Inteligencia Artificial

Informe final: Quadratic Assignment Problem (QAP)

Fabian Lorca Trujillo

June 23, 2023

Evaluación

Resumen (5%):	_____
Introducción (5%):	_____
Definición del Problema (10%):	_____
Estado del Arte (35%):	_____
Modelo Matemático (20%):	_____
Conclusiones (20%):	_____
Bibliografía (5%):	_____
Nota Final (100%):	_____

Abstract

El Problema de Asignación Cuadrática o QAP (Quadratic Assignment Problem) por sus siglas en inglés, es un modelo de optimización combinatorial que consiste en minimizar los costos de instalación de n plantas en n lugares considerando los flujos y distancias entre estos. Este problema no se puede resolver en tiempo polinomial, lo que se traduce en tiempos de ejecución muy grandes. En este informe se explicará en que consiste el modelo. También se discutirán algunos de los algoritmos que se han utilizado para resolver el problema, explicando brevemente en qué consisten y cuál de ellos ha entregado mejores resultados, y se propondrá un algoritmo que utiliza GBJ (Graph-based BackJumping) como técnica de resolución. Tambien se llevara a cabo un experimento para evaluar el rendimiento del algoritmo y se analizaran los resultados obtenidos.

1 Introducción

El propósito de este informe es exponer el Estado del Arte del Problema de Asignación Cuadrática, más conocido como QAP (Quadratic Assignment Problem) en Inglés. Para abarcar mejor los distintos aspectos del problema, el contenido se dividirá en 9 secciones principales. Esta sección corresponde a la introducción al problema del QAP. En las secciones 2, 3 y 4 se encuentran la definición del problema, el estado del arte y el modelo matemático del problema. En estas secciones se explicara mas a fondo en que consiste el problema, como se origina, que métodos se han utilizando para resolverlo y como se expresara matemáticamente. En la sección 5 se explicará la representación a utilizar para la implementación del algoritmo propuesto, en la sección 6 se explicara mas a detalle su funcionamiento. En las secciones 7 y 8 se explicara como

se realizaran los experimentos para evaluar el funcionamiento del algoritmo y luego se mostraran los resultados obtenidos. Finalmente, en la sección 9 se encontraran las conclusiones obtenidas sobre el problema y el algoritmo propuesto.

El problema de Asignación cuadrática, a grandes rasgos, consiste en asignar N unidades a N diferentes lugares minimizando los costos que implica. Mas adelante se verá que no es un problema trivial. La motivacion a la hora de realizar este informe es mostrar al lector los avances realizados en el estudio de este modelo comenzando por la base. También, se propondrá un algoritmo para la resolución del problema. Este algoritmo utilizara GBJ (Graph-based BackJumping), el cual es una técnica completa de resolución.

2 Definición del Problema

El problema de asignación cuadrática (QAP) es un problema de optimización combinatorial que surge del dilema de asignar un conjunto de instalaciones en un conjunto de lugares, conociendo el flujo entre instalaciones y distancia entre lugares. El producto entre estas constantes se conocerá como el costo de asignar una instalación a un lugar determinado. El objetivo es distribuir las instalaciones, de manera que, dicho costo sea mínimo. Hay que considerar ciertas restricciones, como que cada instalación debe estar asignada a exactamente un solo lugar y cada lugar debe estar asociado a exactamente una sola instalación, y que la variable es discreta, es decir, solo puede ser 0 o 1.

Existe una gran variedad de problemas que se pueden resolver con este modelo, aparte de los problemas relacionados con distribuir instalaciones. Uno de ellos es el diseño de circuitos VLSI [2], donde se utiliza la variante BiQAP para colocar de manera eficiente los componentes electrónicos en un circuito impreso. Otro de ellos es la planificación de líneas de producción en paralelo [6], donde se utiliza QAP para reducir costos al distribuir de mejor forma instalaciones de procesamiento tomando en cuenta elementos como el costo de producción, tasa de producción, etc.

También existen variantes del QAP, que se obtienen al realizar cambios en la función objetivo. En el trabajo de Burkard et al.[3] se mencionan estos problemas. Una de estas variantes es Bottleneck QAP (BQAP), cuya diferencia es que se reemplaza la sumatoria por una función max. Otra variante es BiQuadratic Assignment Problem (BiQAP), el cual es, un problema de asignación de cuarto grado. Tambien existe la variante Quadratic Semi-Assignment Problem (QSAP), que es muy similar a QAP, con la diferencia de que, ahora, se tienen n instalaciones y m lugares, con $n > m$. También esta el Traveling Salesman Problem (TSP), que puede ser formulado como un QAP si se considera que los flujos conectan todas las instalaciones formando un solo anillo o ciclo.

3 Estado del Arte

El problema de asignación cuadrática (QAP) fue propuesto por Koopmans y Beckmann [7] en 1957 como un modelo para resolver o analizar el problema de asignar recursos indivisibles. Según el autor, estas indivisibilidades en la mayoría de los casos eran la raíz del aumento de los rendimientos de escala. Si dicho aumento persiste a un nivel comparable al de la demanda total del mercado de interés, se imposibilita la competencia perfecta y reduce la eficacia del sistema de precios en la asignación eficiente de recursos. De ahí la importancia de asignar eficientemente los recursos indivisibles.

A continuación se presentan alguno de los algoritmos que han obtenido buenos resultados a la hora de resolver este problema.

Simulated Annealing (SA): Wilheim et al. [13] ha utilizado un algoritmo de búsqueda meta-heurística llamado Simulated Annealing, el cual mientras itera en busca de una posible solución, utiliza el método de Monte Carlo (Aleatoriedad) para aceptar o rechazar las soluciones. El autor

afirma que este algoritmo puede generar soluciones de alta calidad y un tiempo de ejecución aceptable para valores de $n = 50$ y $n = 100$. Pero, el algoritmo es muy sensible al número de parámetros.

Robust Taboo Search (Ro-TS): Taillard [11] ha utilizado el algoritmo de Taboo Search, el cual consiste en una búsqueda local de soluciones utilizando una lista tabú que registra las soluciones ya visitadas. Esta lista es una memoria de corto plazo que almacena las soluciones visitadas hace menos de n iteraciones atrás.
Para resolver de manera eficiente el QAP, ha modificado este algoritmo, de forma que, requiera menos complejidad y pocos parámetros en comparación con adaptaciones anteriores, generando buenos resultados para valores de n entre 5 y 100.

GRASP with Path-Relinking (GRASP PR): Oliveira et al. [9] ha utilizado GRASP (Greedy randomized adaptive search procedure) añadiendole Path-Relinking, el cual es, una estrategia de intensificacion de busqueda utilizada para mejorar la calidad de las soluciones encontradas.GRASP hace uso de un algoritmo greedy para generar soluciones de forma aleatoria, alguna de estas son almacenadas en una lista de candidatos (Restricted Candidate List) según ciertos criterios. Finalmente, se aplica búsqueda local en las soluciones candidatas.
Los resultados del experimento muestran que GRASP PR genera soluciones de mejor calidad que GRASP.

Greedy Genetic Algorithm (GGA): Ahuja et al. [1] ha desarrollado un algoritmo genético con principios greedy. Estos principios han mejorado el rendimiento general del algoritmo siempre y cuando se evite su uso excesivo, ya que, si bien, un mayor uso de principios greedy mejora la precisión de los resultados, afecta enormemente el rendimiento. Se han realizado pruebas donde se encontró que obtiene mejores resultados que el algoritmo GRASP, pero el autor dice que estos resultados solo son sugestivos y de ninguna manera conclusivos, debido a que hay ciertos parámetros que no se han considerado.

Iterated local search (ILS): Stützle [10] ha utilizado ILS, el cual consiste en una modificación del clásico algoritmo de búsqueda local, donde, se utiliza un algoritmo que perturba el mínimo local encontrado , para luego realizar otra búsqueda local.
El autor ha demostrado que el algoritmo de ILS básico sufre de estancamiento, comprometiendo su rendimiento, para solucionar esto, el autor le ha implementado criterios de aceptación para mejorar el rendimiento general, obteniendo resultados satisfactorios y mejores que otros algoritmos.

De todos los algoritmos presentados, el algoritmo Ro-TS ha sido el que ha tenido resultados mas consistentes en general, superando en velocidad de respuesta a los algoritmos ILS, SA, GGA Y GRASP PR, sobre todo en instancias de tipo i(Problemas aleatorios), mientras que el algoritmo ILS con extensiones (ES) propuesto por Stützle ha tenido un rendimiento ligeramente mejor que Ro-TS en instancias de tipo ii(Flujos aleatorios), iii(Instancias reales) y iv(Instancias generadas aleatoriamente semejantes a instancias reales). Por lo que, de todos los algoritmos mencionados, ILS(ES) y Ro-TS son los mejores.

4 Modelo Matemático

Como ya se ha mencionado anteriormente, el problema consiste en asignar un conjunto de instalaciones a un conjunto de lugares, minimizando los costos. En el trabajo de Koopmans-Beckmann [7] se define el problema de la siguiente forma: Se tiene n plantas y n lugares y se definen las matrices b_{kl}, c_{ij} y p_{ki} de tamaño $n \times n$, donde, $b_{kl}, k \neq l, \quad k, l = 1, ..., n$ es el flujo desde la planta k a la planta l, $c_{ij}, i \neq j, \quad i, j = 1, ..., n$ es el costo de transporte de una unidad, o distancia entre el lugar i y el lugar j y p_{ki} es la matriz de permutación que representa si se instala o no la planta k en el lugar i. También se define la matriz a_{ki} que representa los ingresos semi-netos al colocar la planta k en el lugar i, pero en la mayoría de los textos, la matriz representa el costo de instalación de la planta. Esta matriz no será utilizada en el algoritmo a

implementar para la resolución del problema. Entonces, la función objetivo queda como:

$$\min \sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{kl}p_{ki}c_{ij}p_{lj} + \sum_{k=1}^{n}\sum_{i=1}^{n} a_{ki}p_{ki} \tag{1}$$

El objetivo es encontrar una matriz de permutación p_{ij}, tal que, el costo sea mínimo. Esta ecuación también se puede escribir como:

$$\min_{\phi \in S_n} \sum_{k=1}^{n}\sum_{l=1}^{n} b_{kl}c_{\phi(k)\phi(l)} \tag{2}$$

Donde S_n es el conjunto de todas las permutaciones $\phi : N \to N$, donde $N = \{1, ..., n\}$ y $\phi(k)$ representa el lugar asignado a la planta k en la solución $\phi \in S_n$. ϕ también puede expresarse como una matriz de permutación p_{ij} de tamaño $n \times n$, tal que:

$$p_{ij} = \begin{cases} 1 \text{ si } \phi(i) = j \\ 0 \text{ en otro caso} \end{cases} \tag{3}$$

Las restricciones del modelo son:

$$\sum_{i=1}^{n} p_{ij} = 1, \qquad j = 1, 2, ..., n,$$

$$\sum_{j=1}^{n} p_{ij} = 1, \qquad i = 1, 2, ..., n,$$

$$p_{ij} \in \{0, 1\}, \qquad i, j = 1, 2, ..., n.$$

Las primeras dos restricciones son para asegurarse de que cada planta debe tener asignada solo 1 lugar, la ultima restricción define que p_{ij} es una variable binaria. Estas restricciones se aplican a ambas versiones del modelo (ecuaciones (1) y (2)).

La ecuación (2) sera la función objetivo a utilizar para la propuesta de solución del problema.

Como se ha mencionado antes, en el trabajo de Koopmans-Beckmann se define una matriz a_{ki} que representa los ingresos semi-netos al instalar la planta k en el lugar i. En este caso, la función objetivo queda como:

$$\max \sum_{k}^{n}\sum_{i}^{n} a_{ki}p_{ki} - \sum_{k}^{n}\sum_{l}^{n}\sum_{i}^{n}\sum_{j}^{n} b_{kl}p_{ki}c_{ij}p_{lj} \tag{4}$$

Las restricciones de esta versión son las mismas que las de las ecuaciones (1) y (2). Entonces, las ecuaciones (1) o (2) se utilizan para minimizar costos y (4) se utiliza para maximizar ganancias.

5 Representación

Debido a que GBJ es una técnica completa, se debe adaptar el modelo, de forma que se utilice la menor cantidad de variables para reducir el numero de iteraciones. Esto se puede lograr si se considera una matriz de permutación unidimensional de tamaño n, tal que, el valor x que se encuentre en cierta posición i de dicha matriz, representara la instalación de la fabrica x en el lugar i. Como se puede ver, solo se tendrían n variables, en cambio, si se utilizara la matriz de permutación p_{ij} como el conjunto de variables, se tendrían n^2 variables, lo cual no es bueno considerando la técnica a utilizar. Ahora, como x representa a una fabrica, su dominio serán los n posibles lugares donde puede ser instalado. Luego, considerando las restricciones del

problema, el grafo de restricciones a utilizar sera un grafo completo donde para cada arista se tendrá la restricción binaria de desigualdad, es decir, por cada par de variables conectadas en el grafo, se debe cumplir que son distintos.

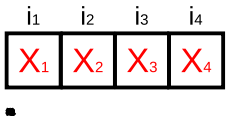

Figure 1: Ejemplo de una matriz de permutación para una instancia de tamaño 4, donde i representa las posiciones (lugares) y x representa las fabricas

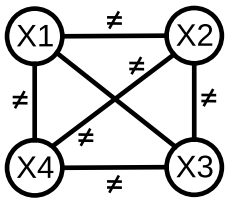

Figure 2: Ejemplo del grafo de restricciones de la instancia

6 Descripción del algoritmo

Para la la resolución del problema, se propone un algoritmo que utiliza GBJ(Graph-based Back-Jumping) para buscar soluciones candidatas a óptimo global. Se ha decidido utilizar un orden de instanciación secuencial $(x_0, x_1, x_2, ...)$, con el objetivo de facilitar todas las operaciones de instanciación de variables, obtención de la variable mas recientemente instanciada para realizar backJump, comprobación de fallos y otras consideraciones menores.

El algoritmo primero define el lugar donde se guardaran las variables y dominios, luego, calcula todas las variables conectadas a cada variable x_i según el grafo de restricciones mencionado en la sección anterior, y los almacena como una serie de conjuntos ordenados que contendrán los vecinos de cada variable. Estos conjuntos serán utilizados para conocer la variable mas recientemente instanciada y realizar backJump. Luego, comienza la parte mas importante del algoritmo que es el bucle. En esta parte del algoritmo se va asignando valores a las variables, y cuando todas las variables tienen un valor asignado (perteneciente su dominio), se evalúan en la función objetivo donde se obtiene una solución candidata que es guardada hasta encontrar una mejor. Luego de encontrar una solución, aplicara GBJ para continuar buscando soluciones. Cuando ya no se puedan realizar backJumps, el algoritmo sale del bucle. Las restricciones son revisadas cuando se asignan valores a las variables. Lo que se hace es asignar a la variable un valor dentro del dominio, luego, se verifica que el valor asignado cumpla con las restricciones. De no ser así, se asigna otro valor del dominio y se repite el proceso hasta encontrar un valor que cumpla con las restricciones. En el caso de que esto no se cumpla, se aplica GBJ. Esto se hace así, ya que, de esta forma se puede asegurar que el valor asignado no genera conflictos.

7 Experimentos

Con el objetivo de evaluar el rendimiento del algoritmo, se realizaran distintas pruebas de instancias conocidas de los autores N. Christofides y E. Benavent [4], C.E. Nugent et al [8] y

E.D. Taillard [11] [12]. Estas instancias han sido generadas de distintas formas, las instancias de Christofides consisten en una matriz de adyacencia que forma un arbol balanceado y otra matriz que forma un grafo completo, es decir, todas las fabricas están conectadas entre si, las instancias de Nugent están basadas en casos reales y las instancias de Taillard fueron generadas uniformemente (Versión a) y aleatoriamente (Versión b). Las instancias de Christofides escogidas son: chr12a.dat, chr12b.dat, chr12c.dat, chr15a.dat, chr15b.dat y chr15c.dat. Las instancias de Nugent escogidas son: nug12.dat, nug14.dat y nug15.dat. Finalmente, las instancias de Taillard escogidas son: tai10a.dat, tai10b.dat, tai12a.dat, tai12b.dat, tai15a.dat y tai15b.dat. Se ha decidido no evaluar las instancias de tamaño mayor a 15 debido a que se esta trabajando con una técnica completa, por lo que, es de esperar que los tiempos de ejecución aumenten drásticamente a medida que aumenta el tamaño de la instancia.

El objetivo del experimento es saber si el algoritmo propuesto es capaz de resolver instancias de tamaño menor o igual a 15 en un tiempo razonable. Esto se hará midiendo el tiempo de ejecución de las distintas instancias y comparar los resultados. El programa se ejecutara en un computador cuya CPU es un Intel Core i5-8400 con una frecuencia base de 2.80Ghz. Se definirá un tiempo limite de 2 horas para la búsqueda de una solución.

8 Resultados

A continuación se presentan los resultados obtenidos al evaluar las distintas instancias:

8.1 Christofides

Instancia	n	óptimo	t
chr12a.dat	12	9552	3923
chr12b.dat	12	9742	4664
chr12c.dat	12	11156	4655
chr15a.dat	15	—	—
chr15b.dat	15	—	—
chr15c.dat	15	—	—

Table 1: Resultados obtenidos de las instancias de Christofides. De izquierda a derecha se tiene: Nombre de la instancia, tamaño de la instancia, óptimo global o solución encontrada y tiempo de ejecución en segundos.

8.2 Nugent

Instancia	n	óptimo	t
nug12.dat	12	578	4673
nug14.dat	14	—	—
nug15.dat	15	—	—

Table 2: Resultados obtenidos de las instancias de Nugent. De izquierda a derecha se tiene: Nombre de la instancia, tamaño de la instancia, óptimo global o solución encontrada y tiempo de ejecución en segundos.

8.3 Taillard

Instancia	n	óptimo	t
tai10a.dat	10	135028	28
tai10b.dat	10	1183760	28
tai12a.dat	12	224416	4714
tai12b.dat	12	39464925	4646
tai15a.dat	15	—	—
tai15b.dat	15	—	—

Table 3: Resultados obtenidos de las instancias de Taillard. De izquierda a derecha se tiene: Nombre de la instancia, tamaño de la instancia, óptimo global o solución encontrada y tiempo de ejecución en segundos.

8.4 Análisis

De los resultados obtenidos se puede ver que, el algoritmo no ha logrado resolver las instancias de tamaño 15 dentro del tiempo limite. Si se comparan los tiempos de ejecución de las instancias de Taillard, se puede observar la enorme diferencia entre la instancia de tamaño 10 y de tamaño 12, por lo que, se concluye que el algoritmo propuesto no es adecuado para resolver instancias grandes de QAP. Por otro lado, se pueden ver tiempos de ejecución similares entre las instancias de tamaño 10 y 12, lo que significa que, el como se generen las instancias no afecta mucho el tiempo de ejecución del algoritmo.

9 Conclusiones

Se puede concluir que el problema de asignación cuadrática (QAP) es un problema muy estudiado debido a la cantidad de aplicaciones en problemas del mundo real, y también, por lo complejo que es (incluso con la tecnología actual) obtener soluciones para valores de n grandes, este ultimo aspecto ya se ha evidenciado en la sección de resultados. Se ha discutido como las técnicas discutidas resuelven el mismo problema, pero utilizando distintas representaciones. Estas teécnicas, las cuales son incompletas, han demostrado ser muy eficientes a la hora de encontrar un resultado óptimo(No necesariamente global), y en base a los resultados obtenidos, se puede concluir que GBJ no es una buena alternativa si se quiere encontrar una solución de calidad para este problema y en poco tiempo.

En lo relacionado con los avances futuros, ya se han desarrollado algoritmos para resolver este modelo utilizando computadores cuánticos como, por ejemplo, el trabajo realizado por Codognet et al. [5] donde se utiliza un algoritmo llamado Quantum Annealing, el cual es análogo al algoritmo Simulated Annealing discutido anteriormente, obteniendo resultados prometedores. Esto abre un mundo de posibilidades para la resolución de problemas complejos como el QAP.

10 Bibliografía

References

[1] Ravindra K Ahuja, James B Orlin, and Ashish Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 27(10):917–934, 2000.

[2] Rainer E Burkard, Eranda Çela, and Bettina Klinz. On the biquadratic assignment problem. *Quadratic Assignment and Related Problems*, 16:117–146, 1993.

[3] Rainer E Burkard, Eranda Cela, Panos M Pardalos, and Leonidas S Pitsoulis. *The quadratic assignment problem*. Springer, 1998.

[4] Nicos Christofides and Enrique Benavent. An exact algorithm for the quadratic assignment problem on a tree. *Operations Research*, 37(5):760–768, 1989.

[5] Philippe Codognet, Daniel Diaz, and Salvador Abreu. Quantum and digital annealing for the quadratic assignment problem. In *2022 IEEE International Conference on Quantum Software (QSW)*, pages 1–8. IEEE, 2022.

[6] Arthur M Geoffrion and Glenn W Graves. Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/lp approach. *Operations Research*, 24(4):595–610, 1976.

[7] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Cowles Foundation Discussion Papers*, (221), 1957.

[8] Christopher E Nugent, Thomas E Vollmann, and John Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations research*, 16(1):150–173, 1968.

[9] Carlos AS Oliveira, Panos M Pardalos, and Mauricio GC Resende. Grasp with path-relinking for the quadratic assignment problem. In *Experimental and Efficient Algorithms: Third International Workshop, WEA 2004, Angra dos Reis, Brazil, May 25-28, 2004. Proceedings 3*, pages 356–368. Springer, 2004.

[10] Thomas Stützle. Iterated local search for the quadratic assignment problem. *European journal of operational research*, 174(3):1519–1539, 2006.

[11] Éric Taillard. Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4-5):443–455, 1991.

[12] Eric D Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location science*, 3(2):87–105, 1995.

[13] Mickey R Wilhelm and Thomas L Ward. Solving quadratic assignment problems by 'simulated annealing'. *IIE transactions*, 19(1):107–119, 1987.