Estudio comparativo entre metaheurísticas de trayectoria y metaheurísticas poblacionales aplicadas al problema de asignación cuadratico QAP

Jose Cipagauta Antonio Alvarez

Abstract

Una descripción breve del paper.

1 Introducción

El problema de asignación cuadrática es un problema combinatorio en el cual hay "n" localidades con distancias definidas entre ellas y almacenadas en una matriz D y "n" sucursales con flujos determinados en una matriz F. La idea es asignar cada sucursal a una localidad distinta tal que se minimice la siguiente ecuación:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{p(i)p(j)}$$

Para esto hay que hallar una permutación "p" que lo haga posible [1]. Dicha permutación expresa el orden de asignación de una localidad con respecto a las sucursales o viceversa.

Este problema fue demostrado NP-completo por Sahni, S. y T. Gonzalez [2] lo cual hace que sea infactible una búsqeda exhaustiva en el espacio de soluciones para n de tamaño moderado y grande por ser n! combinaciones posibles. En su lugar se usan algoritmos de aproximación que consigan una respuesta dentro de un margen de error dado y en un tiempo de cómputo razonable.

Estos algoritmos de aproximación empiezan con una solución inicial y dependiendo de los criterios de selección presentes, el algoritmo se va moviendo por el espacio de soluciones hasta que se cumple el criterio de parada y devuelve la mejor solución encontrada [5].

Para este trabajo se utiliza la librería QAPLIB [4] y en lo referente al resto del informe se tiene que en la sección 2 se exponen los detalles de la representación del problema y los detalles de la implementación de los algoritmos utilizados para la resolución de QAP, en la sección 4 se muestran los resultados obtenidos y en la sección 5 se hacen las conclusiones de este trabajo.

2 Detalles de implementación

Las soluciones factibles se modelan como un arreglo de enteros de tamaño n con una permutación de los número del 1 al n (incluidos) donde n representa el número de localidades y sucursales. La operación de movimiento o vecindad se define como el intercambio de exactamente 2 elementos dentro del arreglo, por lo tanto la vecindad está formada por $n*(n+1)/2$ elementos distintos. La función objetivo es: $\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{p(i)p(j)}$ y una solución es considerada mejor que otra si al evaluarla con la función objetivo se obtiene un resultado menor.

Ahora pasamos a la implementación de los distintos algoritmos utilizados:

2.1 Búsqueda Local

Búsqueda local se usa como la heurística base con la cual se comparan las otras metaheurísticas. La idea de búsqueda local es ir eligiendo soluciones cada vez mejores en cada iteración hasta que no se consiga ninguna mejoría en la vecindad de la solución actual, situación que da por culminada la búsqueda y da como resultado un óptimo local dentro del espacio de soluciones.

Para la solución inicial se optó por una solución aleatoria porque como explica Talbi, E. G. [6] Con una vecindad de grandes dimensiones se mitiga el impacto que genera la elección de la solución inicial como es el caso de las instancias con más de 100 localidades y para las instancias pequeñas (menos de 30 localidades) el aumento en iteraciones del movimiento por el espacio de soluciones no muestra un impacto importante en el tiempo de cómputo cuando en promedio la evaluación de una instancia de 30 toma 0.12 segundos.

Luego se decidió parametrizar búsqueda local de tal manera que pudiera operar bajo distintos criterios de selección de vecinos en cada iteración; estos criterios son: elegir el primer vecino encontrado que mejore la solución actual, elegir la mejor solución de la vecindad, elegir la mejor solución dentro de un porcentaje dado de la vecindad y elegir un vecino aleatorio.

En los procesos de búsqueda en la vecindad se empieza la permutación ordenada de cada par de elementos dentro del arreglo de la solución de tal manera que se verifiquen todas las combinaciones posibles (excepto para la búsqueda aleatoria).

En la figura 1 se presenta el pseudo-código para el local search propuesto. Nótese que f es la función objetivo

Algorithm 1: Busqueda Local

Input: matrices con las distancias y flujos y tipo de selección dentro de la búsqueda
Output: Arreglo con una permutación de las asignaciones de las localidades a las sucursales

```
S = S0    /* escogiendo S0 aleatoriamente */
repeat
    Generar vecino S' según el tipo de búsqueda
    if f(S') ≤ f(S) then
        S = S'
until f(S) ≠ f(S')
return S
```

Table 1: Resultados de la evaluación de la función objetivo de las distintas búsquedas locales

Instancia	Mejor posible	Primer mejor	Aleatorio	Optimo
chr12a	12833.2	13130.2	34148.8	9552
chr15a	15936.4	15004.5	43616.4	9896
chr20a	3398.8	3328.3	9086.6	2298
nug21	2579.6	2502	3454.2	2438
chr22a	6945.6	7294.1	11814	6156
nug24	3685.6	3624.8	4586.9	3488
chr25a	5768.4	5740.1	18706.7	3796
nug27	5392.4	5543.3	6642.6	5234
nug28	5386.5	5325	6434	5166
lipa30a	13273.5	13272.4	13880	13178
ste36a	10798	10710	21406.3	9526
lipa40a	31650	31770.2	32520.1	31538
sko42	16445.5	16066.7	19812.1	15812
wil50	49338	49544	55894	48816
lipa50a	62326	62320.4	63844.2	62093
sko56	35539.3	35108.4	41446.2	34458
lipa60a	108402.6	108362	110316.8	107218
lipa70a	171228.6	171218	174510.1	169755
sko72	68120.6	68072.9	78536.5	66256
lipa80a	255178	255242	259048	253195
sko81	93160.7	92474.8	108354.9	90998
lipa90a	361080	361570.5	367048	360630
sko100a	155578.7	154780.3	178636.2	152002
esc128	66	76	326	64
tai256c	44975794	44950742.2	53359164.7	44759294

2.2 Recocido simulado

Se completa cuando se tengan bien ajustados los parámetros

3 Resultados

Las tablas 1 y 2 muestran los resultados de la evaluación de la función objetivo y tiempo de cómputo para las distintas variantes de búsqueda local (elección de la mejor solución dentro de la vecindad, primera solución que mejore la evaluación de la función de costo actual o elección aleatoria de un vecino).

Table 2: Resultados del tiempo de cómputo de las distintas búsquedas locales

Instancia	Mejor posible	Primer mejor	Aleatorio
chr12a	0.003433	0.003792	0.000421
chr15a	0.00635	0.003551	0.000475
chr20a	0.026156	0.019761	0.000595
nug21	0.027964	0.041645	0.000536
chr22a	0.05023	0.034667	0.000661
nug24	0.057157	0.058387	0.000626
chr25a	0.081039	0.046922	0.000807
nug27	0.12959	0.091156	0.000867
nug28	0.129343	0.079674	0.000954
lipa30a	0.139231	0.116615	0.000999
ste36a	0.523082	0.487484	0.001381
lipa40a	0.571939	0.309533	0.001646
sko42	1.11221	1.39335	0.002022
wil50	3.26927	3.58704	0.002199
lipa50a	2.73087	0.89528	0.002513
sko56	4.67231	6.42853	0.002522
lipa60a	4.14559	2.20789	0.003031
lipa70a	11.1112	3.74318	0.003342
sko72	28.7295	25.636	0.004021
lipa80a	21.0467	10.7404	0.004055
sko81	37.1344	60.6198	0.004808
lipa90a	39.7775	18.5285	0.005018
sko100a	112.649	123.815	0.074144
esc128	65.2117	20.6649	0.011112
tai256c	2079.78	1695.5579	0.638859

Conclusiones

Aquí concluyen.

References

[1] Burkard, R. et al. QAPLIB-A Quadratic Assignment Problem Library. 2002

[2] Sahni, S., and Gonzalez, T. P-complete approximation problems. *J. of ACM*, 23, 555–565. 1976

[3] Misevičius, A. A modified simulated annealing algorithm for the quadratic assignment problem *Informatica*, 14(4), 497-514. 2003

[4] Burkard, R. et al. Quadratic Assignment Problem Library. http://anjos.mgi.polymtl.ca/qaplib/ . 2002

[5] Talbi, E. G. Metaheuristics: from design to implementation *John Wiley* 2009

[6] Ibid. 101

Apéndice

Bla.