

# **PRÁCTICA IV**

**Algoritmos Backtracking**

**Por: Antonio Gámiz**

# 1 Problema

Se va a celebrar una cene de gala a la que asistirán  $n$  invitados. Todos se van a sentar alrededor de una única gran mesa rectangular, de forma que cada invitado tendrá sentados junto a él a otros dos comensales (uno a su izquierda y otro a su derecha). En función de las características de cada invitado (por ejemplo por categoría, puesto, lugar de procedencia...) existen unas normas de protocolo que indican el nivel de conveniencia de que dos invitados se sienten en lugares contiguos (supondremos que dicho nivel es un número entero entre 0 y 100). El nivel de conveniencia total de una asignación de invitados a su puesto en la mesa es la suma de todos los niveles de conveniencia de cada invitado con cada uno de los dos invitados sentados a su lado.

Se desea sentar a los invitados de forma que el nivel de conveniencia global sea lo mayor posible. Diseñar e implementar un algoritmo vuelta atrás para resolver este problema. Realizar un estudio empírico de su eficiencia.

```

class ConvenienceMatrix
{
    private:
        vector<vector<conv> > c;
    public:
        ConvenienceMatrix(){};
        ConvenienceMatrix(int n)
        {
            c.resize(n);
            for(int i=0; i<n; i++) c[i].resize(n);

            int k=0;
            for(int i=0; i<c.size(); i++)
            {
                for(int j=0; j<c.size(); j++)
                {
                    c[i][j].person=j;
                    c[i][j].convenience=(i!=j) ? rand() % 100 : 0; //""pseudo-random""
                }
                k+=1;
            }
        }
}

```

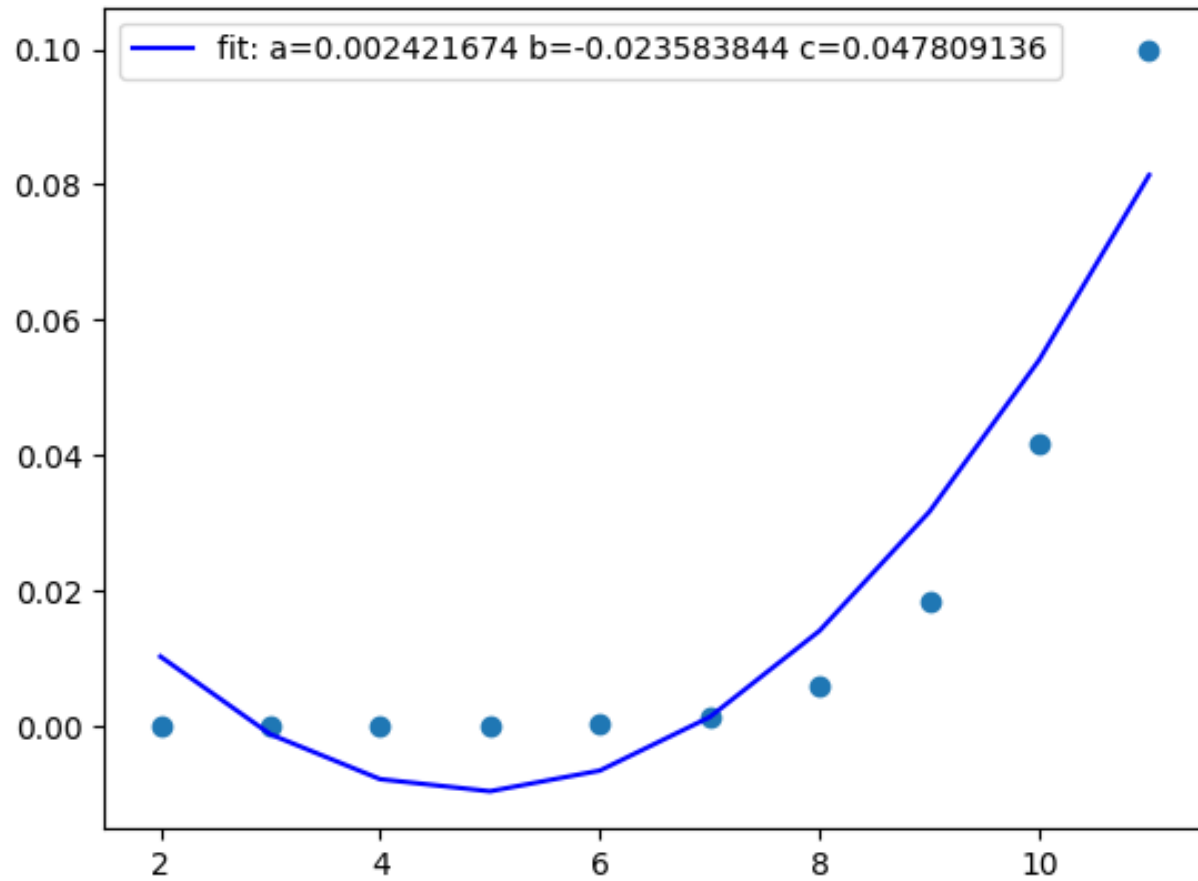
\_\_\_\_\_



\_\_\_\_\_

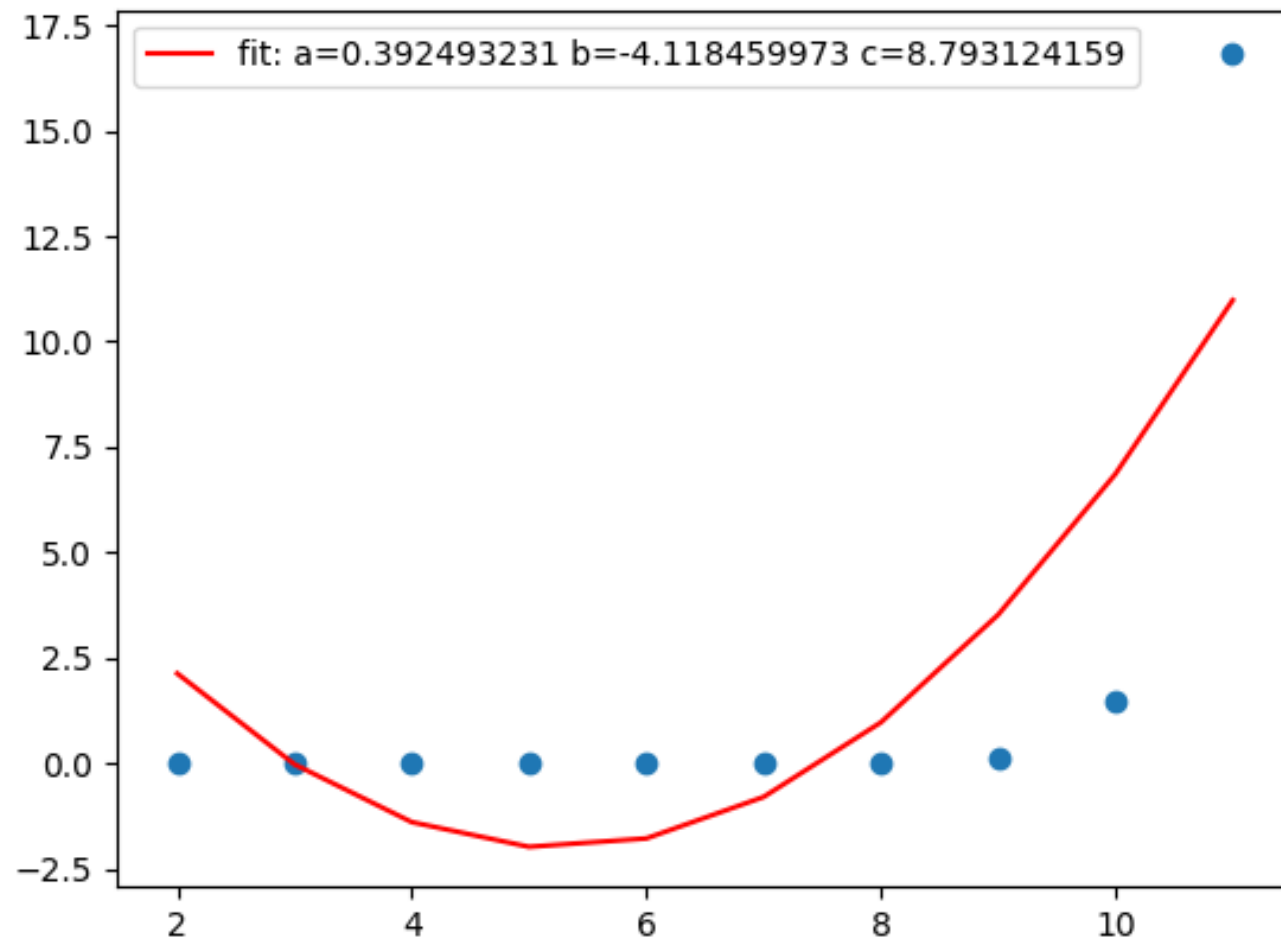


# Método 1

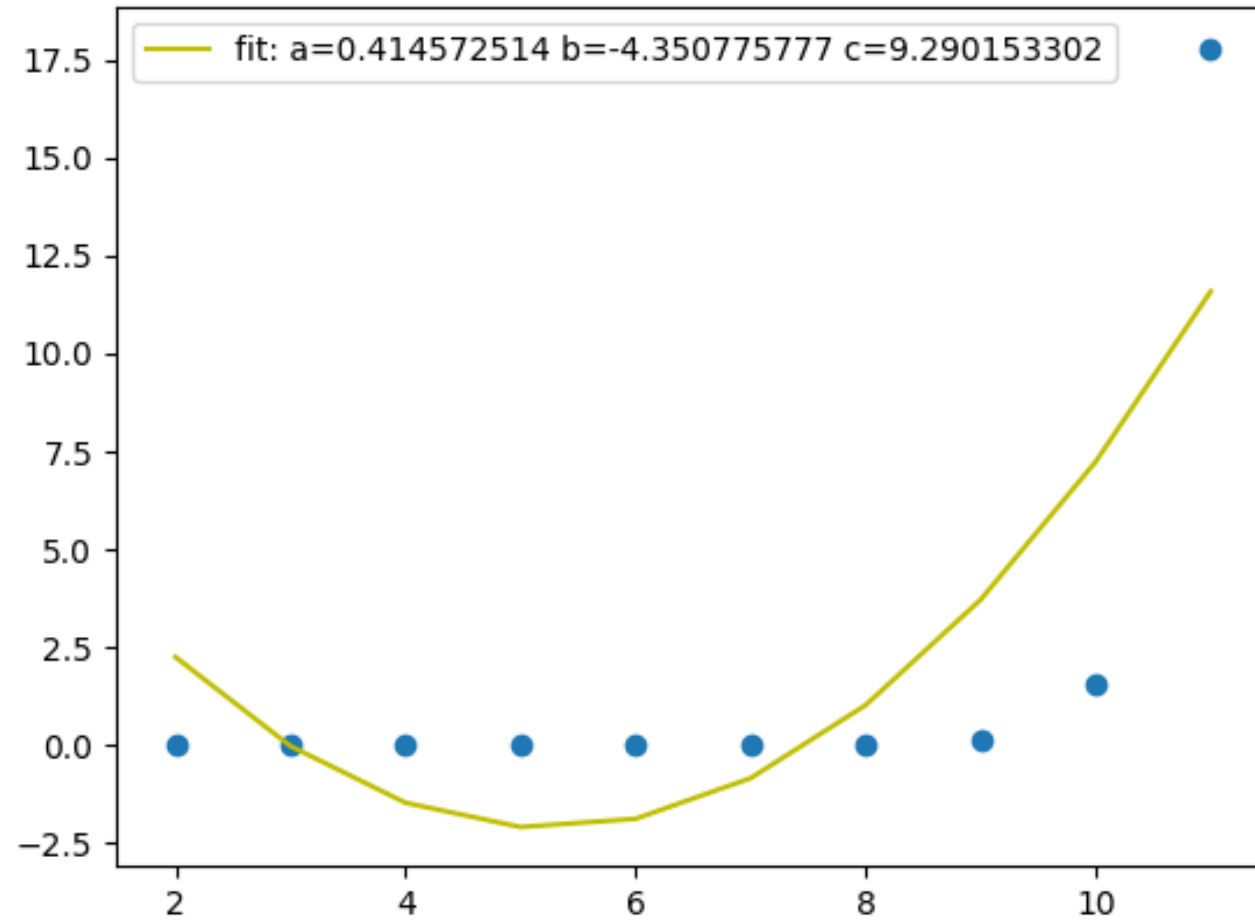


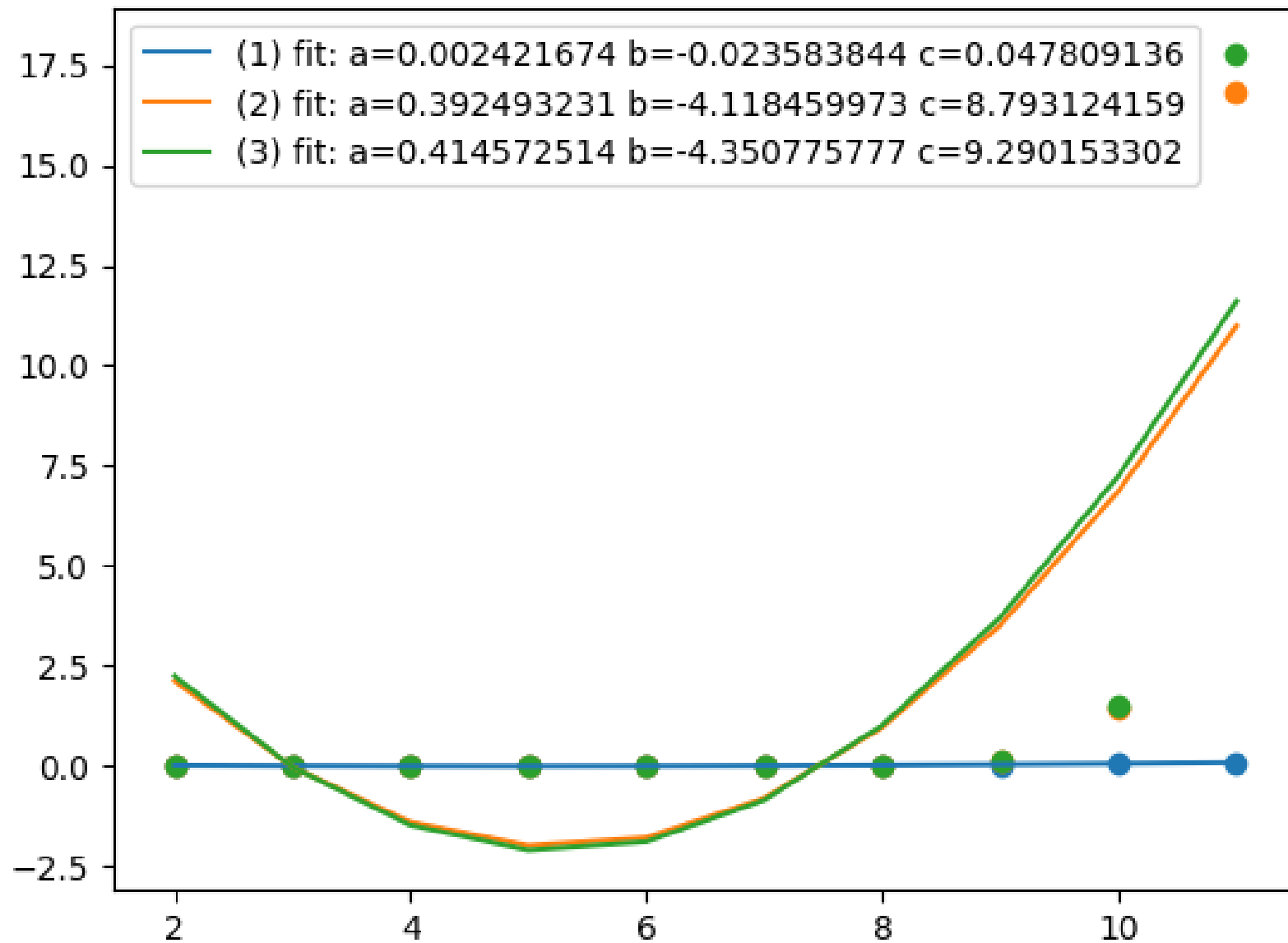



# Método 2



# Método 3







FIN