

ADA II - Proyecto 2

Integrantes:

JUAN ESTEBAN CLAVIJO GARCÍA - 2225709

juan.esteban.clavijo@correounivalle.edu.co

MIGUEL TAMI LOBO - 2228409

miguel.tami@correounivalle.edu.co

MATTEO ZULUAGA LASSO - 1824907

matteo.zuluaga@correounivalle.edu.co

STEVEN NARVAEZ - cod

brayan.steven.narvaez@correounivalle.edu.co

Presentado a:

Jesús Alexander Aranda

Juan Francisco Díaz Friez

Análisis y Diseño de Algoritmos II

Universidad del Valle
Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
2024 - II

Descripción del Modelo

El modelo desarrollado aborda el problema de minimizar la polarización en una población, en la que cada individuo posee una opinión cuantificada en un rango de valores. Este modelo, formulado en MiniZinc, se estructura en torno a una matriz de movimientos, en la cual cada celda representa la cantidad de personas que cambian su opinión de una categoría inicial a una categoría de destino. La implementación sigue una serie de restricciones y una función objetivo con el propósito de reducir la polarización general de la población mientras se respetan las limitaciones de costos y número de movimientos, el modelo es en su conjunto un problema de programación lineal entera mixta (MILP) debido a la presencia de variables enteras en la matriz `movimientos`.

Componentes Principales del Modelo

1. Parámetros de Entrada:

- `n` y `m`: Enteros que representan el total de personas en la población y el número de posibles opiniones, respectivamente.
- `opiniones_iniciales`: Vector de enteros que contiene la distribución inicial de personas en cada opinión.
- `valores_opiniones`: Vector de floats que asocia cada opinión con un valor numérico, que representa el posicionamiento en una escala de opinión.
- `costos_extras`: Vector de floats que representa los costos adicionales cuando una opinión de destino estaba inicialmente vacía.
- `costos_desplazamiento`: Matriz de floats que definen el costo de mover una persona entre opiniones.
- `ct` y `maxMovs`: Definen el costo total máximo y el número máximo de movimientos permitidos, un float y un entero respectivamente.

2. **Matriz de Movimientos:** La matriz `movimientos[i, j]` es una variable de decisión que representa el número de personas que se mueven de la opinión `i` a la opinión `j`. Este enfoque matricial permite calcular de manera eficiente los costos totales de cambio y la distribución final de opiniones, con el fin de evaluar la polarización resultante.

3. Restricciones del Modelo:

- **Restricción de Distribución Inicial:** Cada fila de la matriz de movimientos debe ser menor o igual al número de personas que inicialmente poseen cada opinión, para garantizar que no se muevan más personas de las disponibles en cada opinión, Esta es una restricción lineal de tipo entera, ya que involucra variables enteras en la matriz de movimientos.
- **Restricción de Costo Total:** La suma de los costos de todos los movimientos no puede exceder el límite **ct**. El cálculo del costo considera tanto el costo base de **costos_desplazamiento** como el costo adicional de **costos_extras** si el destino estaba inicialmente vacío, Esta es una restricción lineal mixta, ya que incluye términos condicionales y valores enteros en la matriz de **movimientos**.
- **Restricción de Número de Movimientos:** La cantidad total de movimientos entre opiniones no puede superar **maxMovs**, contabilizando cada cambio como un movimiento. Esta restricción es lineal y entera:

4. Cálculo de Opiniones Nuevas y Polarización:

- **Opiniones Nuevas:** La distribución resultante después de los movimientos es calculada para cada opinión, ajustando las personas que se trasladan hacia o desde cada opinión.
- **Mediana de Opinión y Polarización:** Para medir la polarización, el modelo calcula la mediana ponderada de las opiniones, considerando la distribución de las personas en las opiniones nuevas. La polarización total se define como la suma de las distancias absolutas entre cada opinión y la mediana, ponderada por el número de personas en cada opinión.

5. Función Objetivo: La función objetivo minimiza el valor de **Pol**, la polarización total calculada.

Justificación de la Adecuación del Modelo al Problema

Este modelo es adecuado para el problema de minimizar la polarización en la población porque:

- **Representación Completa de Movimientos:** El uso de una matriz de movimientos permite modelar los cambios de opinión de manera precisa y controlar individualmente cada transición entre opiniones. Esto asegura que el modelo pueda explorar todas las configuraciones posibles para minimizar la dispersión de opiniones, manteniendo el control sobre el costo y número de movimientos.
- **Control de Costos y Restricciones:** Las restricciones impuestas en el modelo son adecuadas para el problema planteado, ya que garantizan que la solución obtenida esté dentro de los límites de costo y movimientos permitidos. Este aspecto es fundamental para asegurar que la solución sea práctica y factible en un contexto de recursos limitados.
- **Minimización de Polarización:** La función objetivo está diseñada para reducir la polarización mediante el cálculo de la mediana y la dispersión de opiniones en torno a ella. Esto asegura que la solución no solo minimice los costos de desplazamiento, sino que también logre la mayor cercanía de opiniones posibles.
- **Flexibilidad y Escalabilidad:** La estructura en MiniZinc facilita la modificación del modelo y su escalabilidad para diferentes instancias.

Bondades y Falencias del Modelo

Bondades:

- El modelo ofrece una representación detallada y precisa de las transiciones de opinión, permitiendo una minimización efectiva de la polarización.
- La estructura del modelo es flexible, lo que permite ajustar parámetros como los costos y el número de movimientos, adaptándose así a diferentes contextos de recursos.
- La implementación en MiniZinc permite ejecutar y probar el modelo fácilmente en diferentes configuraciones y con datos de entrada variados.

Falencias:

- Sin heurísticas adicionales, el modelo puede requerir una gran cantidad de tiempo de cálculo en instancias grandes debido al enfoque exhaustivo de *Branch and Bound*.
- La ausencia de mecanismos de priorización de movimientos hacia opiniones centrales puede hacer que el modelo no sea tan eficiente en la convergencia hacia una solución de mínima polarización en situaciones de alta dispersión inicial.

Argumentación sobre Eficiencia y Optimalidad

- **Eficiencia:** La eficiencia del modelo está condicionada por el tamaño del espacio de búsqueda, debido a la matriz de movimientos. Aunque el método *Branch and Bound* utilizado por MiniZinc realiza un podado efectivo en función de los límites de costos y número de movimientos, el tiempo de resolución puede aumentar considerablemente en instancias grandes debido a la ausencia de heurísticas de priorización de variables. Para mejorar la eficiencia en estos casos, sería beneficioso implementar estrategias de búsqueda como `int_search` para enfocar la exploración en movimientos hacia opiniones de menor valor.
- **Optimalidad:** El modelo garantiza la optimalidad en la minimización de la polarización bajo las restricciones establecidas. El enfoque de *Branch and Bound* asegura que el solver explore exhaustivamente el espacio de soluciones, encontrando una configuración de mínima polarización que cumple con las limitaciones de costo y número de movimientos. Por tanto, el modelo es óptimo dentro del marco de las restricciones impuestas, aunque podría beneficiarse de ajustes para mejorar la velocidad de convergencia en instancias más grandes.

Detalles Importantes de Implementación

La implementación de este modelo en MiniZinc se centra en minimizar la polarización dentro de una población mediante un sistema de movimientos controlados entre opiniones. Cada componente del modelo se estructura matemáticamente para lograr una representación precisa de las transiciones de opinión y asegurar la optimización de costos y la dispersión mínima.

1. Matriz de Movimientos como Representación de Transiciones

El modelo emplea una matriz de variables, `movimientos[i][j]`, que indica el número de personas trasladadas de la opinión i a la opinión j . Esto permite controlar los cambios en cada categoría de opinión individualmente.

- **Restricción de Distribución Inicial:** Para garantizar que no se muevan más personas de las disponibles en cada opinión, se implementa la siguiente restricción para cada fila i de la matriz `movimientos`:

$$\sum_{j=1}^m movimientos[i][j] \leq opiniones_iniciales[i], \forall i \in \{1, \dots, m\}$$

Esta condición asegura que el total de personas movidas desde una opinión i no exceda el número inicial de personas con esa opinión, respetando así la distribución original de la población.

2. Cálculo del Costo de Cada Movimiento

El costo total de todos los movimientos se calcula considerando dos factores: el costo base del movimiento entre opiniones y un costo extra si la opinión de destino estaba inicialmente vacía. Para cada movimiento de i a j :

- **Costo Base:** Si la opinión de destino j ya tiene personas, el costo de mover una persona desde i a j es simplemente `costos_desplazamiento[i][j]`.
- **Costo Extra para Opiniones Vacías:** Si la opinión de destino j estaba vacía (es decir, si `opiniones_iniciales[j] = 0`), se debe añadir el costo extra `costos_extras[j]`.

El costo total de los movimientos es entonces:

$$Costo_total = \sum_{i=1}^m \sum_{j=1}^m movimientos[i][j] \cdot (costos_desplazamiento[i][j] + (opiniones_iniciales[j] = 0 \Rightarrow costos_extras[j]))$$

donde el término condicional añade el costo extra solo si el destino j estaba vacío inicialmente.

3. Limitación de Costos y Número de Movimientos

Para asegurar que la solución cumpla con las restricciones del problema, el modelo limita el costo total y el número total de movimientos. Estas restricciones son:

- **Costo Total Permitido:** La suma de todos los costos de los movimientos no debe exceder el límite máximo permitido ct :

$$Costo_total \leq ct$$

- **Número Máximo de Movimientos:** Cada cambio de una opinión iii a una opinión jjj cuenta como un "movimiento", y el total de movimientos no debe superar $maxMovs$. La suma de movimientos se calcula como:

$$\sum_{i=1}^m \sum_{j=1}^m movimientos[i][j] \cdot |i - j| \leq maxMovs$$

donde $|i - j|$ representa la distancia entre las opiniones y permite contabilizar los movimientos necesarios para lograr la distribución final.

4. Distribución Final de Opiniones y Cálculo de Polarización

La nueva distribución de opiniones, $opiniones_nuevas$, se calcula sumando las personas que llegan y restando las personas que salen de cada opinión iii , lo que permite obtener el estado final de opiniones en la población:

$$opiniones_nuevas[i] = opiniones_iniciales[i] - \sum_{j=1}^m movimientos[i][j] + \sum_{k=1}^m movimientos[k][i], \forall i \in \{1, \dots, m\}$$

Para calcular la polarización, primero se obtiene la mediana ponderada de los valores de opinión en la población, basada en la distribución $opiniones_nuevas$. La polarización se

mide como la suma de las diferencias absolutas entre cada valor de opinión y la mediana, ponderada por el número de personas con esa opinión:

$$Pol = \sum_{i=1}^m opiniones_nuevas[i] \cdot valores_opiniones[i] - mediana$$

La función objetivo minimiza este valor de polarización (**Pol**), buscando una configuración en la que las opiniones estén lo más cerca posible de la mediana.

Análisis de los Árboles Generados y Funcionamiento del Mecanismo de Branch and Bound

Estructura del Árbol de Decisiones

En este modelo, la variable clave es la matriz **movimientos[i][j]**, que representa el número de personas que se trasladan de una opinión *i* a una opinión *j*. Dado que el solver debe decidir el valor de cada celda en esta matriz (es decir, cuántas personas se mueven entre cada par de opiniones posibles), el árbol de decisiones se construye explorando todas las combinaciones posibles de valores en estas variables de movimiento.

Cada nivel en el árbol de *Branch and Bound* representa una decisión sobre el valor de una variable en **movimientos**, y cada nodo en el árbol corresponde a una posible asignación de valores en la matriz de movimientos. A medida que el solver explora el árbol, va construyendo configuraciones de movimientos que satisfacen las restricciones de costos y de número de movimientos, evaluando el impacto de cada decisión en el valor de la función objetivo, **Pol**, que mide la polarización total.

Funcionamiento del Mecanismo de Branch and Bound

1. Fase de Branching (Ramificación):

- En cada paso, el solver toma una variable **movimientos[i][j]** que aún no ha sido asignada y le asigna un valor posible dentro de su dominio. Dado

que nuestro solver no tiene heurísticas, se sigue un orden predeterminado (por defecto en MiniZinc), sin priorizar una variable específica.

- Esto implica que el solver explora cada variable en el orden en que aparece en el modelo, y para cada variable, intenta todos los valores permitidos (de 0 hasta `opiniones_iniciales[i]`, ya que la cantidad de personas movidas no puede superar la cantidad inicial en cada opinión).

2. Evaluación de Ramas (Bound):

- Cada vez que el solver completa una asignación de valores para las variables de `movimientos`, calcula el valor de la función objetivo `Pol` para esta configuración.
- Si el valor de `Pol` para una configuración dada es mayor que el valor de la mejor solución conocida (si existe), el solver descarta toda la rama bajo ese nodo, ya que sabe que cualquier solución derivada de este camino no será mejor que la mejor solución encontrada hasta el momento. Este es el proceso de *Bound*, que permite al solver evitar explorar soluciones subóptimas, reduciendo así el tamaño del árbol de búsqueda.

3. Podado de Ramas (Pruning):

- Al explorar una rama, si el solver detecta que una restricción de costos o de número de movimientos ha sido violada, esa rama es "podada" inmediatamente. Esto significa que el solver descarta la exploración de todas las asignaciones adicionales derivadas de ese nodo, ya que cualquier extensión de esa solución sería inviable.
- Este podado es crucial para reducir la cantidad de nodos que el solver necesita explorar, acelerando la búsqueda de la solución óptima. En este modelo, las restricciones de `ct` (costo total máximo permitido) y `maxMovs` (máximo número de movimientos) contribuyen significativamente al podado, ya que limitan rápidamente las configuraciones que exceden estos valores.

Comportamiento Observado en el Ejemplo

Para el ejemplo específico que hemos considerado, el árbol generado mostró un patrón claro: el solver exploró las configuraciones iniciales sin aplicar ninguna heurística, verificando secuencialmente cada asignación en `movimientos` y podando ramas a medida que se violaban restricciones de costos o movimientos. Sin una heurística que priorice ciertos

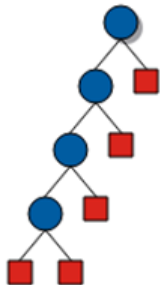
Árbol generado por la instancia 22.

Es así que, para ilustrar esta técnica hemos generado un árbol de menor tamaño correspondiente al archivo “*MinPol22.mpl*” donde cada nodo visualiza la progresión de las posibles asignaciones de movimientos. En este ejemplo, los nodos azules muestran las decisiones iniciales, donde se exploran pocas opciones de movimiento realizando el algoritmo **Branch and bound**, se ramifica para buscar soluciones en distintos rangos para la variable en cuestión. Podemos observar cómo los nodos se dividen en algunas ramas, y algunas llegan a un círculo rojo (hoja terminal) al incumplir con alguna de las restricciones como la de **MaxMvs**. El nodo circular **naranja** representa una solución que cumple con todas las restricciones, aunque no necesariamente es la óptima. Este árbol permite observar cómo el solver filtra progresivamente las asignaciones ineficientes, enfocándose en las soluciones prometedoras y manteniendo el costo de búsqueda bajo control.

Tenemos otro pequeño ejemplo en donde el árbol generado fue sólo esto:

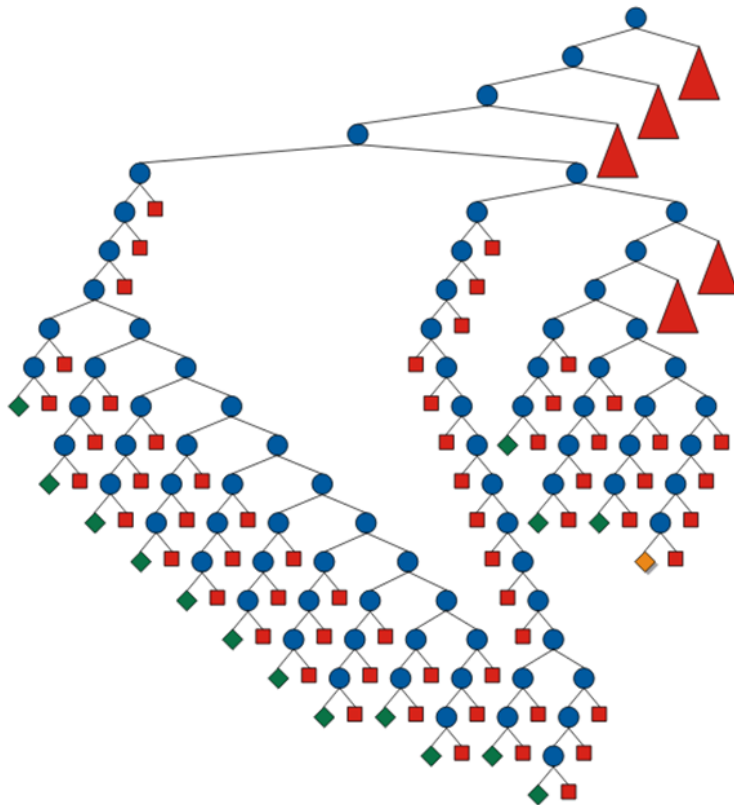


Sabemos que cuando obtenemos un **triángulo rojo** significa que en toda esa rama no se halló ninguna solución que satisficiera las condiciones del modelo, pero si damos “clic” en un triángulo se expandirá la **rama** que se revisó:



Es así, que para esta instancia no se hicieron muchas ramificaciones, y en las pocas que se hicieron rápidamente el solver observa que no se cumplen las restricciones, por lo que en todo el árbol no se haya solución, y sólo se encuentran nodos hoja de círculo rojo, es por esto que todo el árbol se puede resumir en un solo **triángulo rojo**.

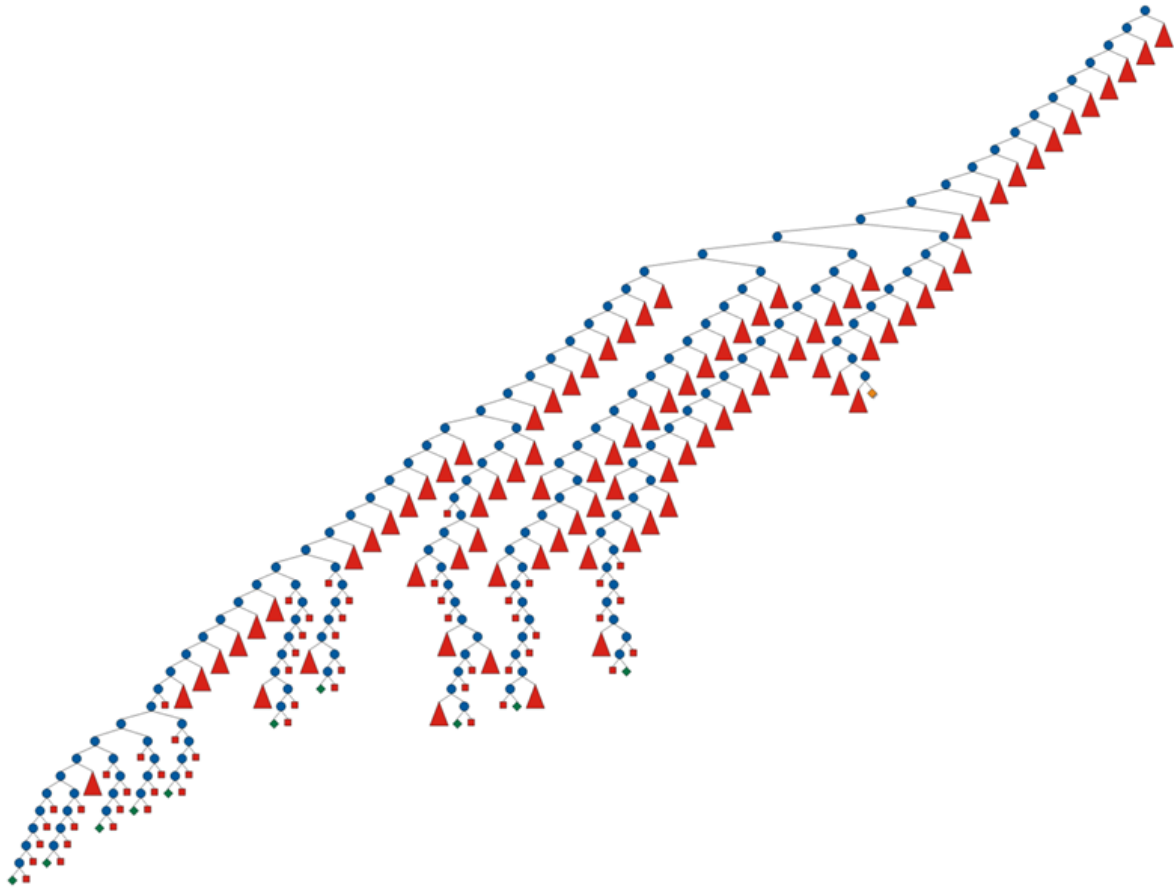
Ejemplo de un árbol de búsqueda más complejo



Árbol generado por la instancia

27.

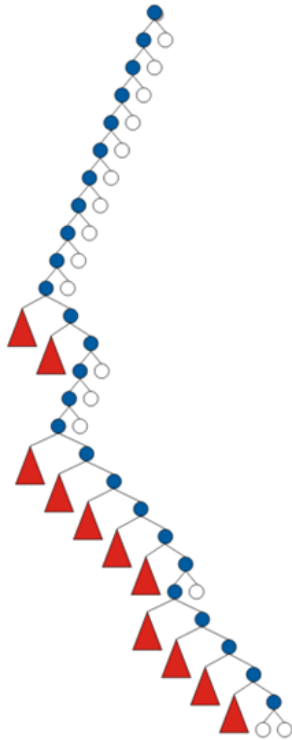
Continuando, en un árbol de búsqueda más grande, generado para un caso de prueba con un mayor número de opiniones y personas, se observa un esquema de búsqueda más amplio y profundo. El visualizador de MiniZinc ilustra cómo el solver navega un espacio de soluciones extensivo, y cómo aplica el criterio de **branch and bound** en cada etapa. En este caso, la mayor densidad de nodos y las múltiples podas indican una alta complejidad en la búsqueda de soluciones. Los nodos **verdes** representan soluciones viables, y es visible cómo la poda de ramas innecesarias (triángulos rojos) optimiza la búsqueda, reduciendo el número de nodos explorados y acercando al solver más rápidamente a soluciones óptimas, notese que el solver no debe detenerse cuando encuentra una solución viable (círculo verde), sólo continúa encontrando así más soluciones.



Árbol generado por la instancia 3.

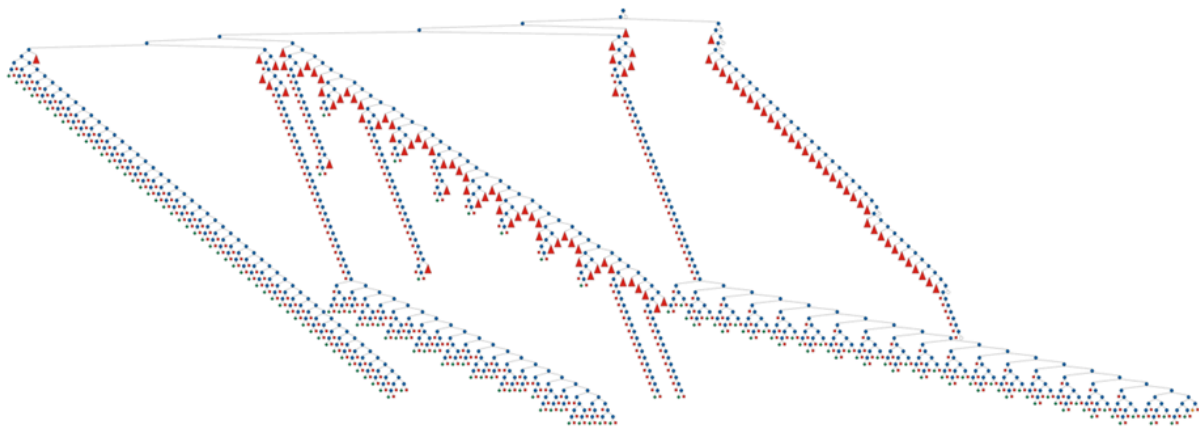
Ya que el problema considera instancias con muchas personas, muchas opiniones o ambas, se puede llegar a árboles tan extensos y profundos como éste dado por la instancia 3

“*MinPol3.mpl*”, donde vemos que encuentra varias soluciones, pero también poda una gran parte del árbol (por los **triángulos rojos**) permitiendo una mejor visibilidad de la búsqueda y el trabajo que realiza el solver.



Árbol generado por la instancia 28 en +20 minutos.

Terminando con esta sección, hay algunas instancias que por tener una gran cantidad de personas y opiniones se llega a un árbol como éste dado por la instancia 28, el cual se ve pequeño, pero el solver se estuvo ejecutando por más de 20 minutos y aún así no terminó de construir el árbol y por ende ver todas las soluciones, ya que también hay que recordar que los triángulos rojos son **ramas podadas**, por lo cual todos esos triángulos son ramas que fueron muy extensas y que a pesar que el solver se tomó el tiempo de revisarlas no encontró ninguna solución, y sólo perdió una gran cantidad de tiempo. También tener en cuenta que los nodos blancos son aquellos que no se han expandido, es decir, no se ha revisado si se debe hacer una **ramificación**, si se incumplen las restricciones y es un nodo **rojo**, o si por el contrario es una solución (óptima o no), por lo cual son nodos que aún quedaban por revisar, dando a entender que aún con todo el tiempo usado le quedaba mucho trabajo.



Árbol generado por la instancia 30 en 60 minutos.

Para terminar, tenemos este árbol generado por la instancia 30, como se ha dicho, instancias con muchas personas y opiniones hace que se tengan que revisar más casos, más ramificaciones, y por tanto el solver se tarde más realizar una búsqueda completa, en 60 minutos de ejecución este fue el árbol que fue generando, que a pesar de verse masivo, tiene nodos podados, por lo que fue mucho más grande la búsqueda que había realizado el solver hasta ese momento, y a pesar de tener tanto tiempo, no había conseguido terminar aún.

Conclusión sobre árboles de búsqueda


En conclusión, en este análisis, se ha evidenciado cómo el enfoque de **branch and bound** implementado por el solver de MiniZinc resulta efectivo para resolver problemas de optimización en el modelo realizado para el problema de *Polarización* con restricciones. Los ejemplos visualizados muestran claramente cómo la estructura del árbol de búsqueda facilita la toma de decisiones eficientes, explorando únicamente las ramas que tienen potencial de llevar a una solución óptima.

Sin embargo, como se observó en las instancias de mayor tamaño, el rendimiento del algoritmo se ve afectado significativamente a medida que aumenta el número de personas y opiniones. Este crecimiento en las variables expande el espacio de búsqueda exponencialmente, haciendo que el solver necesite mucho más tiempo y recursos para completar la búsqueda. En algunos casos, incluso dándole mucho tiempo para la ejecución, el árbol de búsqueda quedaba incompleto, reflejando las dificultades al problema de escalabilidad en la técnica de **branch and bound**.

Es así que, este estudio ilustra la efectividad y las limitaciones del método en escenarios complejos, donde la capacidad de poda es esencial para optimizar tiempos de ejecución, pero también revela cómo la expansión rápida del árbol en instancias grandes plantea un desafío para alcanzar soluciones en tiempos razonables.

En resumen, mientras que el solver "gecode gist" y el método **branch and bound** son herramientas robustas para optimizar la polarización en casos manejables, las instancias de

mayor escala demandan exploraciones adicionales para mejorar la eficiencia computacional o considerar métodos alternativos de optimización.

Pruebas, graficos, tablas:  tests_results

Conclusion final.

En conclusión, el modelo desarrollado para minimizar la polarización en una población utilizando una matriz de movimientos demuestra ser eficaz al ofrecer una representación detallada de las transiciones de opinión. El enfoque de branch and bound implementado en MiniZinc facilita la búsqueda de una solución óptima al explorar únicamente las ramas con potencial de cumplir con las restricciones de costos y movimientos. Sin embargo, a medida que el número de personas y opiniones aumenta, el rendimiento del modelo se ve comprometido, ampliando el espacio de búsqueda y exigiendo mayores recursos de tiempo y procesamiento. Esto resalta la efectividad del modelo para instancias manejables, mientras que en casos de mayor escala podrían explorarse técnicas adicionales o métodos alternativos para optimizar la eficiencia computacional.