Algoritmos evolutivos para solucionar el problema de la mochila multidimensional

Pablo Alberto Osorio Marulanda y *Juan Carlos Rivera*
paosoriom@eafit.edu.co, jrivera6@eafit.edu.co

Departamento de Ciencias Matemáticas
Escuela de Ciencias
Universidad EAFIT
Medellín – Colombia

Resumen

La solución a través de métodos metaheurísticos del problema de la mochila ha sido tratada desde varias perspectivas. La optimización combinatoria es uno de los campos que más se han empleado en trabajar este problema. En este trabajo se expone un algoritmo basado en un método evolutivo, que implementa VND y método aleatorizado GRASP para generar soluciones a 20 instancias de evaluación hipotéticas. Adicionalmente se comparan los resultados obtenidos por los distintos métodos asi como análisis respecto a la variación de los parámetros existentes, ofreciendo finalmente conclusiones respecto a la implementación y posibilidades de mejora. **Palabras Clave:** Heurística, Optimización, Modelación, Problema de la mochila.

1. Introducción

En este artículo se estudia el problema de la mochila, o knapsack problem, el cual es conocido por sus aplicaciones areas desde la medicina hasta finanzas. Este problema pertenece a la clase NP-Hard de problemas [2], algoritmos que debido a su alta complejidad se procuran solucionar mediante métodos heurísticos sobre métodos exactos, dado que no son suficientemente eficaces. La formulación del Knapsack problem o problema de la mochila es la siguiente:

- $w_k =$ El peso de cada item tipo k, para $k = 1, 2, \cdots, N$
- $r_k =$ El valor asociado a cada item tipo k, para$k = 1, 2, \cdots, N$
- $c =$ La capacidad total de la mochila o knapsack

El problema resulta finalmente en

$$\text{Maximize}: \quad \sum_{k=1}^{N} r_k x_k$$

sujeto a:

$$\sum_{k=1}^{N} w_k x_k \leq c$$

donde x_k es una variable no-negativa entera que se refiere al numero de items tipo k que se ingresan en la mochila o knapsack.

Algunas aplicaciones de este problema son pueden ser:

- Selección de proyectos(viendo los proyectos como mochilas)

- Problemas relacionados para detectar problemáticas de corte
- Problemas de distribución de carga
- Problemas de abastecimiento en vehículos de transporte y entrega de productos
- Asignación de procesadores y datos en sistemas distribuidos
- Encontrar equilibrio entre capital y rendimiento financiero
- Selección de oportunidades de inversión
- Maximizar el tiempo de uso de máquinas
- Solución en porblemas de planeación
- Manejo de energía

El planteamiento de este trabajo considera el problema de la mochila multidimensional, en donde se deben tomar decisiones de acuerdo a un conjunto restringido, varias capacidades máximas y por consiguiente una serie de soluciones.
Este trabajo está divido en las siguientes secciones: la primera sección está dedicada a la descripción general del problema acompañada de la formulación matemática respectiva. Posteriormente, se comienza con la siguiente solución, que consiste en la presentación de los algoritmos de solución usados, luego se presentan los experimentos realizados y por último se exponen las conclusiones y las posibles mejoras y aportes futuros del trabajo.

2. Descripción del problema y formulación matemática

El problema multidimensional de define matemáticamente de la siguiente manera: un conjunto P es definido, el cual representa los elementos que se pueden elegir. De este conjunto P se debe escoger un subconjunto de elementos que cumplen las restricciones que se definen a continuación. Adicionalmente, se tiene un conjunto de restricciones b_j, la catidad de recursos que gasta cada elemento por restriccion a_{ij}, cuyo objetivo es no superar la dispoibilidad de recursos. Los elementos son indivisibles y no se puede poner uno dentro de otro. El objetivo del problema considerando su dimensionalidad es maximizar la suma de un conjunto de funciones objetivo (h), lo que en otras palbras significa . La formulación matemática del problema descrito anteriormente es representada por las siguientes ecuaciones, en donde x_i es una variable de decisión binaria, la cual toma el valor $x_i = 1$ si y si se agrega el elemento a la mochila, y $x_i = 0$ en caso contrario.

$$\text{máx } Z = \sum_{h \in P} c_h * x_h \tag{1}$$

$$\sum_{i \in P} a_{ij} * x_i, \forall\ j \in R \tag{2}$$

$$x_{ij} \in \{0, 1\}, \forall\ i \in P \tag{3}$$

La Ecuación (1) representa el conjunto de funciones a maximizar, siendo cada una de estas el beneficio de los elementos seleccionados, la restricción (2) indican el uso de recurso respectivo a las restricciones de capacidad que describe el problema. Finalmente, la ecuación (3) se refiere al dominio de las variables de decisión, que son binarias.

3. Algoritmos de solución

Para la solución del problema presentado se realizó un meétodo basado en un algoritmo evolutivo (tambien llamado algoritmo genético). Sin embargo, este se podría nombrar mejor como un método híbrido, pues usa las soluciones obtenidas a través de un método de búsqueda local VND cuyas soluciones no mejoradas (iniciales) fueron generadas a través de un método aleatorizado Grasp. Veamos con detalle las composiciones de el algoritmo implementado.

3.1. GRASP

El método GRASP implementado en esta práctica puede ser descrito con el siguiente psudocódigo:

1: Se organizan los items en una lista de mayor a menor según el gasto total de recurso
2: Se supone que la solución del problema es ingresar todas los items, esto es $x = 1, 1, 1, \cdots, 1$ donde x es el vector solución para las variables del problema
3: **for** i=1 hasta n(con n número de soluciones requeridas) **do**
4: Se asigna un k, para la estrategia cardinal de seleccion de candidatos
5: Se asigna un α para la estrategia basada en este parámetro de selección de candidatos
6: Se inicializa una lista de candidatos no seleccionados
7: **while** 1 **do**
8: Se realiza un criterio de selección mixto basado en α y k. Se selecciona bajo criterio α y luego, según el número de items dado por este, se hace un filtro según los primeros k candidatos
9: Selecciona aleatoriamente un item de la lista de los candidatos
10: Se remueve el item seleccionado de la solución
11: **if** Se cumplen las restricciones positivas **then**
12: Termina el ciclo y la solución final es la actual
13: **else**
14: Se remueve el item y se elimina este de los no seleccionados
15: **end if**
16: **end while**
17: **end for**
18: Se realiza una selección de las soluciones no dominadas para formar la frontera pareto

Donde el criterio descrito en el paso 1 está representado por la siguiente ecuación:

$$C_i = \sum_{k \in r} a_{ik} \forall i \in j$$

la cual representa el gasto total de cada uno de los items de la mochila, de manera que se procede a seleccionar aquellos aquellos que tengan un mayor gasto total de recurso.
De manera que, basado en un método de selección mixta (cardinal y aleatoria) se genera una lista de posibles items (Rstricted Canditates List, por sus siglas RCL) a eliminar de la mochila, partiendo del hecho de que esta se encuentra llena. De esta manera, se empieza a d̈es-agregar"los elementos que tienen un mayor gasto total de recurso, esto para mejorar la solución iterativamente. Tal lista está descrita de la siguiente manera

$$RCL = \{e \in E | c(e) \geq (1 - \alpha)(c_{max} - c_{min}) + c_{max}\}$$

Una vez generada esta lista, se seleccionan los primeros k candidatos. Una vez se seleccione aleatoriamente el item de esta lista, se elimina como bien se dijo y posteriormente se procede a verificar el cumplimiento de las restricciones para esta nueva solución.
Luego de n iteraciones se genera un conjunto de soluciones aleatroizadas que proceden a mejorarse con el siguiente método.

3.2. VND

VND o *Variable neighborhood descent* es una extensión del algoritmo VNS (Variable neighborhood search), el cual explora distintos vecindarios de una solución y se mueve a uno de ellos si la solución

mejora. Este método amortigua las desventajas de la búsqueda local (Local search), pues al generar distintas estructuras de vecindarios y cambiarlos sistemáticamente escapa de los mínimos locales.
Este algoritmo está basado en 3 hechos:

- Un óptimo local con respecto a una estructura de vecindad no necesariamente lo es respecto a otra
- Un óptimo global es un óptimo local con respecto a todas las posibles estructuras de vecindad
- En muchos problemas los óptimos locales respecto a una vecindad están relativamente cerca

Entendiendo esto, el pseudocódigo que describe esta actividad está comprendido de la siguiente manera:

s -Grasp()
while j¡=3(numero de vecindarios) **do**
    Find $s' \in N_j(s)$
    if s' domina a s **then**
        j=1
        s¡-s
    else
        j=j+1
    end if
end while
$N_1(s)$
Cambiar el 20 % del las variables de manera aleatoria
p=0.2
limit=floor$(p * n)$
s'¡-s
for k=1 to limit **do**
    r= Aleatorio entre 1 y n
    if s'(r)=1 **then**
        s(r)=1
    else
        s(r)=1
    end if
end for
$N_2(s)$
Quitar 2 items y cambiar el 25 % de manera aleatoria por 1
q=0.25
p=2
limit=floor$(q * n)$
s'¡-s
ce=find(s==1)
for k=1 to p **do**
    r= Aleatorio entre 1 y cero
    s'(un(r))=0
end for
for k=1 to limit **do**
    r= Aleatorio entre 1 y n
    if s'(r)=0 **then**
        s(r)=1
    end if
end for
$N_3(s)$
Agregar 2 items y cambiar el 25 % de manera aleatoria por 0
q=0.25

```
p=2
limit=floor(q * n)
s'¡-s
ce=find(s==0)
for k=1 to p do
    r= Aleatorio entre 1 y cero
    s'(un(r))=0
end for
for k=1 to limit do
    r= Aleatorio entre 1 y n
    if s'(r)=1 then
        s(r)=0
    end if
end for
```

De esta manera, para cada solución generada por el GRASP se genera una solución local respecto a los 3 vecindarios. Una vez se evaluaron cada una de las iteraciones respectivas al número de soluciones generadas por el método GRASP se hace una selección de frontera pareto, esto debido a la dimensionalidad de la función objetivo.

3.3. Algorítmo genético

Un algorimo genético es una búsqueda basada en poblaciones, el cual está basado en la dinámica del cambio genético en una población de individuos, esto es:

- Noción Darwiniana de aptitud (fitness) que influye en generaciones futuras
- Apareamiento que produce descendientes en generaciones futuras
- Operadores genéticos que determinan la configuración genética de los descendientes.

Para el algoritmo genético o evolutivo realizado en este trabajo se cosidera el modelo básico descrito por el siguiente pseudocódigo:

```
for i to initial-population-size do
    P[i]=generate-solution()
end for
for i=1 to generations do
    for j=1 to number-of-children do
        (x,y)=selection(P[])
        son[j]= Crossover(x,y)
        if random< −prob-Mutation then
            son[j]=mutation(son[j])
        end if
    end for
    P[]=update(P[])
end for
return better solution in P[]
```

3.3.1. Población inicial

Para la inicialización del método genético se considera una población inicial generada por el método VND mencionado en la sección . Para la generación de esta población se considera un mínimo de población inicial, la cual consiste en la generación iterada de soluciones por VND hasta que se alcanza un límite propuesto de población inicial.

3.3.2. Selección

Para la seleccion de individuos aptos para la reproducción se comienza con una prueba de aptitud. La modalidad de esta selección es de *torneo*, donde se eligen de manera aleatoria una pareja (dos soluciones del conjunto de soluciones) y posteriormente comienzan a competir. En esta competencia gana el miembro más apto, donde el más apto es la solucion que domina a la otra, o en el caso en que ninguna esté dominada, aquella que tenga el valor de la suma de las funciones objetivo más álto. Si se produce un empate nuevamente, el miembro más apto de esta selección se elige de manera aleatoria.

3.3.3. Cruce

Una vez elejidos los dos padres, se comienza el método de cruce, el cual para este caso consiste en la modalidad de *cruce uniforme*. Esta modalidad de cruce consiste en que ambos padres tienen la misma probabilidad de heredar a sus hijos un gen en específico. Para la realización de este método se generó un vector aleatorio, con números entre 0 y 1. Si el número es mayor a 0.5, hereda el gen relativo a la posición en ese vector del padre 1, de lo contrario lo hereda del padre 2. Si bien con esta modalidad de cruce es posible generar fácilmente más de un hijo sin caer en probables redundancias (hijos iguales), se generaron solamente 2 hijos a partir de este cruce para este trabajo.

3.3.4. Mutación

Para la mutación de los hijos resultantes del cruce se aplicó la modalidad de *vecindario*, donde el hijo tiene una baja probabilidad de cambiar su composición genética (soluciones) a través de una solución dada por un vecindario. Para este caso en particular, el vecindario que se utilizó es el mismo relativo al segundo vecindario del método VND mostrado en la sección. Este vecindario está descrito por:

$N_2(s)$
Quitar 2 items y cambiar el 25 % de manera aleatoria por 1
q=0.25
p=2
limit=floor($q * n$)
s'$<-$s
ce=find(s==1)
for k=1 to p **do**
    r= Aleatorio entre 1 y cero
    s'(un(r))=0
end for
for k=1 to limit **do**
    r= Aleatorio entre 1 y n
    if s'(r)=0 **then**
        s(r)=1
    end if
end for

3.3.5. Actualización de la población

Para la selección final de la población que sobrevive para pasar de generación se considera la modalidad de *selección de los mejores individuos entre las dos poblaciones*

3.3.6. Criterio de parada

El criterio de parada relativo a este algoritmo viene representado por el tiempo máximo de cómputo. Este es en general de 5 minutos. En específico uno de los métodos que más se ve afectado por esta restriccion es el método de generación de la población inicial, dado que entre más miembros tenga la

población inicial, más se demora el algoritmo, y por consiguiente le queda menos tiempo para la ejecución apropiada de la reproducción en las posteriores generaciones.

4. Experimentación computacional

Para la experimentación computacional ejecutada para la solución del problema se desarrollaron los algoritmos en el programa MATLAB (The MathWorks, MA) a través de un hardware Intel i5 8th Gen CPU 2.40 GHz, RAM 16 GB, Windows 10.
Los resultados obtenidos por los métodos descritos consideran instancias que corresponden a 18 problemas tomados de (3) y dos restantes propuestos por (2).

4.1. Parametrización del algoritmo

El algoritmo propuesto tienen 2 parámetros importantes. El primero es el componente relativo al número de individuos (número de soluciones) iniciales para dar un comienzo al método. Tal como se mencionó, las soluciones son generadas por un método de búsqueda en vecindarios, que a su vez hacen parte de un método GRASP, por lo que el aumento en la cantidad de individuos impacta de manera directa en el tiempo de cómputo ejecutado. Para experimentos de comprobación se verificaron poblaciones iniciales de 2,4 y 5, los cuales no tinian un impacto considerable en la calidad de las soluciones . Sin embargo es pertinente realizar esta evaluación con más detalle.
Adicinalmente, otro de los parámetros más importantes del método empleado es la tasa de mutación, esto es, la probabilidad de que un hijo mute a través de un método local. Los resultados con entradas pertinentes, considerando el número de soluciones dominadas es el siguiente:

Mutación			
param	0.05	0.1	0.15
S	Número de soluciones no dominadas		
1	3	2	1
2	4	2	1
3	2	2	2
4	1	1	1
5	1	2	1
6	1	1	1
7	2	2	1
8	1	1	1
9	1	1	21
10	4	2	1
11	2	2	1
12	2	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	0	0	0
20	0	0	0

Figura 1: Numero de soluciones dominadas respectivas al un cambio en la mutación

4.2. Comparaciones con otros métodos

En esta sección procedemos a comparar los métodos utilizados comparándolos con entregas anteriores relativas a métodos de busqueda local y método VND. Para analizar esto se tendrán en cuenta dos indicadores, donde dados dos conjuntos de frontera pareto, A y B que se hallan con métodos distintos, se puede definir el conjunto F como el conjunto de soluciones no dominadas resultante de la unión de los conjuntos A y B.

$$I_1 = \frac{|A \cap B|}{|F|}$$

$$I_2 = \frac{|F \cap A|}{|A|}$$

El indicador I_1 calcula la relacion entre el número de soluciones no domindas generadas por cada método y el número total de soluciones en la frontera pareto F, lo que quiere decir que representa la probabilidad de generar soluciones en la Frontera de Pareto.
El indicador I_2 por otra parte calcula la relación entre las soluciones no dominadas generadas por cada método y el número total de soluciones no dominadas generadas por dicho método, lo que significa que representa la probabilidad de generar soluciones no dominadas. Las soluciones de comparación son las siguientes:

	Algoritmo genético	VND	I_1		I_2	
Solución	No dominada	No dominada	Algoritmo genético	VND	Algoritmo genético	VND
1	3	1	1.000	0.000	1	0
2	4	1	1.000	0.000	1	0
3	2	1	0.667	0.333	1	1
4	1	1	1.000	0.000	1	0
5	1	1	1.000	0.000	1	0
6	1	1	1.000	0.000	1	0
7	2	1	0.667	0.333	1	1
8	1	1	1.000	0.000	1	0
9	1	1	0.500	0.500	1	1
10	4	1	0.000	1.000	0	1
11	2	1	0.667	0.333	1	1
12	2	1	0.667	0.333	1	1
13	1	1	0.500	0.500	1	1
14	1	1	1.000	0.000	1	0
15	1	1	0.500	0.500	1	1
16	1	1	0.500	0.500	1	1
17	1	1	1.000	0.000	1	0
18	1	1	1.000	0.000	1	0
19	0	0	0.000	0.000	0	0
20	0	0	0.000	0.000	0	0

Figura 2: Indicador 1 y 2 para comprar algoritmo genético con VND

	Algoritmo genético	Local search	I_1		I_2	
Solución	No dominada	No dominada	Algoritmo genético	Local search	Algoritmo genético	Local search
1	3	1	0.667	0.333	0.66666667	1.00
2	4	4	1.000	0.000	1	0.00
3	2	2	1.000	0.000	1	0.00
4	1	1	1.000	0.000	1	0.00
5	1	1	0.500	0.500	1	1.00
6	1	4	0.250	1.000	1	1.00
7	2	4	1.000	0.000	1	0.00
8	1	1	1.000	0.000	1	0.00
9	1	11	1.000	0.000	1	0.00
10	4	5	0.800	0.200	1	0.20
11	2	5	0.000	1.000	0	1.00
12	2	8	0.111	0.889	0.5	1.00
13	1	19	0.000	1.000	0	1.00
14	1	22	0.000	1.000	0	1.00
15	1	7	0.000	1.000	0	1.00
16	1	9	0.000	1.000	0	1.00
17	1	6	0.000	1.000	0	0.83
18	1	2	0.000	1.000	0	1.00
19	0	0	0.000	0.000	0	0.00
20	0	0	0.000	0.000	0	0.00

Figura 3: Indicador 1 y 2 para comprar algoritmo genético con LS

Cabe resaltar que, el algoritmo hasta la instancia 13 forzaba a el VND a generar soluciones factibles. A partir de allí, se dejó de imponer esta condición para que el tiempo computacional empleado fuera también utilizado en la generación de nuevos hijos y no solamente en la generación de la población inicial.

4.3. Tiempo de cómputo

Considerando que para este caso el tiempo de cómputo no representa un significado analítico dado que es preestablecido como límite, es importante hacer un análisis respecto a la complejidad interna de solucionar cada problema, para ello se calcula el número de generaciones que se hicieron para llegar a las soluciones en cada una de las instancias del problema. Los resultados obtenidos son los siguientes:

Solución	Generaciones
1	10
2	10
3	18
4	11
5	9
6	11
7	15
8	10
9	11
10	9
11	9
12	10
13	10
14	9
15	9
16	9
17	9
18	9
19	0
20	0

Figura 4: Numero de generaciones formadas con el algoritmo genético para las diferentes soluciones

5. Conclusiones

El algoritmo genético implementado tiene una configuración correcta para solucionar un problema de esta especie. El número de generaciones formadas en el tiempo estimado es aproximadamente constante, pues se buscó que con el tiempo de cómputo máximo que se tenía se generaran más de 8 generaciones, para garantizar una mejor solución. Respecto a la mutación, es claro que su afeccion es considerable al momento de la generación de soluciones no dominadas, pues al momento de ampliar su parámetro, la generación de vecindarios que respectan a la mutación de una solución termina generalmente en soluciones no tan apropiadas.
Ahora bien, respecto a la calidad de los indicadores, es necesario decir, que para la comparación del método evolutivo con el método VND, el primero sale altamente beneficiado, cosa que es el resultado esperado, pues el algoritmo implementado busca mejorar las soluciones otorgadas por un algoritmo VND, esto para ambos indicadores.
En el segundo caso, tenemos que el método de búsqueda local es muy bueno en terminos generales, pues tiende a tener más probabilidades de generar la frontera pareto. Pero este efecto es visible más que nada en soluciones desde la número 13 o más. De aqui se nota claramente que haber hecho que el modelo, por cumplir con el tiempo y el número de generaciones estimadas para una nueva solución, no tuviera una población inicial estrictamente factible disminuye la calidad de la solución.
Para futuros trabajos es recomendable realizar varias implementaciones y experimentos, tales como:

- Hacer que desde la instancia 13 se genere una población inicial estrictamente factible.
- Generar poblaciones iniciales más grandes

- Forzar al momento del cruce la generación de hijos que contengan soluciones factibles, esto es, cruzar a los padres tanto como sea necesario para generar una solución factible.
- Hacer que la población sea constante, esto es, si queremos que la población permanezca en 100, y en una iteración los no dominados son 56, generar 44 soluciones más para la siguiente generación y agregarlos a la población.

Referencias

[1] Fernando Sandoya Sánchez , *EL PROBLEMA DE LA MOCHILA, COMPLEJIDAD, COTAS Y MÉTODOS DE BÚSQUEDA EFICIENTES*, Vol.12,No.2, FCNM– ESPOL, DF, 2014.

[2] Juan Carlos Rivera, *Apuntes de curso - Heurística*,2020 (Noviembre)

[3] Chu, Paul C, Beasley, John E.*A genetic algorithm for the multidimensional knapsack problem*Journal of heuristics, 4(1), 63–86.