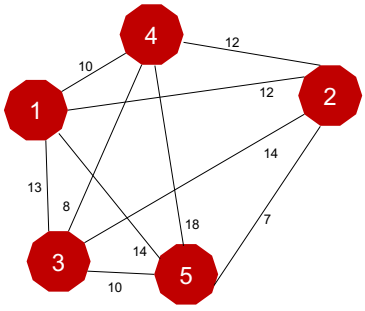
- Pasos
  1. Partir de un ciclo que represente una solución al problema
  2. Elegir 2 enlaces no incidentes sobre el mismo nodo
  3. Comprobar si el ciclo que obtendríamos al sustituir dichos enlaces por otros dos es mejor que el que tenemos
  4. Si es así, realizar la sustitución de los enlaces. En caso contrario, dejar el ciclo sin alterar
  5. Si quedan cambios por probar, ir al paso 2

55

■55

- Ejemplo:

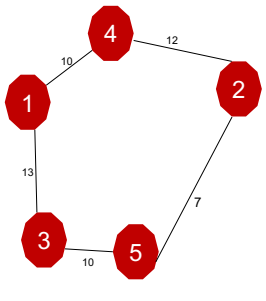
Dado el siguiente grafo



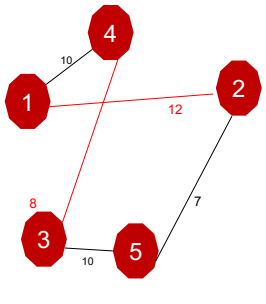
56

■56

Aplicar el algoritmo de mejora 2-opt. a la siguiente solución de partida:



TOTAL: 52

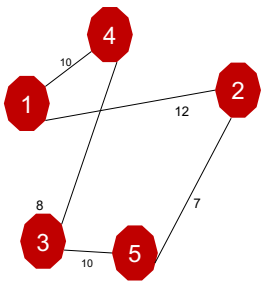


TOTAL: 47

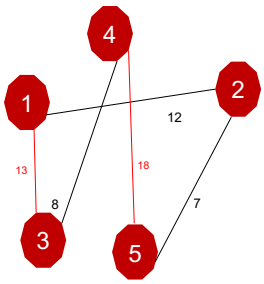
57

■57

Aplicar el algoritmo de mejora 2-opt. a la siguiente solución de partida:



TOTAL: 47



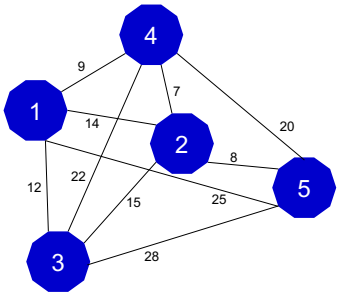
TOTAL: 58

¿Quedan cambios por probar?…

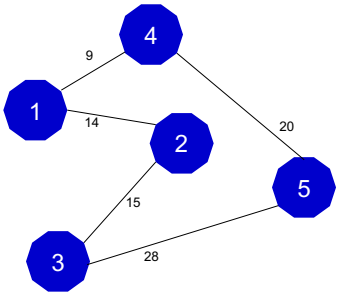
58

■58

- Ejercicio



Dado el grafo anterior, donde los valores representan el coste de transporte entre ambos nodos, aplicar el algoritmo de mejora 2-opt a la solución de partida:



59

■59

Resumen

- Los problemas de optimización combinatoria son difíciles por el elevado esfuerzo computacional que requieren
- El número de posibles soluciones es finito, pero tan elevado que enumerarlas se hace inabordable
- El número de soluciones crece de forma exponencial
- Las técnicas exactas no son aplicables en la práctica
- Las técnicas heurísticas se convierten en una (la única) alternativa, obteniendo una solución aceptable con un relativo esfuerzo computacional
- Un nuevo tipo de técnicas, las metaheurísticas, son las que obtienen actualmente las mejores soluciones
- Algunas técnicas heurísticas sencillas se utilizan como parte del proceso en algunas metaheurísticas

60

■60

Bibliografía

- Meta-heuristics and Artificial Intelligence (2020), J.K. Hao and C. Solnon, in: A guided tour for Artificial Intelligence research 2: AI algorithms, P. Marquis, O. Papini, H. Prade (Eds), pp. 27-52, Springer.
- Handbook of heuristics (2020), R. Martí, P. Panos, M.G.C. Resende. Springer.
- Combinatorial optimization: Theory and algorithms (2012), B. Korte, J. Vygen. Springer.
- Handbook of combinatorial optimization (1998), D. Dhu, P.M. Pardalos. Springer Science and Business Media.
- Modelling techniques and heuristics for combinatorial problems (1975), H. Müller-Merbach, in: Roy, B. (eds) Combinatorial Programming: Methods and Applications. NATO Advanced Study Institutes Series, vol 19. Springer.

61

■61