Aproximación GRASP-VND para el problema de asignación cuadrática

Rogelio González, Beatriz Bernábe, Martín Estrada,Antonio Alfredo Reyes
Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación, Puebla, Pue. México
{rgonzalez,mestrada}@cs.buap.mx
{beatriz.bernabe,alfred.reymo}@gmail.com

Resumen. En este artículo se presentan soluciones al problema de asignación cuadrática (Quadratic Assignment Problem - QAP), el cual es clásico en optimización combinatorial y está clasificado como un problema NP-Completo. Este problema consiste en encontrar una asignación óptima de *n* instalaciones a *n* ubicaciones, de tal manera que se minimice el costo de transportación de materiales entre las instalaciones y las ubicaciones. Se debe considerar la distancia entre las ubicaciones, así como el flujo de materiales entre las instalaciones. Para buscar soluciones a QAP, hemos implementado un procedimiento que une metaheurística de un procedimiento denominado Randomized Adaptive Search Procedure Greedy (GRASP), con otro llamado Variable Neighborhood Search Descent (VND).

1 Introducción

El QAP es uno de los problemas de alta complejidad computacional de Optimización Combinatoria (OC) [1] y consiste en encontrar una permutación de asignación óptima de *n* instalaciones a *n* localidades con el propósito de minimizar el costo de transporte, dadas dos matrices simétricas una de distancias y otra de flujos. El QAP fue propuesto por Koopmans y Beckmann en 1957 [4], en 1976 Shani y González probaron que QAP es un problema NP-completo [5]. Hasta hoy sólo se han encontrado soluciones óptimas usando métodos exactos para instancias de tamaño menores que 30 [6]. El QAP aparece en muchas aplicaciones, tales como el diseño de teclados de computadora, la programación de manufactura, el diseño de terminales en aeropuertos y procesos de comunicaciones, entre otras [7, 8]. El algoritmo exacto más popular para resolver el QAP es el de ramificación y acotamiento (Branch and Baund B&B) con algunas variantes [9]. El primer algoritmo exacto de ramificación y acotamiento en paralelo fue propuesto por Roucairol [9] , sin embargo en los últimos años se ha propuesto métodos de búsqueda se soluciones conocidos como metaheurísticas (MH) que son procedimientos de aproximación de propósito general, tales como los algoritmos genéticos, el recocido simulado, búsqueda tabú y GRASP [6] entre otras. La característica principal de una MH es que producen soluciones muy cercanas a la óptima en un tiempo razonable de cómputo. El modelo matemático del QAP como problema de OC es la siguiente:

Sean N= {1,2,..., n} y F= (f_{ij}) y D= (d_{kl}) dos matrices cuadradas de n x n simétricas se trata de encontrar una permutación $p \in \Pi_N$ que minimice $\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\, d_{P(i)P(j)}$

Donde Π_N es el conjunto de todas las permutaciones del conjunto N. Los datos de entrada son f_{ij} representa el flujo de materiales de la planta i a la planta j y d_{kl} es la distancia de la ubicación ciudad k a la ubicación l.

Supongamos que las matrices de flujo y las distancias F y D respectivamente son simétricas, entonces se tiene que $f_{ij} = f_{ji}$ y $d_{kl} = d_{lk}$, además fij = 0 y d_{kl} = 0, para $i = j$, entonces podemos escribir las instancias de datos en una sola matriz que compacte a F y D como sigue:

$$C = \begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} & \cdots & d_{1n} \\ f_{21} & 0 & d_{23} & d_{24} & \cdots & d_{2n} \\ f_{31} & f_{32} & 0 & d_{34} & \cdots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ & & & & 0 & d_{n-1\,n} \\ f_{n1} & f_{n2} & \cdots & \cdots & f_{n\,n-1} & 0 \end{bmatrix}$$

Se encuentra disponible una página en Internet llamada QAPLIB [6], en donde podemos hallar las disertaciones, artículos, resultados e instancias de prueba del QAP más recientes.

1.2 GRASP

GRASP es un procedimiento iterativo en donde cada paso consiste en una fase de construcción y una de mejora. En la fase de construcción se aplica un procedimiento heurístico constructivo para obtener una buena solución inicial. Esta solución se mejora en la segunda fase mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final. [1], Festa y Resende muestran diferentes aplicaciones de GRASP [2].

La palabra GRASP proviene de las siglas de *Greedy Randomized Adaptive Search Procedures* que en castellano sería algo así como: Procedimientos de Búsqueda basados en funciones "Greedy" Aleatorizadas Adaptativas. Las principales componentes de la fase de construcción son: una función de evaluación voraz, un procedimiento de elección al azar y un proceso de actualización adaptativo [3].

```
Procedure GRASP (iter, inicio)
1   Input(instancia);
2       While i ≤  iter  do
3            S0← Construcción (inicio);
4            Sol ← Busqueda_Local(S0)
5            ActualizaSolucion(Sol,);
6               end {while}
8   Return (MejorSol);
End {GRASP}
```

Fig. 1. Pseudocódigo genérico para GRASP

El objetivo de la fase de construcción es generar una solución factible de buena calidad, y entre más eficiente sea esta fase en términos de calidad se espera que el trabajo de la fase de postprocesamiento sea menor y así cada iteración GRASP ocurriría más rápido. En la fase de construcción, GRASP genera una lista de candidatos formada por elementos de alta calidad, esta lista es llamada lista restringida de candidatos (LRC). La solución inicial se construye iterativamente considerando un elemento cada vez. En cada iteración del procedimiento constructivo, un elemento es elegido en forma aleatoria de la lista de candidatos para añadirlo a la subestructura como parte de la solución que se construye. La adición de un elemento a la subestructura se determina mediante una función de tipo voraz, esta función mide el beneficio de la selección del cada elemento, mientras la selección de un elemento de esa lista de candidatos depende de los que se hayan elegido previamente.

```
Procedure Búsqueda Local ( p, V(p), s )
1      While s no sea optima local do
2          Encontrar una  mejor solución t ∈ V(s);
3          Sea s = t ;
4      End ;{ While }
5      Return ( s como optima local )
End {local};
```

Fig. 2. Pseudocódigo genérico para la búsqueda local

1.3 Búsqueda de Vecindad Variable Descendente

La VNS también conocida como búsqueda de entorno variable está basada en un principio simple: cambiar sistemáticamente de estructura de entornos dentro de la búsqueda local. Se han realizado muchas extensiones, principalmente para permitir la solución de problemas de gran tamaño, siempre tratando de conservar la simplicidad del esquema básico. Se han implementado extensiones, híbridos y aplicaciones en

inteligencia artificial [12]. La VNS está basada en tres hechos: un mínimo local con una estructura de entorno no lo es necesariamente con otra, un mínimo global es mínimo local con todas las estructuras de entornos y para muchos problemas, los mínimos locales con la misma o distinta estructura de entornos están relativamente cerca. Esta última observación, aunque empírica implica que los óptimos locales proporcionan información acerca del óptimo global. La VND es una de las variantes de la VNS se pueden considerar descendente [12]. La VNS ha sido incorporada con otras metaherísticas que han dado lugar a diversos híbridos con búsqueda tabú (*Tabu Search* TS), GRASP y búsqueda multi-arranque (*MultiStart Search* MS). Se ha aplicado en diversos problemas de inteligencia artificial como satisfactibilidad, aprendizaje en redes bayesianas y la planificación, también en problemas de optimización de empaquetado, localización problemas de rutas así como en el problema del agente viajero [12]. En la Fig. 3 se da el pseudocódigo de VND.

```
Procedure VND ( p, Vk(p), s )
1 Input(V_k , ∀ k=1,…,k_max)
2    While k < k_max l do
3       Encontrar una  mejor solución t ∈ V_k(s);
4         Sea s = t ;
5          k←k+1
4     End ;{ While }
5    Return ( s como optima local )
End {local}:
```

Fig. 3. Pseudocódigo genérico VND

2 GRASP Y VND para el QAP

2.1 GRASP para el QAP

El diseño de este GRASP ha sido empleado por algunos investigadores para resolver el QAP para diferentes instancias Li, Resende y Pardalos [1]. El pseudocódigo se muestra en la Fig. 1. A continuación se describe la fase de construcción en dos etapas.

Primera etapa. Las dos asignaciones iniciales se hacen simultáneamente, específicamente diremos que el recurso i es asignado a la localidad k y el recurso j es asignado a la localidad l, cuando su costo correspondiente a este par de asignaciones es $f_{ij}\ d_{kl}$.

Sean α, β, ($0 < \alpha, \beta < 1$) los parámetros que restringen la lista de candidatos, $F = (f_{ij})$ y $D = (d_{kl})$ las matrices simétricas $n \times n$ con ceros en la diagonal de entrada con las cuales se forma una matriz compacta cuadrada no simétrica.

Sea $[x]$ la parte entera de x y sea $m = n\,(n\text{-}1)/2$ el número de entradas en los triángulos inferior y superior de la matriz compacta. Se procede a listar estas entradas de las distancias y los flujos en orden creciente y decreciente respectivamente, es decir:

$$d_k\,\mathbf{1}_{l^1} \le d_{k\,2\,l^{\varsigma}} \le d_{k\,3\,l^3} \le \ldots \le d_{k\,m\,l^m}$$

$$f_{i1j1} \geq f_{i2j2} \geq f_{i3j3} \geq \ldots \geq f_{imjm}$$

Ahora tenemos dos listas ordenadas; usaremos el parámetro β para restringir ambas listas, por lo cual se cortan hasta el elemento [βm]. Se genera una nueva lista de costos multiplicando las distancias por los flujos en el orden correspondiente, así, se tiene la nueva lista:

$$f_{i1j1} d_{k1l1}, f_{i2j2} d_{k2l2}, f_{i3j3} d_{k3l3}, \ldots, f_{imjm} d_{kmlm}$$

La cual se ordenada en forma creciente, ahora usamos el parámetro α para obtener la lista restringida y definitiva de los candidatos (LRC), de la cual sólo se tomaran los primeros [αβm] elementos y se elegirá aleatoriamente un elemento f_{ij} d_{kl} que representa un costo de hacer un par de asignaciones (i, *k*) y (*j, l)*, es decir, tenemos dos componentes de la solución, que para simplificar será escrita como permutación, donde la componente *k*-ésima y la componente *l*-ésima son colocadas. Aquí se puede apreciar la componente aleatoria del método. Con esto concluye la primera etapa de la fase de construcción.

Segunda etapa. En esta etapa lo que se pretende es completar la solución inicial calculando las *n*-2 asignaciones restantes, mediante un procedimiento ávido que produce una a una las asignaciones que tienen el costo mínimo con respecto a las asignaciones ya existentes y que en caso de empate se romperá aleatoriamente y apoyándose en una componente adaptativa que se encarga de actualizar la solución a medida que esta se va construyendo.
Sea:

$$\Gamma = \{ (j_1, l_1), (j_2, l_2), ..., (j_r, l_r) \}$$

El conjunto de asignaciones que está en construcción. La etapa 2 inicia con $|\Gamma| = 2$ a consecuencia de los resultados de la etapa1.

Sea $C_{ik} = \sum_{(j,l)\in\Gamma} f_{ij} d_{kl}$ el costo de asignar la fábrica *i* a la localidad *k* con respecto a las asignaciones ya existentes. Seleccionamos de las parejas (*i*, *k*) no asignadas la que tenga el costo mínimo C_{ik}, en esto consiste el procedimiento ávido.

En esta parte también hay una lista restringida de candidatos, se ordenan los C_{ik} en forma creciente y se toma aleatoriamente uno de los primeros [αz], donde z es la cantidad de parejas aun no asignadas, nuevamente aparece la componente aleatoria.

La componente adaptativa de GRASP tiene como función actualizar el conjunto Γ adicionando nuevas parejas asignadas, es decir $\Gamma = \Gamma \cup \{(i, k)\}$

Al finalizar esta etapa concluye también la primera fase, se ha construido una solución contenida en el conjunto $\Gamma = \{ (j_1, l_1), (j_2, l_2), ..., (j_n, l_n) \}$ ordenando las primeras componentes de las parejas, tomamos las segundas componentes para formar la permutación equivalente a la solución. En resumen tenemos una solución de buena calidad para arrancar la segunda fase.

2.2 GRASP_VND para QAP

Empezamos la segunda fase o fase de mejoramiento tomando como solución inicial la solución obtenida en la primera fase de GRASP y en la siguiente fase se emplea la metaheurística de VND con tres estructuras de entorno como adyacente, λ-intercambio y 2-intercambio. En cada estructura vecinal encuentra un mínimo local con la clásica búsqueda descendente que consiste en reemplazar iterativamente la solución actual por el resultado de la búsqueda local, siempre que se obtenga una mejor solución. Así la unión de GRASP-VND produce una aproximación a los óptimos como se discute en la siguiente sección.

3 Resultados y experiencia computacional

Se muestran los resultados fueron obtenidos para doce instancias, propuestas por Nugent [10]. Todos los resultados mostrados se obtuvieron restringiendo la lista de candidatos con los parámetros $\alpha = 0.5$ y $\beta = 0.1$ determinados experimentalmente. Las estadísticas de las tabla se obtuvieron de realizar 10 corridas del programa secuencial, para cada instancia, las tablas contienen: **Problema** : nombre de la instancia, ***n***: tamaño de la instancia **MVC**: mejor valor conocido, **MVE**: mejor valor encontrado por GRASP-VND, **CM**: cota mínima, **PE:** porcentaje de error con respecto a la cota mínima, **PSO**: porcentaje en que se obtuvo la solución óptima o el mejor valor conocido, **Iter**: promedio de iteraciones GRASP-VND en que se obtuvo **MVC**. Finalmente la columna correspondiente a **TCPU** contiene el promedio del tiempo de ejecución de las corridas que alcanzaron el mejor valor de la función objetivo.

Tabla 1. Resultados GRASP-VND para las matrices de Nugent.

Problema	*n*	MVC	MVE	CM	PE	PSO	Iter	TCPU
Nug5	5	25	25	25	0	100	2	0
Nug6	6	43	43	43	0	100	3	0
Nug7	7	74	74	74	0	100	5	0
Nug8	8	107	107	97	9.34%	100	4	0
Nug12	12	289	289	264	8.65%	95	65	0.23
Nug15	15	575	575	542	5.73%	90	67	0.79
Nug20	20	1285	1285	1119	12.91%	100	73	3.61
Nug21	21	1219	1219	1004	17.63%	10	485	32.97
Nug22	22	1798	1798	1417	21.19%	85	250	24.05
Nug24	24	1744	1744	1419	18.63%	70	435	63.89
Nug25	25	1872	1872	1532	18.16%	55	420	56.65
Nug30	30	3062	3062	2886	5.75%	5	529	224.78

Los números en la tabla 1 de la columna MVC son los reportados en la literatura como los valores óptimos desde $n = 5$ hasta $n = 30$, se observa que en todas las corridas se obtuvo el óptimo para las instancias de dimensión 5, 6, 7, 8 y 20 en ambas versiones de la búsqueda local. En la tabla 2 se muestran las permutaciones de asignación de los recursos a las localidades cuyo valor asociado es en todos los casos los de la tercera columna.

Tabla 2. Permutaciones de asignación óptimas o con **MVE.**

Problema	Permutación de mejor asignación
Nug5	3,4,5,1,2
Nug6	6,5,4,3,2,1
Nug7	1,2,5,3,4,7,6
Nug8	3,4,8,2,1,5,6,7
Nug12	5,9,1,8,12,11,3,7,2,10,6,4
Nug15	1,2,7,6,14,13,9,4,5,11,10,15,3,8,12
Nug20	19,7,4,6,17,20,18,14,5,3,9,8,15,2,12,10,16,1,11,13,10
Nug21	3,16,19,21,15,9,4,5,18,10,12,2,17,14,8,11,1,13,6,7,20
Nug22	16,22,17,3,11,9,18,14,20,19,6,8,10,2,1,7,12,4,15,13,21,5
Nug24	7,6,17,23,5,3,9,20,10,8,21,2,4,18,13,11,19,16,14,24,12,15,22,1
Nug25	24,4,13,11,5,18,17,14,19,22,10,6,21,12,20,8,1,3,23,15,7,25,16,2,9
Nug30	17,26,3,7,30,28,15,16,21,22,10,29,27,2,6,9,18,5,14,1,25,11,12,23,13,24,4,19,20,8

4 Conclusiones

Se establece de los resultados que es ventajoso utilizar la estrategia hibrida de GRASP con VND que explora parcialmente una vecindad actualizando la búsqueda cuando encuentra una mejor solución que la actual, esto produce que se disminuya el tiempo de ejecución, sin perder diversificación en la búsqueda y obtener con mayor frecuencia el MVC. También se deduce que la estrategia de evaluar todos los vecinos de una solución retarda cada iteración con lo cual se consume mucho tiempo de CPU.

Existen otras estructuras vecinales más elaboradas con las cuales se puede experimentar como son Corrimiento, *N**, 3-intercambio.

La fase de mejoramiento de GRASP, puede complementarse con un esquema de intensificación como reencadenamiento de trayectorias [11].

La metaheurísticas búsqueda dispersa puede ser desarrollada para resolver el QAP utilizando la fase constructiva de GRASP para generar la población inicial y construir el conjunto referencia.

Referencias

1. Y. Li, P.M. Pardalos, M.G.C. Resende. A Greedy Randomized Adaptative Search Procedure for the Quadratic Assignment Problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, vol. 16 of DIMACS *Series on Discrete Mathematics and Theoretical Computer Sience*, pp 237-261. American Mathematical Society, 1994.

2. P. Festa and M.G. C. Resende. GRASP: An annotated Bibliography. To appear in *Ensays and Surveys on Metaheuristics*. P. Hansen and C.C. Riveiro, eds., Kluwer Academic Publihers, 2000.
3. A.D. Dáz, F. Glober, H. M.Ghaziri, J.L. Gonzalez, P. Moscato, F.T. Tseng. Optimización Heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería. Editorial Paraninfo,1996.
4. T.C. Koopmans,and M.J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, vol 25: pp 53-76,1957.
5. S. Sahni and T. Gonzalez. P-complete aproximations problems. *J. Asssoc. Comp. Machine*. vol. 23 , pp 555-565, 1976.
6. R.E. Burkard, S.E. Karisch ,and F. Rendl, QAPLIB – A Quadratic Assignment Problem, Library, *internet*, http://www.imm.dtu.dk/~sk/qaplib/ins.html.
7. P.M. Pardalos, L.S. Pitssoulis, and M.G.C. Resende. A Parallel GRASP implemetation for the Quadratic Assignment problem. In A. Ferreira and J. Rolim, editors, *Parallel Algorithms for Irregularly Structured Problems – Irregular'94,* pp 111-130. Klower Academic Publishers, 1995.
8. E.L. Lawler. The Quadratic Assignment Problem. *Managnament Sci*., vol. 9, pp 586-599,1963.
9. C. Roucairol. A parallel branch and baund algorithm for the quadratic assignment problem. *Discrete Applied Mathematics*, vol. 18, pp 211-225, 1987.
10. C.E. Nugent, T.E. Vollman, and J. Ruml. An Experimental Comparison of Tecniques for the Assignment of Facilities to locations. *Journal of Operations Research,* vol. 16 : pp 150-173, 1969.
11. M.G.C. Resende, J. L. Gonzalez GRASP: Greedy Randomized Adaptative Search Procedures. *Revista Iberoamericana de Inteligencia Artificial No.19 pp.61-76* (2003).
12. Pierre Hansen, Nenad Mladenovic, Jose Andrés Moreno Pérez. Variable Neighbourhood Search. *Revista Iberoamericana de Inteligencia Artificial* No. 19 pp 79-92. 2003.