Andrés Felipe Romero Silva - 201515617

Metaheurísticas - IIND 4109 - Parcial 1

1. (1.5 puntos) David, ya demostró que su problema actual es NP-Hard:

a. Formule un modelo matemático para este problema.

Para resolver este problema se asume que cada una de las caracteríasticas de los estudiantes tiene el mismo peso en la función objetivo. Es decir, estudiantes que tengan calificación [10,0] o [0,10] van a estar igual de alejados de un estudiante [0,0]. Lo anterior sirve para simplificar el modelamiento de la siguiente manera.

I. Conjuntos

Estudiantes: E = { Andrés, Sofía, Javier, ... }

II. Parámetros

$d_{ij} = \ distancia\ entre\ el\ estudiante\ i \in E\ y\ el\ estudiante\ j \in E$

III. Variables

$x_i = \ binaria, 1\ si\ el\ estudiante\ i \in E\ es\ elegidos, 0\ d.l.c.$
$z_{ij} = \ binaria, 1\ si\ los\ estudiantes\ i, j \in E^2\ son\ elegidos, 0\ d.l.c.$

IV. Restricciones

a) Se deben elegir exactamente 5 estudiantes

$$\sum_{i \in E} x_i = 5$$

b) Linealización de la multiplicación de variables binarias

$$z_{ij} \geq \ x_i + x_j - 1$$
$$z_{ij} \leq \ x_i$$
$$z_{ij} \ \leq \ x_j$$

V. Función Objetivo

$$\max \sum_{i \in E} \sum_{j \in E} z_{ij} d_{ij}$$

b. Diga qué clase de problema es su modelo resultante y que algoritmo(s) serían interesante(s) utilizar.

El modelo resultante es un problema restricto lineal con variables reales, binarias.

Sería interesante resolver dicho problema de manera exacta mediante:
- Primal Simplex, Dual Simplex o Punto Interior.
- Algún algoritmo exacto especializado en knapsack, pues en esencia el problema anterior es una mochila unidimensional, en dónde se tiene que elegir una cantidad de estudiantes n de un grupo N con tal de aumentar el beneficio.

Por otra parte, sería interesante resolverlo de manera aproximada con una heurística constructiva (eliminación) donde la inicialización sea un grupo que

contenga todos los estudiantes de la clase (es decir, todos pasan), sin embargo, dicha solución no es factible, pues se necesitan únicamente 5. Entonces, la heurística se encargaría de eliminar secuencialmente un estudiante mediante algún criterio de decisión y una afectación a la función objetivo. Este algoritmo se usa para solucionar un problema de localización logística. También se podría usar el equivalente de adición en dónde se inicia con 0 estudiantes elegidos.

Finalmente, sería interesante definir una metaheurística que gobierne la heurística anterior y haga búsquedas locales y genere zonas aleatorias de exploración para que se puedan encontrar mejores soluciones.

c. Piense un poco sobre la cantidad de posibles soluciones (de un número) y diga que clase de problema cree que es y qué tipo de herramientas de optimización serían útiles.

La cantidad de posibles soluciones es un número combinatorio. Si x es el número de estudiantes que pasan la materia y n el total. Entonces el número de soluciones es nCx, es decir:

$$\frac{n!}{x!\ (n-x)!}$$

Para el problema actual, x = 5 y n = 9, por lo tanto, tenemos un total de **126** posibles soluciones.

Dada la similitud de este problema con el knapsack, yo diría que se trata de un problema NP-Completo con tiempo de resolución polinomial. Este tipo de problemas se puede resolver con optimización exacta para instancias pequeñas, pero a medida que se agranda se vuelve ineficiente y es necesario recurrir a heurísticas/metaheurísticas.

2. Diseñe un algoritmo heurístico constructivo pseudo-aleatorizado que permita alcanzar una solución factible del problema e ilustre cuidadosamente una iteración de su algoritmo.

El algoritmo que diseñé es un constructivo que va eliminando de manera consecutiva el estudiante que resulte en la mejor mejora a la función objetivo. De esta manera, la función objetivo de la formulación se cambia por la siguiente:

$$\frac{\sum_{i\in E}\sum_{j\in E} z_{ij}d_{ij}}{n}$$

Dónde n es el número de estudiantes en el grupo seleccionado. Lo anterior permite que un grupo de 6 estudiantes sea comparable con uno de 5, por ejemplo. Adicionalmente, hay un componente aleatorio (alpha) que consiste en eliminar, no el mejor candidato, sino un estudiante aleatorio del grupo, esto permite tener soluciones diversas y no siempre la misma.

El constructivo funciona de la siguiente manera:

```
function constructive(students, n, alpha):
1.  group ← students
2.      while elements(group) > n:
3.          if  random_number < alpha:
4.              group ← randomElimintation(group)
5.          else
6.              group ← bestElimination(group)
7.  return group
```

La función bestElimination tiene que evaluar la función objetivo para todas las eliminaciones posibles, es decir, lenght(group) y decidir la mejor. El detalle de esto se encuentra en el archivo parcial1.py adjunto.

3. Diseñe un algoritmo metaheuristico tipo GRASP para este problema.

a) Ilustre una iteración de su algoritmo.

El algoritmo grasp que diseñé gobierna el constructivo anterior mediante una busqueda local. El pseudo código es el siguiente:

```
function GRASP(students, n, alpha, maxIters):
8.   bestSol ← empty solution
9.       for i = 1 .. maxIters:
10.          randomSol ← constructive(students, n, alpha)
11.          randomSol ← localSearch(randomSol)
12.          if best(randomSol, bestSol) == randomSol:
13.              bestSol ← randomSol
14. return bestSol
```

La búsqueda local que se diseñó genera un candidato a eliminación, el cual es aquel estudiante que se eliminaría si nos pidieran un grupo de n-1 estudiantes. Una vez este candidato es generado, se cambia por alguno de los estudiantes que no están en el grupo. El detalle de esta función se encuentra en el archivo parcial1.py

b) Explique cómo puede afectar la codificación y definición de vecindad en el desempeño de su algoritmo.

El problema tiene una codificación de conjuntos, en dónde hay un conjunto global de estudiantes, conformado por 2 subconjuntos; el grupo de estudiantes seleccionados y el grupo de estudiantes no seleccionados. Esto puede afectar la velocidad del algoritmo pues cada vez que se elimina o agrega un estudiante se pregunta por la diferencia entre el conjunto en cuestión con respecto a la adición/eliminación de un elemento. Posiblemente una codificación binaria sería la mejor forma de codificar este problema.

Por otra parte, la vecindad es bastante pequeña, se evalúa un vecindario de tamaño (students-n) lo cual podría llevar a optimos locales muy rápidos y que el algoritmo no explore muchas soluciones. Esto se podría solucionar con un constructivo mas aleatorizado, es decir, con un alpha mas grande.