

Proyecto de Modelos Matemáticos Aplicados

Ana Karla Caballero Gonzales C411

Alejandro Camacho Pérez C412

Contents

1	Introducción	2
2	Cuestiones Generales de los problemas	2
2.1	Datos	2
2.2	Variables	3
2.3	Restricciones comunes	3
3	Sobre la implementación	4
3.1	Módulo de interfaz de usuario	4
3.1.1	Uso de la interfaz de usuario	5
3.2	Módulo de resolución de problemas	5
4	Problemas Resueltos	9
4.1	Problema 1: Maximizar la duración del acto sexual	9
4.2	Problema 2: Maximizar el placer del que menor placer alcance al finalizar el acto sexual	9
5	Consideraciones	9
6	Sobre importar y exportar datos	10

1 Introducción

En una investigación anterior realizada en la facultad de Matemática y Computación de la Universidad de La Habana, MATCOM, se proponía basándose en trabajos anteriores varios modelos matemáticos para modelar y resolver problemáticas relacionadas con el acto sexual. En el mismo se plantean los siguientes 5 problemas:

1. Maximizar la duración del acto sexual.
2. Maximizar el placer del que menor placer alcance al finalizar el acto sexual.
3. Minimizar el cansancio del participante con mayor cansancio al finalizar el acto sexual.
4. Minimizar la energía inicial de todos los participantes de forma que al terminar todos hayan alcanzado el orgasmo y tengan la misma energía.
5. Maximizar el placer inicial de un participante específico, de forma tal que todos los participantes, excepto el específico, alcancen el orgasmo.

El objetivo de este trabajo, es hacer una implementación, utilizando un lenguaje de programación, de los modelos matemáticos propuestos en dicha investigación.

2 Cuestiones Generales de los problemas

La investigación realizada utilizó los siguientes parámetros:

2.1 Datos

- $J = \{1 \dots j\}$: Conjunto de participantes.
- $N = \{1 \dots n\}$: Posturas a adoptar
- A_{ij} : Energía de la persona j al terminar la postura $i \forall i = \overline{1, n}$.
- P_{ij} : Placer de la persona j al terminar la postura $i. i \forall i = \overline{1, n}$. y $i \forall j \in J$.

- \hat{p}_j : Placer necesario para que la persona j alcance el orgasmo. $\forall j \in J$.
- T_i : Tiempo empleado en la postura i . $\forall i = \overline{1, n}$.
- C_{ij} : Cantidad de energía que consume la postura i al participante j por unidad de tiempo $\forall i = \overline{1, n}$ y $\forall j \in J$.
- Pa_{ij} : Cantidad de placer que otorga la postura i al participante j por unidad de tiempo. $\forall i = \overline{1, n}$ y $\forall j \in J$

2.2 Variables

- X_i : Tiempo empleado en la postura i . $\forall i = \overline{1, n}$.
- Pp_{ok} : Placer inicial del participante k , $k \in J$. (Sólo el problema 5)
- E_{0j} : Energía inicial de la persona j . $\forall j \in J$. (Sólo el problema 4)

2.3 Restricciones comunes

- Después de cada postura, la energía disminuye de manera proporcional al tiempo que se permanezca en ella:

$$A_{ij} = A_{(i-1)j} - C_{ij}X_i \quad \forall i \in N \quad \forall j \in J$$

- Después de cada postura, el placer de cada participante aumenta de manera proporcional al tiempo que se permanezca en ella:

$$P_{ij} = P_{(i-1)j} + P_{ij}X_i \quad \forall i \in N \quad \forall j \in J$$

- En todo momento la energía es mayor igual que cero

$$A_{ij} \geq 0 \quad \forall i \in N \quad \forall j \in J$$

- El placer después del acto sexual es mayor o igual al placer necesario para alcanzar el orgasmo

$$P_{nj} \geq \hat{P}_j \quad \forall j \in J$$

3 Sobre la implementación

La implementación del trabajo fue hecha en Python, utilizando la librería `pulp` para la resolución de problemas de programación lineal. El mismo consta de dos módulos principales, uno para la interfaz de usuario y la comunicación del mismo con el programa, y otro para la resolución de los problemas.

3.1 Módulo de interfaz de usuario

La interfaz de usuario ofrece la posibilidad de:

- Crea, editar y eliminar participantes.
- Crea, editar y eliminar posturas.
- Listar participantes y posturas con todos sus detalles.
- Exportar e importar los datos de los participantes y posturas a un archivo de json.
- Seleccionar y resolver uno de los problemas planteados.

```
Options:
0: Create a sexual position
1: Edit a sexual position
2: Delete a sexual position
3: List all sexual positions
4: Create a people
5: Edit a people
6: Delete a people
7: List all people
8: List all people and their pleasure and hardness for each sexual position
9: Choose a problem to solve
10: Export people and positions to a file
11: Load people and positions from a file
Select an option: █
```

Figure 1: Interfaz de usuario

3.1.1 Uso de la interfaz de usuario

La interfaz de usuario sigue las siguientes constantes:

- Para la creación de una nueva postura es necesario sólo indicar su nombre
- Para la creación de un nuevo participante es necesario indicar su nombre, placer necesario para alcanzar el orgasmo y energía inicial. Así cómo también los valores de energía y placer para cada postura.
- Tras modificar un participante, se pedirá actualizar los valores de energía y placer para cada postura.
- Tras añadir una postura si existían participantes, se pedirá actualizar los valores de energía y placer para cada participante.
- La letra c cancela la acción que se esté realizando por completo.

Tras tener al menos un participante y una postura, se puede seleccionar uno de los problemas planteados y resolverlo. El programa entonces pasa a enviar los datos en un formato adecuado al módulo de resolución de problemas.

3.2 Módulo de resolución de problemas

El módulo de resolución de problemas consta de una clase abstracta, `AbstractProblem`, que encapsula el procedimiento para cada problema, utilizando los métodos `objective_function`, `constraints` y `solve`. Cada problema

```

1 class AbstractProblem:
2     def __init__(self, peoples: list[People], positions: list[SexualPosition]):
3         self.J = peoples
4         self.N = positions
5         self.P_t = [people.orgasm_pleasure for people in peoples]
6         self.C = self._build_C()
7         self.Pa = self._build_P()
8
9     def _build_C(self) -> list[list[int]]:
10        C = [[0 for _ in range(len(self.J))] for _ in range(len(self.N))]
11
12        for i, position in enumerate(self.N):
13            for j, people in enumerate(self.J):
14                C[i][j] = people.get_hardness_of_position(position.name)
15
16        return C
17
18    def _build_P(self) -> list[list[int]]:
19        P = [[0 for _ in range(len(self.J))] for _ in range(len(self.N))]
20
21        for i, position in enumerate(self.N):
22            for j, people in enumerate(self.J):
23                P[i][j] = people.get_pleasure_of_position(position.name)
24
25        return P
26
27    def objective_function(self):
28        """
29        The objective function of the problem to solve
30        """
31        raise NotImplementedError("objective_function method must be implemented")
32
33    def constraints(self):
34        """
35        The constraints of the problem to solve
36        """
37        raise NotImplementedError("constraints method must be implemented")
38
39    def solve(self):
40        """
41        Solve the problem
42        """
43        raise NotImplementedError("solve method must be implemented")

```

Figure 2: Clase AbstractProblem

hereda de esta clase abstracta y sobrescribe los métodos mencionados. Para poder resolver un problema, es necesaria la creación de una clase que herede de AbstractProblem y sobrescriba los métodos mencionados.

```

1 def objective_function(self):
2     return lpSum([self.X[i] for i in range(len(self.N))])

```

Figure 3: Función objetivo para el problema 1

```

1 def constraints(self):
2     constraints = []
3
4     # Después de cada postura, la energía disminuye de manera proporcional al tiempo que se permanece en ella
5     for i in range(1, len(self.N)):
6         for j in range(len(self.J)):
7             constraints.append(
8                 self.A[i][j] == self.A[i - 1][j] - self.C[i][j] * self.X[i]
9             )
10
11     # Después de cada postura, el placer de cada participante aumenta de manera proporcional al tiempo que permanece en ella
12     for i in range(1, len(self.N)):
13         for j in range(len(self.J)):
14             constraints.append(
15                 self.P[i][j] == self.P[i - 1][j] + self.Pa[i][j] * self.X[i]
16             )
17
18     # En todo momento la energía es mayor igual que cero
19     for i in range(len(self.N)):
20         for j in range(len(self.J)):
21             constraints.append(self.A[i][j] >= 0)
22
23     # El placer después del acto sexual es mayor o igual al placer necesario para alcanzar el orgasmo
24     for j in range(len(self.J)):
25         constraints.append(self.P[-1][j] >= self.P_t[j])
26
27     return constraints

```

Figure 4: Restricciones para el problema 1

```

1 def solve(self):
2     problem = LpProblem("MaxTime", LpMaximize)
3     self.X = [LpVariable(f"X{i}", lowBound=0) for i in range(len(self.N))]
4
5     self.A = [
6         [LpVariable(f"A{i}{j}", lowBound=0) for j in range(len(self.J))]
7         for i in range(len(self.N))
8     ]
9     self.P = [
10         [LpVariable(f"P{i}{j}", lowBound=0) for j in range(len(self.J))]
11         for i in range(len(self.N))
12     ]
13
14     problem += self.objective_function()
15
16     for constraint in self.constraints():
17         problem += constraint
18
19     solution = problem.solve()
20
21     return solution

```

Figure 5: Resolución del problema 1

```

1 def __init__(self, peoples: list[People], positions: list[SexualPosition]):
2     super().__init__(peoples, positions)
3     self.h = LpVariable("h", cat="Continuous")
4
5 def objective_function(self):
6     return self.h

```

Figure 6: Función objetivo para el problema 2

```

1 def constraints(self):
2     constraints = []
3
4     # Después de cada postura, la energía disminuye de manera proporcional al tiempo que se permanece en ella
5     for i in range(1, len(self.N)):
6         for j in range(len(self.J)):
7             constraints.append(
8                 self.A[i][j] == self.A[i - 1][j] - self.C[i][j] * self.X[i]
9             )
10
11     # Después de cada postura, el placer de cada participante aumenta de manera proporcional al tiempo que permanece en ella
12     for i in range(1, len(self.N)):
13         for j in range(len(self.J)):
14             constraints.append(
15                 self.P[i][j] == self.P[i - 1][j] + self.Pa[i][j] * self.X[i]
16             )
17
18     # En todo momento la energía es mayor igual que cero
19     for i in range(len(self.N)):
20         for j in range(len(self.J)):
21             constraints.append(self.A[i][j] >= 0)
22
23     # Placer mínimo al finalizar el acto sexual
24     for j in range(len(self.J)):
25         constraints.append(self.P[-1][j] >= self.h)
26
27     return constraints

```

Figure 7: Restricciones para el problema 2

```

1 def solve(self):
2
3     problem = LpProblem("MaxMinPleasure", LpMaximize)
4
5     self.X = [LpVariable(f"X{i}", lowBound=0) for i in range(len(self.N))]
6     self.A = [
7         [LpVariable(f"A{i}{j}", lowBound=0) for j in range(len(self.J))]
8         for i in range(len(self.N))
9     ]
10    self.P = [
11        [LpVariable(f"P{i}{j}", lowBound=0) for j in range(len(self.J))]
12        for i in range(len(self.N))
13    ]
14
15    problem += self.objective_function()
16
17    for constraint in self.constraints():
18        problem += constraint
19
20    solution = problem.solve()
21
22    return solution

```

Figure 8: Resolución del problema 2

4 Problemas Resueltos

Ambas soluciones de los problemas resueltos fueron expuestas en las anteriores imágenes. A continuación se detallan los problemas resueltos:

4.1 Problema 1: Maximizar la duración del acto sexual

Se centra en encontrar que tiempo se debe estar en cada postura para que el tiempo del acto sexual sea el mayor posible. Se utilizan las variables X_i , que indican el tiempo que se permanecerá en cada postura y se resuelve el problema:

$$\max \sum_{i=1}^n X_i$$

4.2 Problema 2: Maximizar el placer del que menor placer alcance al finalizar el acto sexual

Este problema busca encontrar que cantidad de tiempo darle a cada postura para que la persona con menor placer acumulado al finalizar el acto sexual tenga el mayor placer posible. Se usan las variables X_i y se resuelve el problema $\max h$, donde h es una nueva variable, que satisface:

$$P_{nj} \geq h, \forall j \in J$$

5 Consideraciones

Primeramente, abarcar todos los problemas planteados en la investigación no fue posible, por lo que se decidió implementar los problemas 1 y 2. La implementación de los problemas 3, 4 y 5 se deja como trabajo futuro.

Los problemas pueden ser modelados teniendo en cuenta la preferencia sexual de cada participante, es decir, si el participante J_i está dispuesto a ejecutar la postura dada con el participante J_j , con $0 \leq i < j \leq |J|$.

6 Sobre importar y exportar datos

Dado que una de las funcionalidades del trabajo es la de exportar e importar los datos de los participantes y posturas a un archivo de json, puede ser útil conocer cómo crear un archivo reconocible por el programa de forma manual. El archivo ha de llamarse data.json y tener la siguiente estructura:

```
{
  "people": [
    {
      "name": "PERSON_NAME",
      "sexual_positions": {
        "POSITION_1": {
          "pleasure": 80.0,
          "hardness": 60.0
        },
        "POSITION_2": {
          "pleasure": 70.0,
          "hardness": 40.0
        }
      },
      "initial_energy": 90.0,
      "initial_pleasure": 50.0,
      "current_energy": 90.0,
      "current_pleasure": 50.0,
      "orgasm_pleasure": 90.0
    }
  ],
  "positions": [
    {
      "name": "POSITION_1"
    },
    {
      "name": "POSITION_2"
    }
  ]
}
```

Siguiendo los convenios de un archivo json, cada persona que se incluya debe estar dentro de "people", y cada posición dentro de "positions". Las llaves que están en mayúsculas son las que se deben cambiar por los valores deseados. Es importante aclarar, que cómo se ve en el ejemplo anterior, cada persona ha de tener los datos de placer y dificultad por cada postura siguiendo la estructura indicada. Los valores numéricos puestos en el ejemplo son arbitrarios y pueden ser cambiados por los valores deseados.