Taller 1 — Modelado y resolución de CSP en MiniZinc

Estudio de Sudoku, Kakuro, Secuencia Mágica, Acertijo Lógico, Reunión y Rectángulo

John Freddy Belalcazar
Samuel Galindo Cuevas
Nicolas Herrera Marulanda

1 de octubre de 2025

Índice

1. Sodoku

El *Sudoku* es un rompecabezas lógico en una cuadrícula 9×9 dividida en nueve cajas 3×3. El tablero se entrega con algunas celdas ya llenas (*pistas*) y el objetivo es completar todas las celdas con dígitos del 1 al 9 cumpliendo simultáneamente: (i) en cada fila no se repiten dígitos, (ii) en cada columna no se repiten, y (iii) en cada caja 3×3 no se repiten.

El *Sudoku clásico 9×9* puede modelarse naturalmente como un *Problema de Satisfacción de Restricciones* (CSP): cada celda es una variable con dominio $\{1, \ldots, 9\}$, y las reglas del juego se expresan como restricciones sobre filas, columnas y subcuadrículas 3×3.

1.1. Modelo

Parámetros

P1 — N**:** Tamaño del tablero (lado). En el Sudoku clásico, $N = 9$.

$$N = 9.$$

P2 — S**:** Índices válidos de filas y columnas.

$$S = \{1, \ldots, N\}.$$

P3 — DIG**:** Conjunto de dígitos permitidos en cada celda.

$$DIG = \{1, \ldots, N\}.$$

P4 — G**:** Matriz de *pistas*; 0 indica celda vacía y un valor en DIG fija la celda.

$$G \in \{0, \ldots, N\}^{S \times S}, \qquad G_{r,c} = 0 \text{ (vacía)}, \quad G_{r,c} \in DIG \text{ (valor fijo)}.$$

Variables

V1 — $X_{r,c}$**:** Valor de la celda en fila r y columna c.

$$X_{r,c} \in DIG, \qquad (r, c) \in S \times S.$$

Restricciones principales

R1 — **Fijación de pistas:** Las pistas son hechos inmutables: si hay pista en (r, c), la celda queda fijada.

$$\forall (r, c) \in S \times S : \quad (G_{r,c} > 0) \;\Rightarrow\; (X_{r,c} = G_{r,c}).$$

R2 — **No repetición por fila:** En cada fila, los nueve valores deben ser todos distintos.

$$\forall r \in S : \quad \mathit{all\_different}([\, X_{r,c} \mid c \in S \,]).$$

R3 — **No repetición por columna:** En cada columna, los nueve valores deben ser todos distintos.

$$\forall c \in S : \quad \mathit{all\_different}([\, X_{r,c} \mid r \in S \,]).$$

R4 — **No repetición por caja** 3×3**:** En cada subcuadrícula 3×3, los nueve valores deben ser todos distintos.

$$\forall b_r, b_c \in \{0, 1, 2\} : \quad \mathit{all\_different}([\, X_{3b_r+i,\, 3b_c+j} \mid i, j \in \{1, 2, 3\} \,]).$$

Restricciones redundantes

R5 — Suma por fila = 45: Cada fila contiene los dígitos 1.,9 sin repetición; por tanto su suma es 45. Aporta poda lineal cuando faltan pocas celdas.

$$\forall r \in S : \sum_{c \in S} X_{r,c} = 45.$$

R6 — Suma por columna = 45: Análogo para columnas; refuerza la propagación vertical.

$$\forall c \in S : \sum_{r \in S} X_{r,c} = 45.$$

R7 — Suma por caja 3×3 = 45: Cada subcuadrícula 3×3 reúne los nueve dígitos sin repetición; su suma también es 45. Útil para cerrar cajas cuando faltan pocas celdas.

$$\forall\, b_r, b_c \in \{0,1,2\} : \sum_{i=1}^{3} \sum_{j=1}^{3} X_{\,3b_r+i,\; 3b_c+j} = 45.$$

1.2. Implementación

Archivos y organización

- `sudoku.mzn`: modelo completo.
- `tests/*.dzn`: instancias con la matriz G (0 = vacío).

Modelo

Se construye el conjunto de ramificación $\mathcal{B} = \{\, X_{r,c} \mid G_{r,c} = 0 \,\}$ para explorar únicamente celdas no fijadas por las pistas. Esta elección evita trabajo inútil sobre variables ya determinadas, concentra la búsqueda en la parte incierta del tablero y reduce el árbol sin alterar la corrección: toda solución factible coincide con G y difiere únicamente en $\mathcal{B}$.

Restricciones redundantes

Se activan de manera permanente porque fortalecen la propagación sin alterar el conjunto de soluciones. En primer lugar, la *suma por fila* fija que cada fila contiene los dígitos 1.,9 sin repetición, por lo que su suma es 45; esto aporta poda lineal cuando faltan pocas celdas:

$$\forall r \in S : \quad \sum_{c \in S} X_{r,c} = 45.$$

En segundo lugar, la *suma por columna* refuerza la propagación vertical con el mismo argumento:

$$\forall c \in S: \quad \sum_{r \in S} X_{r,c} = 45.$$

Por último, la *suma por caja* 3×3 impone el mismo invariante en cada subcuadrícula, útil para cerrar cajas cuando restan pocos valores:

$$\forall\, b_r, b_c \in \{0,1,2\}: \quad \sum_{i=1}^{3} \sum_{j=1}^{3} X_{\,3b_r+i,\,3b_c+j} = 45.$$

Estas igualdades son lógicamente redundantes, pero adelantan la detección de inconsistencias locales (por ejemplo, sumas parciales que ya exceden 45 o que no pueden alcanzarlo con los dominios restantes), reduciendo nodos y fallos sin excluir ninguna solución válida.

Ruptura de simetría

Con *pistas fijas* las simetrías clásicas del Sudoku (permutaciones de filas/columnas/bandas, renombrado de dígitos, transposición) no preservan la instancia, pues desplazarían las pistas. Por ello no imponemos ruptura de simetría global: podría eliminar la única solución compatible con G. La instancia ya está "anclada" por las pistas y cualquier rompedor global sería potencialmente incorrecto.

Estrategias de búsqueda

Para las pruebas sobre el modelo de Reunion se plantean diferentes combinaciones de heurísticas, con el objetivo de analizar su impacto en el tamaño del árbol de búsqueda y el tiempo de resolución.

Solvers

Se utilizarán los *solvers* XXX, YYY y ZZZ para contrastar el comportamiento del modelo bajo motores de propagación diferentes. El objetivo es observar variaciones en tiempo y tamaño del árbol manteniendo la misma formulación.

1.3. Pruebas

Se evaluó el modelo sobre una batería de instancias `.dzn`. En cada corrida se registraron *tiempo*, *nodos*, *fallos*, *profundidad* y *número de soluciones*. A continuación se presenta una tabla plantilla para consolidar dichos resultados.

Tabla 1: Resultados de pruebas.

Archivo	Solver	Var heur	Val heur	time	nodes	fail	depth
example-e.dzn	Chuffed	first_fail	indomain_min	0.000	0	0	0
example-e.dzn	Gecode	dom_w_deg	indomain_split	0.000	0	0	0

1.4. Arboles de búsqueda

En esta sección se presentan las visualizaciones del árbol de búsqueda generadas por el modelo de Sudoku bajo distintas combinaciones de *solver* y *heurísticas*. Cada imagen muestra la estructura explorada para una instancia específica.

1.5. Análisis

Con base en la Tabla 1.3 y las figuras del árbol de búsqueda, comparamos las configuraciones por tres criterios: **nodes**, **failures** y **tiempo**.

- **Tamaño del árbol.** A
- **Fallos.** A
- **Profundidad.** A
- **Tiempo.** A
- **Solver.** A

Conclusión breve. Para *[instancias difíciles]*, recomendamos *dom_w_deg + indomain_split*; para *[fáciles/medias]*, *first_fail + indomain_min* es suficiente. Active siempre las redundantes.

1.6. Conclusiones

- A
- B
- C
- D

2. Kakuro

Introducción al problema y alcance del modelado. Supuestos y parámetros clave.a

2.1. Modelo

Definición de variables, dominios y restricciones principales. Justificación del modelo.

2.2. Detalles de implementación

Restricciones redundantes, rompimiento de simetrías y decisiones técnicas.

2.3. Pruebas

Casos de prueba, entradas, métricas y tablas o figuras de apoyo.

2.4. Árboles de búsqueda

Nodos explorados, fallos, tiempos y efecto de estrategias de distribución.

2.5. Análisis

Comparación de variantes y discusión de resultados.

2.6. Conclusiones

Lecciones, limitaciones y trabajo futuro.

3. Secuencia Mágica

Introducción al problema y alcance del modelado. Supuestos y parámetros clave.

3.1. Modelo

Definición de variables, dominios y restricciones principales. Justificación del modelo.

3.2. Detalles de implementación

Restricciones redundantes, rompimiento de simetrías y decisiones técnicas.

3.3. Pruebas

Casos de prueba, entradas, métricas y tablas o figuras de apoyo.

3.4. Árboles de búsqueda

Nodos explorados, fallos, tiempos y efecto de estrategias de distribución.

3.5. Análisis

Comparación de variantes y discusión de resultados.

3.6. Conclusiones

Lecciones, limitaciones y trabajo futuro.

4. Acertijo Lógico

Introducción al problema y alcance del modelado. Supuestos y parámetros clave.

4.1. Modelo

Definición de variables, dominios y restricciones principales. Justificación del modelo.

4.2. Detalles de implementación

Restricciones redundantes, rompimiento de simetrías y decisiones técnicas.

4.3. Pruebas

Casos de prueba, entradas, métricas y tablas o figuras de apoyo.

4.4. Árboles de búsqueda

Nodos explorados, fallos, tiempos y efecto de estrategias de distribución.

4.5. Análisis

Comparación de variantes y discusión de resultados.

4.6. Conclusiones

Lecciones, limitaciones y trabajo futuro.

5. Ubicación de personas en una reunión

Un grupo de N personas desea tomarse una fotografía en una sola fila. Algunas parejas de personas imponen preferencias de proximidad: *adyacencia* (`next`), *no adyacencia* (`separate`) y *cota máxima de distancia* (`distance`), que limitan cuántas personas pueden quedar entre dos individuos.

Este problema puede modelarse como un *Problema de Satisfacción de Restricciones* (CSP): cada persona debe ocupar exactamente una posición en la fila y las preferencias se expresan como restricciones sobre las posiciones relativas (por ejemplo, $|\text{pos}(A) - \text{pos}(B)| = 1$ para `next`, $|\text{pos}(A) - \text{pos}(B)| \geq 2$ para `separate`, y $|\text{pos}(A) - \text{pos}(B)| \leq M + 1$ para `distance`). El objetivo es encontrar cualquier orden que satisfaga simultáneamente todas las preferencias, o certificar que no existe.

5.1. Modelo

Parámetros

P1 — N**:** Número de personas a ubicar.

$$N \in \mathbb{Z}_{\geq 1}.$$

P2 — S**:** Índices válidos para personas.

$$S = \{1, \ldots, N\}.$$

P3 — POS**:** Conjunto de posiciones disponibles en la fila.

$$POS = \{1, \ldots, N\}.$$

P4 — `personas`**:** Vector de nombres.

$$\texttt{personas} \in \text{String}^S.$$

P5 — $K_{\textsf{next}}, K_{\textsf{sep}}, K_{\textsf{dist}}$**:** Cantidad de preferencias de cada tipo.

$$K_{\textsf{next}},\ K_{\textsf{sep}},\ K_{\textsf{dist}} \in \mathbb{Z}_{\geq 0}.$$

P6 — `NEXT`, `SEP`, `DIST`**:** Matrices de preferencias:

$$\texttt{NEXT} \in S^{K_{\textsf{next}} \times 2}, \quad \texttt{SEP} \in S^{K_{\textsf{sep}} \times 2}, \quad \texttt{DIST} \in \left(S \times S \times \{0, \ldots, N-2\}\right)^{K_{\textsf{dist}}}.$$

Cada fila codifica un par (o trío) de personas y, en `DIST`, una cota M de separación.

Variables

V1 — POS_OF_p**:** Posición que ocupa la persona p.

$$POS\_OF_p \in POS, \qquad p \in S.$$

V2 — PER_AT_i**:** Persona ubicada en la posición i.

$$PER\_AT_i \in S, \qquad i \in POS.$$

Restricciones principales

R1 — **Biección (canalización):** La asignación es una permutación válida: cada persona ocupa exactamente una posición y cada posición contiene exactamente una persona.

$$\mathit{inverse}(POS\_OF,\ PER\_AT).$$

R2 — **Preferencias next**(A, B)**:** A y B deben quedar adyacentes.

$$\forall (A, B) \in \texttt{NEXT}: \quad \left| POS\_OF_A - POS\_OF_B \right| = 1.$$

R3 — **Preferencias separate**(A, B)**:** A y B no pueden quedar adyacentes.

$$\forall (A, B) \in \texttt{SEP}: \quad \left| POS\_OF_A - POS\_OF_B \right| \geq 2.$$

R4 — **Preferencias distance**(A, B, M)**:** A lo sumo M personas entre A y B, equivalente a cota sobre distancia de posiciones.

$$\forall (A, B, M) \in \texttt{DIST}: \quad \left| POS\_OF_A - POS\_OF_B \right| \ \leq \ M + 1.$$

Restricciones redundantes

R5 — **Refuerzo de permutación:** Duplican la biección y aceleran la propagación.

$$\mathit{all\_different}([POS\_OF_p \mid p \in S]), \qquad \mathit{all\_different}([PER\_AT_i \mid i \in POS]),$$

$$\sum_{p \in S} POS\_OF_p \ = \ \frac{N(N+1)}{2}.$$

R6 — **Validación de datos:** Índices válidos y pares distintos; en `DIST`, cota M dentro de $[0,\ N-2]$.

$$\forall (A, B) \in \texttt{NEXT}: \ A, B \in S, \ A \neq B,$$
$$\forall (A, B) \in \texttt{SEP}: \ A, B \in S, \ A \neq B,$$
$$\forall (A, B, M) \in \texttt{DIST}: \ A, B \in S, \ A \neq B, \ 0 \leq M \leq N - 2.$$

R7 — No contradicción next vs sep: Se prohíbe declarar simultáneamente que dos personas deban y no deban estar juntas.

$$\forall (A, B) \in \texttt{NEXT},\ \forall (C, D) \in \texttt{SEP}:\ (A, B) \notin \{(C, D), (D, C)\}.$$

R8 — Implicación local: $\mathsf{distance}(A, B, 0)$ es equivalente a $\mathsf{next}(A, B)$.

$$\forall (A, B, 0) \in \texttt{DIST}: \quad |\, POS\_OF_A - POS\_OF_B \,| = 1.$$

Restricciones de simetrías

R9 — Rompimiento de simetría izquierda–derecha: La solución reflejada (espejada) es equivalente; para evitar exploración duplicada, se fija un orden canónico en los extremos.

$$PER\_AT_1 \ < \ PER\_AT_N.$$

Justificación: si $[p_1, \ldots, p_N]$ es solución, entonces $[p_N, \ldots, p_1]$ también suele serlo cuando las preferencias dependen sólo de distancias absolutas. Forzar $PER\_AT_1 < PER\_AT_N$ descarta exactamente una de las dos, reduciendo el espacio de búsqueda sin eliminar soluciones no simétricas.

5.2. Detalles de implementación

Archivos y organización

- `reunion.mzn`: modelo completo.
- `tests/*.dzn`: instancias con:
  - `N`, `personas` (vector de nombres),
  - `K_NEXT`, `K_SEP`, `K_DIST`,
  - `NEXT` $\in \mathbb{Z}^{K_{\mathsf{next}} \times 2}$, `SEP` $\in \mathbb{Z}^{K_{\mathsf{sep}} \times 2}$,
  - `DIST` $\in \mathbb{Z}^{K_{\mathsf{dist}} \times 3}$ con triples (A, B, M).

Modelo

Modelo

Usamos dos vistas de la permutación, POS_OF (persona→posición) y PER_AT (posición→persona), enlazadas con *inverse*, porque refuerzan la propagación sin cambiar la semántica. Con una sola vista (p. ej., POS_OF) ya podríamos expresar $|POS\_OF_A - POS\_OF_B|$, pero la canalización bidireccional permite a los propagadores recortar dominios antes y con menos fallos que usar solo *all_different*. Además, la salida es más directa y limpia: imprimir la fila en orden $1..N$ se reduce a recorrer $PER\_AT[i]$ sin accesos indirectos ni restricciones *element*. También simplifica la ruptura de simetría $PER\_AT[1] < PER\_AT[N]$. En suma, la doble vista mejora eficiencia, claridad de salida y facilidad de formulación, manteniendo dominios exactos $POS = S = \{1, \ldots, N\}$ sin sentinelas.

Restricciones redundantes

Se activan de manera permanente porque fortalecen la propagación sin alterar el conjunto de soluciones. En primer lugar, se duplican las vistas de permutación mediante *all_different* tanto sobre POS_OF como sobre PER_AT; aunque la biección ya está garantizada por *inverse*(POS_OF, PER_AT), imponer *all_different*$([POS\_OF_p \mid p \in S])$ y *all_different*$([PER\_AT_i \mid i \in POS])$ añade redundancia estructural que los propagadores explotan para reducir dominios antes y con menos fallos. En segundo lugar, la igualdad

$$\sum_{p \in S} POS\_OF_p \;=\; \frac{N(N+1)}{2}$$

no introduce información nueva sobre las soluciones, pero actúa como restricción global lineal útil cuando quedan pocas posiciones libres, cerrando brechas y detectando inconsistencias parciales con bajo costo. En tercer lugar, se materializa la implicación local $\mathsf{distance}(A, B, 0) \Rightarrow |POS\_OF_A - POS\_OF_B| = 1$; esta equivalencia ya está contenida semánticamente en distance, pero declararla explícitamente evita que el solver deba inferirla por combinación de propagadores, acelerando los casos triviales de "cero personas entre A y B".

Además, integramos como redundantes los chequeos de consistencia de datos, pues acotan tempranamente la búsqueda sin modificar el conjunto de soluciones factibles cuando la instancia es válida. Exigir $A, B \in S$ y $A \neq B$ en `NEXT` y `SEP`, así como $A, B \in S$, $A \neq B$ y $0 \leq M \leq N-2$ en `DIST`, no cambia la semántica del problema, pero evita que dominios inválidos o cotas imposibles entren al árbol de búsqueda. De igual modo, prohibir la contradicción obvia entre next y separate para el mismo par (A, B) (en cualquier orden) no elimina soluciones legítimas: únicamente descarta instancias mal especificadas o ramas inconsistentes que, de otro modo, el solver podaría más tarde a mayor costo.

Ruptura de simetría

El modelo presenta simetría de reflexión izquierda–derecha: si $[p_1, \ldots, p_N]$ es solución, $[p_N, \ldots, p_1]$ también lo es, pues todas las restricciones dependen de distancias absolutas entre posiciones ($|POS\_OF_A - POS\_OF_B|$) y de la biyección *inverse*, que son invariantes ante la transformación $i \mapsto N{+}1{-}i$. Por ello conviene imponer $PER\_AT[1] < PER\_AT[N]$, que fija un representante canónico por cada par de soluciones espejo sin eliminar soluciones no simétricas: dado que $PER\_AT[1] \neq PER\_AT[N]$, exactamente una de las dos disposiciones reflejadas satisface $<$, mientras la otra la viola. Esta ruptura reduce a la mitad, en términos ideales, el espacio de búsqueda explorado por el solver sin afectar satisfacibilidad ni, si se añadiera un objetivo, la óptimalidad. La condición es segura mientras no existan reglas que distingan explícitamente los extremos, caso en el cual la simetría ya no está presente.

Estrategias de búsqueda

Para las pruebas sobre el modelo de Reunion se plantean diferentes combinaciones de heurísticas, con el objetivo de analizar su impacto en el tamaño del árbol de búsqueda y el tiempo de resolución.

Solvers

Se utilizarán los *solvers* XXX, YYY y ZZZ para contrastar el comportamiento del modelo bajo motores de propagación diferentes. El objetivo es observar variaciones en tiempo y tamaño del árbol manteniendo la misma formulación.

5.3. Pruebas

Se evaluó el modelo sobre una batería de instancias `.dzn`. En cada corrida se registraron *tiempo*, *nodos*, *fallos*, *profundidad* y *número de soluciones*. A continuación se presenta una tabla plantilla para consolidar dichos resultados.

Tabla 2: Resultados de pruebas.

Archivo	**Solver**	**Var heur**	**Val heur**	**time**	**nodes**	**fail**	**depth**
example-e.dzn	Chuffed	first_fail	indomain_min	0.000	0	0	0
example-e.dzn	Gecode	dom_w_deg	indomain_split	0.000	0	0	0

5.4. Arboles de búsqueda

En esta sección se presentan las visualizaciones del árbol de búsqueda generadas por el modelo de Reunion bajo distintas combinaciones de *solver* y *heurísticas*. Cada imagen muestra la estructura explorada para una instancia específica.

5.5. Análisis

Con base en la Tabla 5.3 y las figuras del árbol de búsqueda, comparamos las configuraciones por tres criterios: **nodes**, **failures** y **tiempo**.

- **Tamaño del árbol.** A
- **Fallos.** A
- **Profundidad.** A
- **Tiempo.** A
- **Solver.** A

Conclusión breve. Para *[instancias difíciles]*, recomendamos *dom_w_deg + indomain_split*; para *[fáciles/medias]*, *first_fail + indomain_min* es suficiente. Active siempre las redundantes.

5.6. Conclusiones

- A
- B
- C
- D

6. Construcción de un rectángulo

Introducción al problema y alcance del modelado. Supuestos y parámetros clave.

6.1. Modelo

Definición de variables, dominios y restricciones principales. Justificación del modelo.

6.2. Detalles de implementación

Restricciones redundantes, rompimiento de simetrías y decisiones técnicas.

6.3. Pruebas

Casos de prueba, entradas, métricas y tablas o figuras de apoyo.

6.4. Árboles de búsqueda

Nodos explorados, fallos, tiempos y efecto de estrategias de distribución.

6.5. Análisis

Comparación de variantes y discusión de resultados.

6.6. Conclusiones

Lecciones, limitaciones y trabajo futuro.