

Sistema de Minimización de Polarización

Optimización mediante Programación Entera Mixta

Andrey Quiceño Iván Francesco Jonathan

Universidad del Valle
Análisis de Algoritmos II

Diciembre 2025

El Problema de la Polarización

Contexto

- Población con n personas distribuidas en m opiniones
- Cada opinión tiene un valor $v_i \in [0, 1]$
- Personas con diferentes niveles de resistencia al cambio

Objetivo

Minimizar la polarización final mediante movimientos estratégicos entre opiniones, sujeto a:

- Presupuesto limitado (c_t)
- Número máximo de movimientos
- Costos variables según resistencia

Modelo Matemático

Parámetros

- n : total de personas, m : opiniones
- p_i : personas en opinión i
- $s_{i,k}$: personas en opinión i con resistencia k ($k = 1, 2, 3$)
- Factores: baja(1.0), media(1.5), alta(2.0)

Variables de Decisión

$$x_{k,i,j} \geq 0 \quad \forall k \in \{1, 2, 3\}, i, j \in \{1, \dots, m\}$$

Número de personas con resistencia k que se mueven de i a j

Función Objetivo y Mediana

Polarización

$$\text{Pol}(p', v) = \sum_{i=1}^m p'_i \cdot |v_i - \text{mediana}(p', v)|$$

Donde p'_i es la distribución final de personas

Cálculo de Mediana

Enfoque acumulativo con restricciones:

- $\text{cum}_1 = p'_1$
- $\text{cum}_i = \text{cum}_{i-1} + p'_i$ para $i \geq 2$
- Posición mediana: $\text{median_pos} = \lfloor (n + 1)/2 \rfloor$
- median_opinion : primera opinión donde $\text{cum}_i \geq \text{median_pos}$
- $\text{median_value} = v_{\text{median_opinion}}$

Restricciones del Modelo

- 1 Capacidad por resistencia:

$$\sum_{j=1}^m x_{k,i,j} \leq s_{i,k} \quad \forall k, i$$

- 2 No auto-movimientos:

$$x_{k,i,i} = 0 \quad \forall k, i$$

- 3 Conservación de población:

$$p'_i = p_i + \sum_{k,j} (x_{k,j,i} - x_{k,i,j})$$

- 4 Costo total:

$$\sum_{k,i,j} r_k \cdot x_{k,i,j} \leq c_t$$

- 5 Movimientos máximos:

$$\sum_{k,i,j} x_{k,i,j} \leq \text{maxMovs}$$

Branch and Bound con Gecode

Estrategia de Búsqueda

- **Variable selection:** first_fail
- **Value choice:** indomain_min
- **Solver:** Gecode

Proceso

- 1 Exploración del árbol
- 2 Branching en $x_{k,i,j}$
- 3 Podas por restricciones
- 4 Actualización de cota
- 5 Convergencia a óptimo

Resultado

Solución óptima garantizada con polarización mínima

Arquitectura del Sistema

Componentes

- `model/Proyecto.mzn`: Modelo MiniZinc
- `input_output/`: Parser I/O
- `gui.py`: Interfaz gráfica
- `scripts/`: Testing y build

Tecnologías

- MiniZinc 2.6+
- Python 3.8+
- Tkinter (GUI)
- PyInstaller (.exe)

Flujo

Archivo .txt → Parser → .dzn → MiniZinc → Resultado → GUI

Validación y Resultados

Batería de Pruebas

- 35 casos de prueba
- Rangos: $n \in [10, 100]$, $m \in [3, 5]$
- Validación automática contra resultados esperados
- Tolerancia: 0.001

Desempeño

- Casos pequeños ($n \leq 20$): < 1 segundo
- Casos medianos ($20 < n \leq 50$): 1-10 segundos
- Casos grandes ($n > 50$): 10-60 segundos
- Timeout configurado: 5 minutos

Interfaz Gráfica

Características

- Diseño moderno con tema cyan/azul oscuro
- Carga de archivos .txt intuitiva
- Visualización de parámetros
- Ejecución asíncrona (no bloquea UI)
- Resultados con formato profesional
- Exportación a archivos

Uso

- 1 Seleccionar archivo de entrada
- 2 Cargar y visualizar datos
- 3 Ejecutar optimización
- 4 Revisar polarización y movimientos
- 5 Guardar resultado

Conclusiones y Trabajo Futuro

Logros

- ✓ Modelo completo y funcional en MiniZinc
- ✓ Implementación robusta con manejo de errores
- ✓ Interfaz profesional y amigable
- ✓ 35 pruebas validadas exitosamente
- ✓ Ejecutable portable para Windows

Extensiones Posibles

- Análisis de sensibilidad de parámetros
- Optimización multiobjetivo
- Visualización gráfica de resultados
- Modelos de predicción temporal

¿Preguntas?