Meta Heurísticas

P1 - Búsqueda por trayectorias para el problema de la asignación cuadrática

Tercero Grado Ing. Informática

Alejandro Alcalde.
Grupo 2 miércoles a las 17.30

Contents

1 Abstract

El problema de la asignación cuadrática (QAP) es un problema estándar en teoría de localización. En éste se trata de asignar N unidades a una cantidad N de sitios o localizaciones en donde se considera un costo asociado a cada una de las asignaciones. Este costo dependerá de las distancias y flujo entre las unidades, además de un costo adicional por asignar cierta unidad a cierta localización específica. De este modo se buscará que este costo, en función de la distancia y flujo, sea mínimo.

2 Definición matemática

$$\underset{S\in\prod_N}{min}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\cdot d_{S(i)S(j)}\right)$$

Donde $\prod_N$ es el conjunto de todas las permutaciones posibles de $N = 1, 2, \ldots, n$

3 Representación del problema

En éste problema las soluciones se pueden representar como permutaciones de un conjunto. Si el problema es de tamaño cuatro, por ejemplo, una solución vendría dada por la permutación $N = \{3, 2, 1, 4\}$. Si tomamos los índices de éste conjunto como las unidades, y el valor en dicho índice como localizaciones, la localización 3 estaría asignada a la unidad 1, la localización 2 a la unidad 2 etc.

El objetivo del problema es **minimizar** la expresión mostrada anteriormente. Ésta será la función objetivo:

$$\underset{S\in\prod_N}{min}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\cdot d_{S(i)S(j)}\right)$$

Hará falta un mecanismo para **generar la solución inicial**, en nuestro caso, la permutación N inicial sobre la que lanzar los distintos algoritmos será aleatoria.

De igual modo será necesario un esquema de **generación de soluciones vecinas**. Para ello, se realizará un intercambio entre dos localizaciones, cambiando sus respectivas unidades por la otra. Para calcular el coste de ésta nueva permutación no será necesario evaluar de nuevo la función objetivo, se puede emplear la siguiente factorización:

$$\sum_{k\neq r,s}\begin{bmatrix} f_{rk}\cdot(d_{\pi(s)\pi(k)}-d_{\pi(r)\pi(k)})+f_{sk}\cdot(d_{\pi(r)\pi(k)}-d_{\pi(s)\pi(k)})+ \\ f_{kr}\cdot(d_{\pi(k)\pi(s)}-d_{\pi(k)\pi(r)})+f_{ks}\cdot(d_{\pi(k)\pi(r)}-d_{\pi(k)\pi(s)}) \end{bmatrix}$$

El **criterio de parada** para todos los algoritmos salvo para la Búqueda local será 10000 evaluaciones. La búsqueda local se detendrá cuando no encuentre una solución mejor en todo el entorno.

3.1 Algoritmo Greedy

El primer algoritmo será el más sencillo y más ineficiente a la hora de minimizar la función de coste. Consite simplemente en asociar unidades de gran flujo con localizaciones céntricas en la red y viceversa. Para ello se calcula el potencial de flujo y de distancia. A mayor potencial de flujo, más peso tendrá dicha unidad en el intercambio de flujos y, a menor flujo de distancia, más céntrica será la localización. Por tanto el algoritmo selecciona la unidad disponible con mayor potencial de flujo y le asignará la localización de menor potencial de distancia.

```
Calculo potenciales.
Inicialzar N a 0.
para i = 1 hasta el tamaño del problema
    Seleccionar unidad de mayor potencial de flujo
    Seleccionar localización de menor potencial de distancia
    Añadir la localización seleccionada al índice correspondiente a la unidad en N
```

3.2 Algoritmo Búsqueda local

En éste algoritmo se empleará una técnica llamad *Don't Look Bits* que permite focalizar la Búsqueda en una zona del espacio en la que se puede encontrar una solución mejor que la actual.

Explicación de la DLB. Con ésta máscara marcamos con 0 todas la unidades inicialmente, indicando que se exploran todos en orden secuencial. El 1 indica que ese vecino no interesa ser explorado. Por ejemplo, si en 1 hay un 1, no lo cogemos para intercambiarlo. El 1 se establece cuando se ha terminado una iteración del bucle interno sin escoger una solución. Es decir, se ha probado a intercambiar todos.

```
        DLB = 1..N inicializado a 0
        for i = 1 .. n
            si DLB_i == 0 // lo puedo mirar
                for j = 1 .. n
                    vecino = intercambio(i,j)
                    comprar(vecino, solucion actual)
                    Si mejora solucion actual = vecino
                    DLB_i = DLB_j = 0
```

Pseudo Código de la BL.

```
    Iterar sobre el tamaño del problema
        Si en la DLB hay un 0 para el elemeno actual // Se considera mirar ese movimiento
            Se vuelve a iterar sobre el tamaño del problema
                Se realiza el intercambio
                Si hubo mejora
                    Actualizar el coste actual
                    Actualizar los correspondientes elementos de la DLB a 0
            Si tras analizar todo el espacio posible no se mejoró
                Actualizar DLB con 1 en el elemento correspondiente
```

3.3 Algoritmo Enfriamiento Simulado

Éste algoritmo pretende simular el cómo actúan las partículas cuando se encuentran bajo una temperatura alta, y cómo van reduciendo su movimiento a medida que la temperatura decrementa. De ésta forma, al inicio del algoritmo, con una temperatura alta, podremos aceptar soluciones que parezcan malas, pero nos puedan conducir a alguna mejor.

```
    mientras el número de evaluaciones sea menor a una constante
        generar vecnios mientras se pueda seguir explorando el entorno
            Si el vecino mejora la solución actual o lo aceptamos con la probabilidad indicada
                actualizar solucion actual
                Actualizar mejor solucion encontrada si la actual es mejor
            Enfriar la temperatura
    devolver la mejor solucion encontrada
```

4 Procedimiento considerado

Para realizar la práctica he comenzado completamente desde cero escribiendo la práctica en python. La razón inicial era la simplicidad de éste lenguaje. Como contramedida he sacrificado algo de eficiencia, y éste es el motivo de que no haya entregado el algoritmo búsqueda tabú, ya que es muy ineficiente[**q1**]. El único requisito para ejecutar el programa es instalar el módulo numpy.

5 Análisis de resultados

La semilla usada ha sido 3264321546.

```
Media Desv: 13.77
Media Tiempo: 0.09

Algoritmo Local Search
Caso    Coste obtenido  Desv    Tiempo
Els19   23039740    33.85   0.0052919388
Chr20a  3664    67.15   0.005936861
Chr25a  5440    43.31   0.0172650814
Nug25   3862    3.15    0.0176279545
Bur26a  5423456 -0.06   0.0201458931
Bur26b  3820940 0.08    0.0200889111
```

```
Tai30a  1932076 6.27    0.0206849575
Tai30b  769806538   20.83   0.0317540169
Esc32a  166 27.69   0.0370919704
Kra32   99300   11.95   0.036011219
Tai35a  2585344 6.74    0.0347859859
Tai35b  329733820   16.38   0.0333559513
Tho40   254802  5.94    0.0555479527
Tai40a  3317708 5.68    0.0723600388
Sko42   16576   4.83    0.0555119514
Sko49   24742   5.80    0.1251809597
Tai50a  5227752 5.85    0.0928778648
Tai50b  472624311   3.01    0.123623848
Tai60a  7643446 6.07    0.2409579754
Lipa90a 363756  0.87    0.8072071075
```

Como se puede observar, la desviación con respecto a las mejores soluciones encontradas hasta el momento es considerable. Se puede intuir por tanto, que éste algoritmo se ha quedado en un mínimo local.

```
Media Desv: 8.45
Media Tiempo: 0.44

Algoritmo Simulated Annealing
Caso    Coste obtenido  Desv    Tiempo
Els19   17937024    4.21    0.3101348877
Chr20a  2960    35.04   0.2148449421
Chr25a  5704    50.26   0.2483611107
Nug25   3880    3.63    0.3764858246
Bur26a  5360431 -1.22   0.3954520226
Bur26b  3865077 1.24    0.3683228493
Tai30a  1885504 3.70    0.2872378826
Tai30b  734668508   15.31   0.4554789066
Esc32a  152 16.92   0.3011479378
Kra32   93590   5.51    0.4045190811
Tai35a  2539994 4.87    0.4867529869
Tai35b  290761479   2.63    0.3325610161
Tho40   247458  2.89    0.5354738235
Tai40a  3251852 3.58    0.3710420132
Sko42   16354   3.43    0.5607919693
Sko49   24110   3.10    0.4178740978
Tai50a  5164202 4.56    0.4377248287
Tai50b  477784303   4.13    0.6221811771
Tai60a  7521828 4.38    0.7958261967
Lipa90a 363448  0.78    0.7854909897
```

```
Algoritmo   Desv    Tiempo
Greedy  74.4658360474   0.001180172
BL  13.7700430291   0.092665422
ES  8.4482194635    0.4353852272
```

Bibliografía