Trabajo Práctico Especial 1 - Métodos de Búsqueda No Informados e Informados

Sistemas de Inteligencia Artificial

Grupo 12

Pedro Balaguer - 55795
Agustín Izaguirre - 57774
Juan Li Puma - 55824

Índice

Trabajo Realizado

El problema modelado es el de la búsqueda de soluciones del juego Oh-n0, que consta de un tablero de NxN círculos blancos, azules o rojos con una configuración inicial. Los círculos azules iniciales contienen un número que indica exactamente cuántos círculos azules deberían poder ver en líneas horizontales o verticales. Los círculos rojos "cortan" la visión de los círculos azules. El jugador tiene permitido cambiar el color de los círculos blancos a azul (estos azules no tienen número) o a rojo hasta completar el tablero. El tablero se considera solución si se cumplen las siguientes condiciones:

- Se satisfacen todas las restricciones de azules
- No hay círculos blancos
- No hay ningún círculo azul cuyos vecinos son todos rojos ("islas")

Utilizando conceptos vistos en clase, se implementaron varias estrategias de búsqueda para para encontrar tableros solución (o indicar que uno no existe) a partir de tableros iniciales con un resolvedor general de problemas (GPS).

Se tuvieron en cuenta dos enfoques para el desarrollo del trabajo práctico:

- Formulación incremental (Ir llenando el tablero).
- Formulación completa (Tablero lleno e ir Arreglandolo)

Implementaciones de Métodos de Búsqueda

BFS/DFS

Estas dos estrategias de búsqueda se implementaron como se suelen implementar clásicamente: Con una cola para BFS y una pila para DFS.

IDDFS

Varias particularidades surgen de la implementación de esta estrategia de búsqueda. En primer lugar, para agilizar la búsqueda el aumento de profundidad del algoritmo es exponencial. En su primera iteración la profundidad está limitada a 16, y si no se encuentra una solución (la pila de nodos a revisar queda vacía) la profundidad se duplica en cada intento subsiguiente.
Si el problema no tiene solución, eventualmente un aumento en el límite de profundidad no producirá una búsqueda efectivamente más profunda, ya que todo el grafo de movimientos ha sido explorado. Por esto para cada intento de IDDFS se guarda la máxima profundidad alcanzada en el intento anterior. Cuando dos corridas con límites de profundidad distintos lleguen a la misma profundidad real máxima, se concluye que el problema no tiene solución. IDDFS deja de iterar y reporta que no se encontró una solución.
En caso de haber una solución y encontrarse ésta, IDDFS no la retorna inmediatamente ya que la solución puede ser subóptima: Puede existir una solución de menor costo que se encontraría con un límite de profundidad entre el recién intentado y el anterior. Para encontrarla (o cerciorarse que no existe) realiza búsqueda binaria, tomando como piso

$profundidad\ anterior + 1$ y como techo $profundidad\ solución - 1$. Si no se encuentra una solución mejor, efectivamente la original es la óptima y se devuelve ésta.

Greedy/A*

La iteración y explotación de Greedy y A* son idénticas: ambas al explotar ordenan los sub-estados según un criterio y las agregan a su PriorityQueue de nodos a explotar. La diferencia es en su criterio: Greedy sólo considera el costo de cada nodo, mientras que A* toma la suma del valor de la heurística para ese nodo y su costo; si hay un empate vuelve a comparar considerando únicamente el valor de la heurística. Este criterio de orden se implementa con una cadena de Comparators, distinta para cada estrategia. Cabe destacar que la PriorityQueue de Java resuelve empates de manera arbitraria pero determinista; es decir, dados los mismos A y B considerados iguales, siempre terminarán en el mismo orden relativo en la cola. Para quitar este determinismo e introducir aleatoriedad a la exploración, cada cadena de Comparators tiene agregado como último eslabón de la cadena un Comparator que retorna un número aleatorio en el intervalo [-0.5, 0.5), independiente de A y B (el número siempre es distinto).

Generador

Para generar tableros, utilizamos el mismo generador que el juego original en www.0hn0.com. El código se encuentra público en www.github.com/Techdojo/0hn0, por lo que extrayendo las funciones fundamentales para la generación del tablero y agregando algunas propias se creó un generador de tableros que imprime en archivos en un formato de tablero que luego puede ser interpretado como entrada a nuestro programa. La ventaja de utilizar el el mismo código que el juego original es que todos los tableros generados tienen solución.

Output

Cuando finaliza, el programa imprime sus salida de dos formas:

1. Por salida estándar se obtienen las siguientes estadísticas: tamaño de tablero, estrategia de búsqueda, método de resolución (formulación incremental o reparación heurística), método de llenado inicial de tablero (para reparación heurística), profundidad de la solución, costo de la solución, cantidad de nodos expandidos, cantidad de estados analizados, cantidad de nodos frontera y tiempo total de ejecución.
2. Se imprime en un archivo (*p5/output.out*) todos los pasos desde el estado inicial hasta el estado meta. Para una mejor visualización de la solución, se puede abrir el archivo *p5/index.html* en el navegador. Este último incluye *p5.js*, librería JavaScript de visualización y gráficos, y un código JavaScript que lee *solution.out* y dibuja paso a paso la evolución de estados. (NOTA: probar abrir *index.html* con distintos navegadores ya que pueden existir problemas de CORS, Firefox suele funcionar correctamente).

Reglas

Cada enfoque de resolución tiene sus propias reglas:

- Reglas enfoque incremental: Cada blanco puede convertirse en azul o rojo. Las reglas son aplicables si el tablero generado es un tablero válido.
- Reglas enfoque completo: Para este enfoque las reglas dependen de la manera en la que se relleno el tablero(todos los blancos azules, todos los blancos rojos o cada blanco era random su color azul o rojo). Para el llenado azul las reglas eran para cada punto no fijo este podía cambiar su color a rojo, para llenado rojo cada punto no fijo podía cambiar su color a azul y para llenado random cada punto no fijo podía cambiar su color(este último tiene el doble de reglas que los dos anteriores). En reparación heurística las reglas son aplicables aunque se generen tableros inválidos ya que se mueve por tableros inválidos hasta llegar a la solución.

Heurísticas

Así como las reglas, se desarrollaron diferentes heurísticas para cada enfoque y algunas fueron compartidas entre ambos enfoques.
Se detallaran con el número asociado a la heurística en el archivo de configuración(settings.properties) y el nombre que tiene la heurística en el código. Se verán las faltantes en el anexo.

Heurísticas de Formulación Incremental y Completa

Heurística 1 (Conflicting Numbers Heuristic)

Decimos que un número azul está en conflicto si los puntos azules que puede ver no coinciden con los que dice su número, ya sea que le falten o que se pase.
El valor de esta heurística es el total de números azules en conflicto.
No es una heurística admisible, ya que con un solo cambio se puede solucionar más de un conflicto.(Ejemplo y Contraejemplo en el anexo)

Heurísticas de Formulación Incremental(Enfoque Fill Blanks)

Heurística 0 (Filling Blanks Heuristic)

Es una heurística trivial, que dado un tablero inicial, el valor de su heurística es la cantidad de blancos que le falta para completar el tablero.
Es una heurística admisible si se parte de tableros válidos, ya que las reglas son aplicables únicamente cuando generan tableros válidos y en cada paso se cambia de color exactamente un único blanco.(Demostración y Ejemplo en el anexo)

Heurística 2 (Fill Blanks Non-Trivial Admissible Heuristic)

Consiste en 3 pasos:

1. Se cuenta la cantidad de números azules que no están cumpliendo su restricción, ya sea porque ven azules de más o de menos (a éstos los llamaremos números en conflicto).

2. Para cada blanco se cuenta por cuantos números en conflicto es visible (únicamente por caminos azules) y se queda con el máximo de éstos (lo llamaremos potencial de resolución de conflictos).

3. Se cuenta cuántos blancos no participan (no alcanzables por azules con número) y cuántos blancos hay cuyos vecinos azules con número no están en conflicto (blancos restantes).

A partir de esta información se calcula el valor de la heurística con la siguiente ecuación:
Si números en conflicto > 0

$$H2(X) = Max(\text{números en Conflicto} - \text{potencial de resolución}, 1) + \text{blancos que no participan} + \text{blancos restantes}$$

Sino

$$H2(x) = \text{blancos que no participan} + \text{blancos restantes}$$

Es admisible(Demostración y Ejemplo en anexo).

Heurísticas de Formulación Completa (Enfoque Heuristic Repair)

Heurística 3 (Heuristic Reparation Admisible Heuristic)

Cuenta el total números en conflicto que hay. Luego para cada punto no fijo cuenta cuántos números en conflicto alcanza solo moviéndose por esa columna o esa fila siguiendo únicamente puntos azules,y de todos los valores se queda con el máximo (a este máximo lo llamamos potencial de resolución de conflictos). Por último cuenta cuántas islas azules hay (puntos azules que sólo tienen como adyacentes puntos rojos).
Resta el total de números en conflicto con el máximo de adyacentes en conflicto, si la resta da menor o igual a cero y hay 1 o más números en conflicto, se queda con el valor 1.
Si no habia numeros en conflictos la resta dio 0 y le suma la cantidad de islas azules y devuelve ese valor, sino devuelve el valor de la resta antes mencionada.
Es una heurística admisible.(Demostración y Ejemplo en el anexo)..

Resultados

Filling Method

Podemos observar (gráficos 1 y 2) que tanto en tableros 5x5 cómo 6x6, el método más eficiente es llenar el tablero de azules. Por otro lado observamos que llenar inicialmente con rojos permite encontrar una solución más rápidamente

que el llenado aleatorio en tableros 5x5, pero en 6x6 sucede lo contrario. Analizando además los gráficos de profundidad y de cantidad de nodos analizados (gráficos 3, 4, 5 y 6), observamos que cuando la profundidad de la solución es igual (en el caso de tableros 5x5), filling reds expande menos nodos por tener la mitad de reglas. Sin embargo, en tableros 6x6, la profundidad de la solución con filling reds es mayor que con filling randomly (porque los tableros solución suelen tener mayor cantidad de azules que rojos, por lo que un tablero llenado inicialmente con rojos suele estar más lejos de la solución que con un llenado aleatorio que contiene un ~50% de azules colocados), y tener menos reglas no llega a contrarrestar la gran cantidad de nodos a analizar (recordemos que en reparación heurística no se pueden podar los caminos con tableros inválidos como con formulación incremental).

Heurísticas

a. Formulación Incremental (gráfico 7)

Para formulación incremental la heurística

b. Reparación Heurística (gráfico 8)

En reparación heurística, *ConflictingNumbers* es la heurística por lejos más eficiente a pesar de no ser admisible.
Incluimos en este análisis AddAll ya que a pesar de sumar heurísticas de formulación incremental, el resultado de ellas será 0, por lo que AddAll es el equivalente a sumar *MissingReds* y *MissingVisibleBlues*.

Estrategias de búsqueda

Tanto en reparación heurística como en formulación incremental A* es la estrategia de búsqueda más veloz (gráficos 9 y 10). También podemos observar que, con excepción de A*, la solución se encuentra más rápidamente con formulación incremental que con reparación heurística. Esto se debe a que, como observamos en los gráficos 11 y 12 del anexo, la cantidad de nodos analizados en reparación heurística es diez veces mayor que en formulación incremental. Esta diferencia puede explicarse debido a que en formulación incremental se podan caminos con un camino inválido mientras que en reparación heurística no se puede hacer, por lo que se van a explorar muchos más nodos.

Conclusiones

- Cuando se trata de resolver mediante reparación heurística, la mejor forma es llenar el tablero de azules debido a que tiene la mitad de reglas que llenarlo de forma aleatoria y se encuentra más cerca de la solución que cuando se llena de rojos (los estados finales suelen tener mayor cantidad de azules que rojos).

- Una heurística admisible no implica necesariamente ser más eficiente.

- Con excepción de A*, la búsqueda resulta más veloz con formulación incremental.

Anexo

Librerías Externas Utilizadas

- org.apache.commons.commons-lang3 - En particular el HashCodeBuilder, que ayuda a armar una buena función de hash para cualquier objeto, especificando cuáles campos a tener en cuenta para el cálculo del hash. Se utilizó para hacer una buena función de hash para los tableros (de manera que tableros distintos efectivamente tengan hash codes distintos), sin que ésta sea prohibitivamente costosa de calcular. Esta librería se utilizó con previa aprobación de la cátedra.

Demostraciones

Admisibilidad de heurística 0 (Filling Blanks Heuristic)

Veamos primero que la heurística es 0 si solo si es un estado meta.
Demostracion:

Heurística es 0 ⟶ es un estado meta
Si la heurística es 0, entonces significa que el tablero no tiene blancos y por lo tanto el tablero está lleno, como las reglas al aplicarse solo generan tableros válidos, este es un tablero valido y esta lleno. Por lo tanto es un estado meta.

Es un estado meta ⟶ heurística es 0
Si es un estado meta, entonces no posee blancos. Por lo tanto la heurística es 0.

Veamos ahora que el costo real a partir de cualquier estado siempre es mayor o igual a la heurística.
Demostracion:
Supongo que existe algún caso en el que el costo real a partir de un estado para llegar a la
Solución es menor a la heurística de ese estado. Entonces para llegar a la solución a partir de ese estado hago menos movimientos que la cantidad de blancos que tengo en el estado(ya que el costo es constante y de valor 1), pero en cada movimiento cambio un único blanco por un punto de color rojo o azul. Por lo tanto en la solución hay blancos.

Absurdo contradice la condición de solución. Por lo tanto la heurística nunca sobreestima el costo real.

Por lo tanto la heurística es admisible.

Admisibilidad de heurística 2 (Fill Blanks Non-Trivial Admissible Heuristic)

Veamos primero que la heurística es 0 si solo si es un estado meta.
Demostracion:

Heurística es 0 ⟶ es un estado meta

Si la heurística es 0, entonces significa que el tablero no tiene números en conflictos y que no tiene blancos y por lo tanto el tablero está lleno. Por lo tanto es un estado meta.

Analicemos en detalle porque decimos que no tiene blancos, si tuviera blancos estos caerían en una de las siguientes dos categorías:

1. Alcanza por un camino azul a algun numero que esta conflicto.
2. No alcanza por ningún camino a azul a un número en conflicto.

En la primer categoría no pueden caer por que ya dijimos que no tiene conflictos, sino la heurística sería mayor a cero

Si caen en la segunda categoría podemos volver a subdividir en dos categorías:

1. Alguno de sus adyacentes es un número sin conflicto.
2. Ninguno de sus adyacentes es un número, es decir sus adyacentes son puntos blancos rojos o azules sin número.

Si caen en el primer caso estarían contados dentro de los blancos restantes y no puede suceder ya que si hay algún blanco restante la heurística sería mayor a cero.

Si caen en el segundo caso estarían dentro de los blancos que no participan y no puede suceder ya que si hay algún blanco que no participa la heurística sería mayor a cero.

Por lo tanto podemos concluir que no hay blancos y de esta manera como dijimos antes es un estado meta.

Es un estado meta ⟶ heurística es 0

Si es un estado meta, entonces no posee conflictos ni blancos. Como no posee conflictos el número de números en conflicto es 0 y como no posee blancos el número de blancos restantes y el número de blancos que no participan es 0.

Si reemplazamos esto en la ecuación de evaluación de la heurística como números en conflicto es 0 nos da:

$H2(x) = 0 + 0 = 0$

Veamos ahora que el costo real a partir de cualquier estado siempre es mayor o igual a la heurística.

Demostracion:

Supongo que no, entonces significa que algún estado logró llegar a la solución con menos costo que el valor de la heurística. Entonces logró cambiar más de un blanco en un paso o logro arreglar más conflictos que el máximo potencial de resolución.

El primer caso es absurdo puesto que solo se puede cambiar un blanco por movimiento.

El segundo caso es absurdo por definición del potencial de resolución de conflictos.

Por lo tanto la heurística nunca sobreestima el costo real.

Por lo tanto la heurística es admisible.

Admisibilidad de heurística 3 (Heuristic Reparation Admisible Heuristic)

Veamos primero que la heurística es 0 si solo si es un estado meta.
Demostracion:

Heurística es 0 ⟞ es un estado meta
Heuristica da 0 → no hay números en conflicto y no hay islas azules →
→ el tablero está lleno y no tiene conflictos → es un estado meta.

Es un estado meta ⟞ heurística es 0
Es un estado meta → no hay números en conflicto y no hay islas azules →
→ heuristica da 0.

Veamos ahora que el costo real a partir de cualquier estado siempre es mayor o igual a la heurística.
Demostracion:
Supongo que hay un estado que llega a la solución con menos costo que la heurística, entonces significa que en algún cambio logra solucionar más de una isla azul o logra eliminar más números en conflicto que el potencial de resolución de conflictos.
No puede solucionar más de una isla azul en un cambio ya que el cambio modifica un único punto y un mismo punto no puede formar dos islas azules.
Tampoco puede eliminar más números en conflictos que el potencial de resolución de conflictos por definición del potencial de resolución de conflictos.

Por lo tanto la heurística nunca sobreestima el costo real.
Por lo tanto la heurística es admisible.

Heuristicas Faltantes

Heurísticas de Formulación Incremental y Completa

Heurística 5 (Missing Reds Heuristic)

Calcula la suma de todos los números azules, cuenta los puntos fijos (los puntos de color rojo o azul del tablero inicial, estos no pueden modificarse) y cuenta los rojos.

$$H5(x) = Max(dim \times dim - suma\ numeros\ azules - puntos\ fijos - rojos\ no\ fijos,\ 0)$$

Donde dim es la dimensión del tablero.
No es admisible(Ejemplo y Contraejemplo en el anexo).

Heurística 6 (Missing Visible Blues Heuristic)

Para cada número azul cuenta cuántas azules le faltan o sobran, y su valor es la suma.

No es admisible.(Ejemplo y Contraejemplo en el anexo).

Heurísticas de Formulación Incremental

Heurística 4 (Approximate Reds Heuristic)

Para cada número azul suma cuántos azules le faltan ver (a la suma total se la llamará azules faltantes) y cuenta también cuántos blancos hay.

$H4(x) = Max(blancos - azules\,faltantes,\ 0)$

No es admisible, (Ejemplo y Contraejemplo en el anexo).

Heurística 7 (Add All Heuristic)

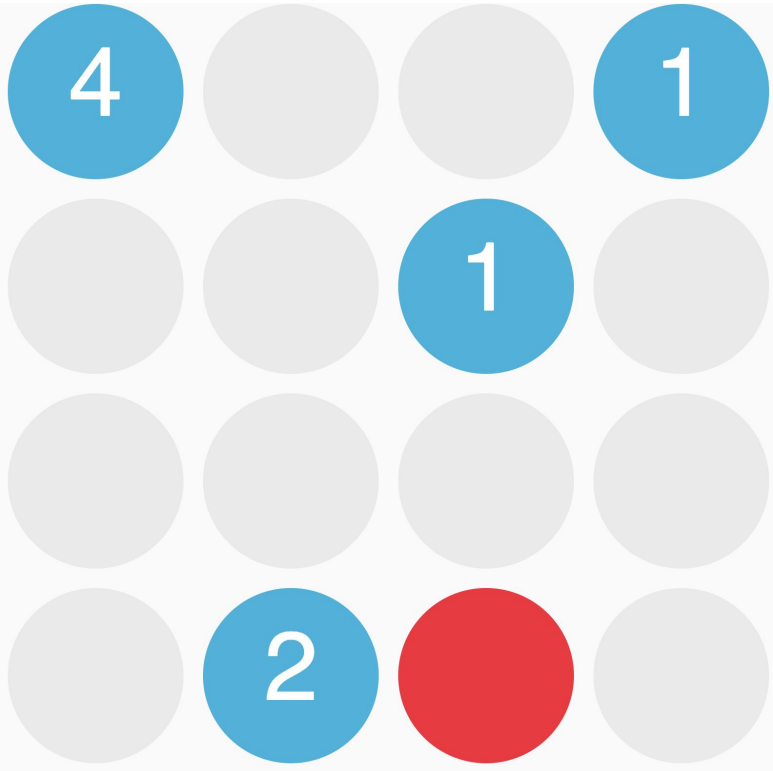
Esta heurística fue diseñada debido a que muchas de las heurísticas anteriores daban valores bajos muy rápido, para ver si mejoraba algo la búsqueda.

$H7(x) = H0(x) + H4(x) + H5(x) + H6(x)$

No es admisible.(Ejemplo y Contraejemplo en el anexo)

Ejemplos de Heurísticas

Ejemplo Heurística 0 (Filling Blanks Heuristic)

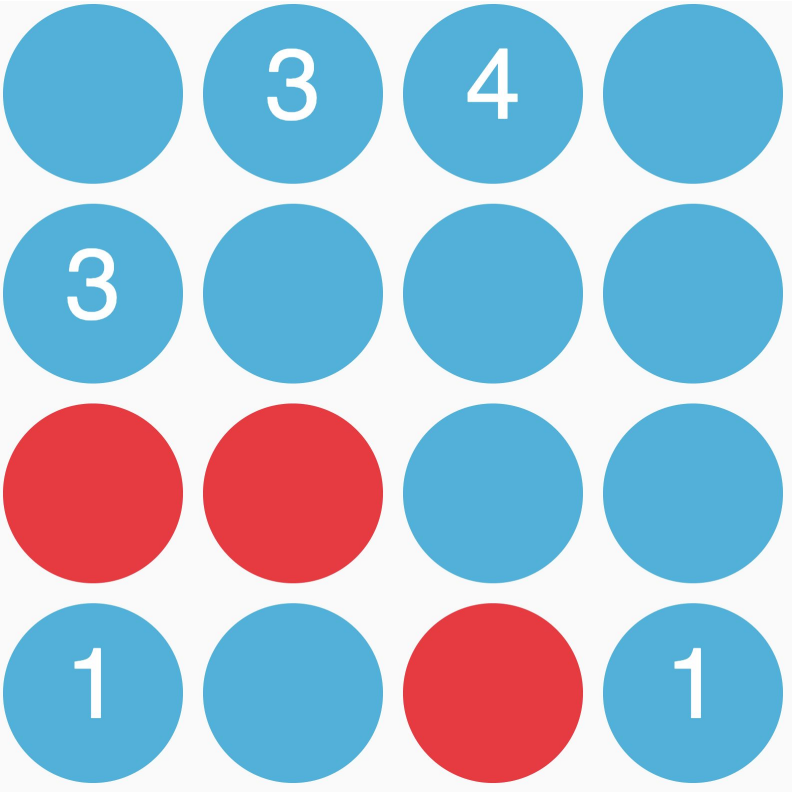


Analicemos el tablero

- Primer fila = 2 blancos
- Segunda fila = 3 blancos
- Tercer fila = 4 blancos
- Cuarta fila = 2

H1(x) = blancos = 11

Ejemplo Heurística 1 (Conflicting Numbers Heuristic)



Analizamos el tablero:

Primer fila:

- número 3 ve 4 azules así que está en conflicto.
- número 4 ve 5 azules así que está en conflicto.

Total fila 1 = 2

Segunda fila:

- Número 3 ve 4 azules así que esta en conflicto.

Total fila 2 = 1

Tercer fila

- No posee numeros

Total fila 3 = 0

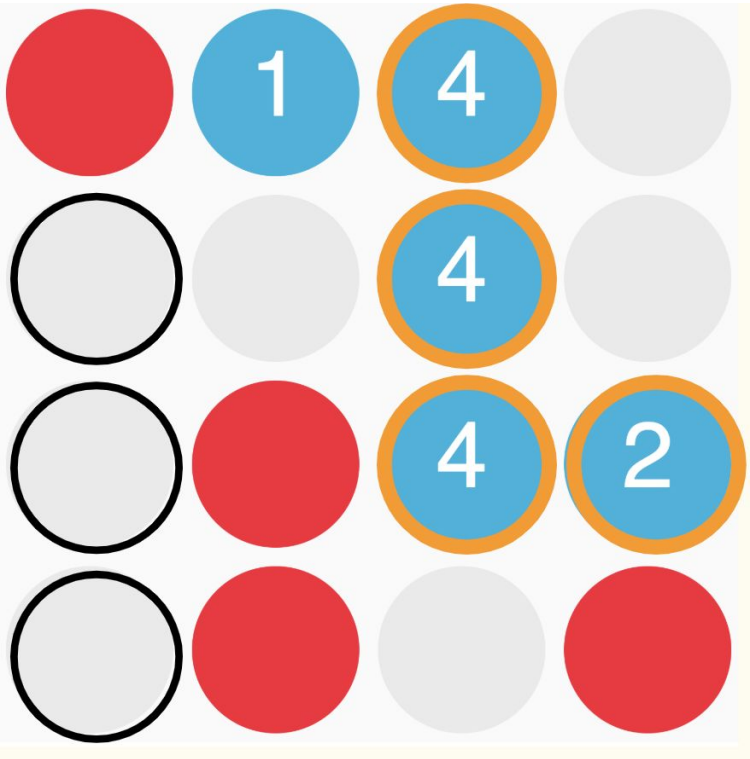
Cuarta fila:

- Primer 1 ve un azul así que no está en conflicto.
- Segundo 1 ve 3 azules así que está en conflicto.

Total fila 4 = 1

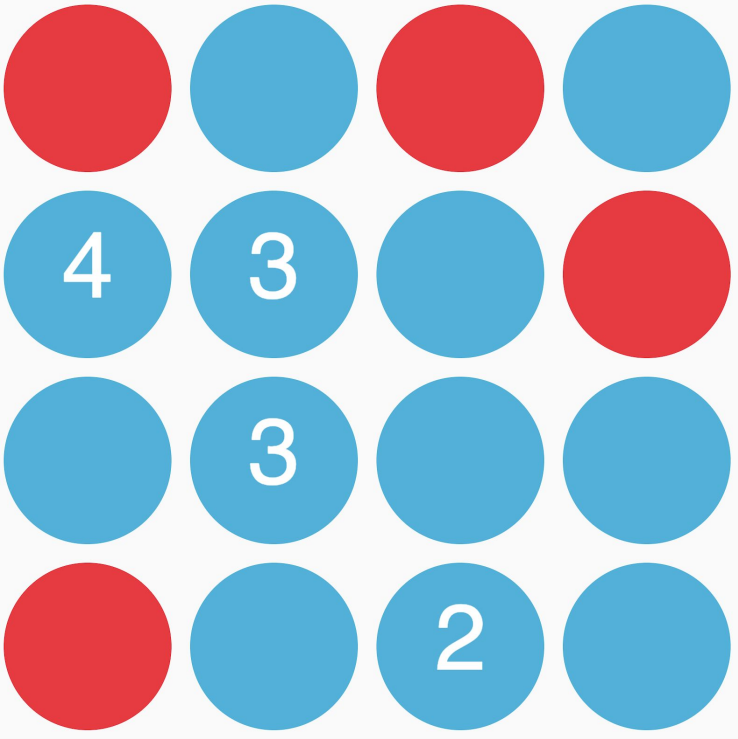
H1(x) = 2 + 1 + 0 + 1 = 4

Ejemplo Heurística 2 (Fill Blanks Non-Trivial Admissible Heuristic)



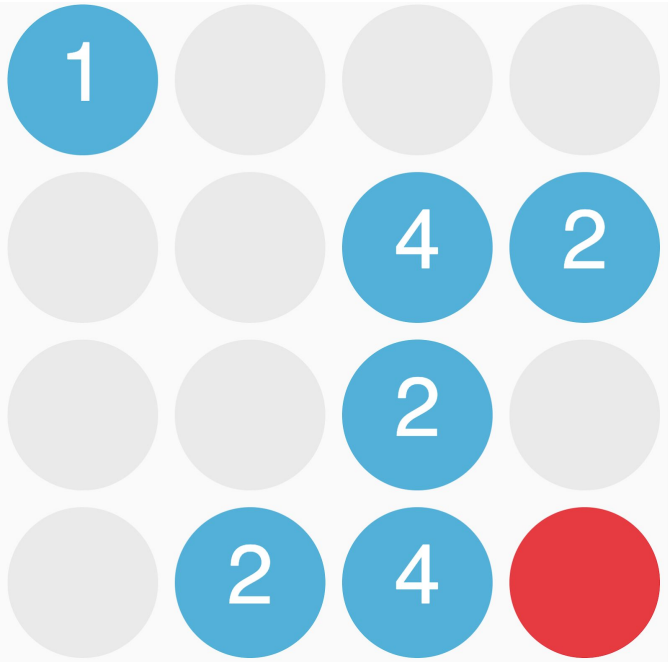
Analicemos el tablero:
Blancos no participan: 3 (en borde negro) ya que sus adyacentes son únicamente de color rojo o blanco
Números en Conflicto: 4 (borde naranja), ya que cada uno de ellos ve menos cantidad de azules de la que debería
Potencial de resolución: 2 ya que la máxima cantidad de números en conflictos alcanzables desde un blanco es dos, para el blanco de la primer fila o para el blanco de más a la derecha de la segunda fila
Blancos Restantes: 0, ya que todos los blancos no participan o alcanzan un número en conflicto.
H2(x) = 4 - 2 + 3 + 0 = 5

Ejemplo Heurística 3 (Heuristic Reparation Admisible Heuristic)



Analicemos el tablero:
Números en Conflicto = 1 + 1 + 1 + 1 = 4
Potencial de resolución = Max(1, 2, 1, 0, 3, 2, 2, 2, 1, 2, 3, 1) = 3
Islas azules = 1 (el azul de más a la derecha de la primer fila, ya que todos sus adyacentes son rojos)
Total = 4 - 3 = 1

Ejemplo Heurística 4 (Approximate Reds Heuristic)



Analicemos el tablero:

Blancos:

- Primer fila = 3
- Segunda fila = 2
- Tercer fila = 3
- Cuarta fila = 1
- Total = 9

Azules faltantes:

Primer Fila: Al 1 azul le falta 1

Fila 1 = 1

Segunda Fila: Al 4 azul le falta 1 y al 2 azul el falta 1

- Fila 2 = 1 + 1 = 2

Tercer Fila: Al 2 azul no le falta ninguna

- Fila tres = 0

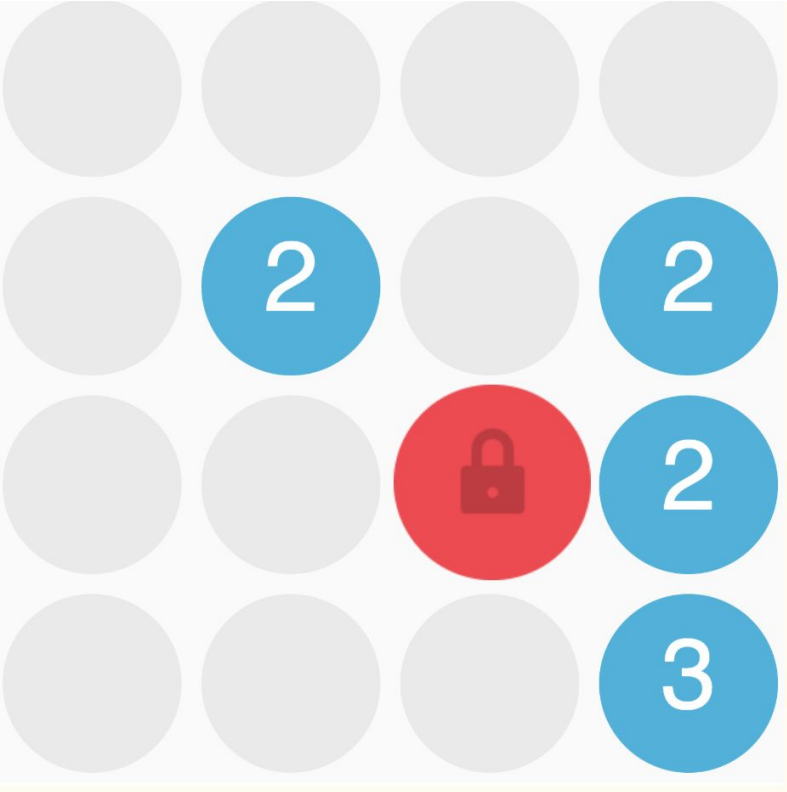
Cuarta fila: Al 2 azul le falta 1 y al 4 azul le falta 1

- Fila 4 = 1 + 1 = 2

Total = 1 + 2 + 0 + 2 = 5

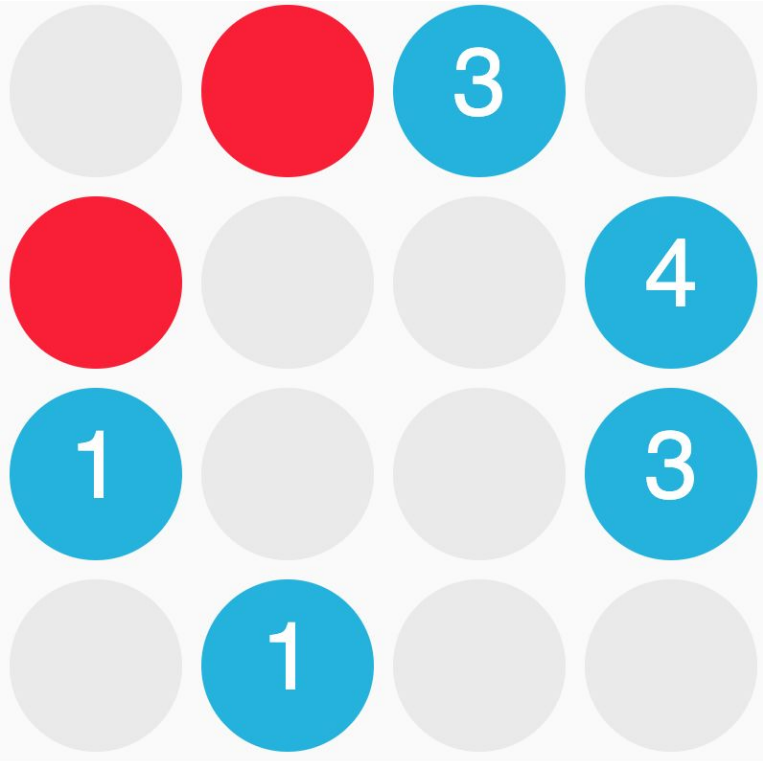
H4(x) = 9 - 5 = 4

Ejemplo Heurística 5 (Missing Reds Heuristic)



Analicemos el tablero:
Tamaño tablero: 16
Suma totall de números azules: 2 + 2 + 2 + 3 = 9
Fijas: 5
Rojos no Fijos: 0
H5(x)l = 16 - 9 - 5 - 0 = 2

Ejemplo Heurística 6 (Missing Visible Blues Heuristic)



Analicemos el tablero:

Primer Fila: Al 3 azul le faltan ver 3.

Fila 1 = 3

Segunda Fila: Al 4 azul le faltan ver 3.

Fila 2 = 3

Tercer Fila: Al 1 azul le falta ver 1 y al 3 azul le faltan ver 2

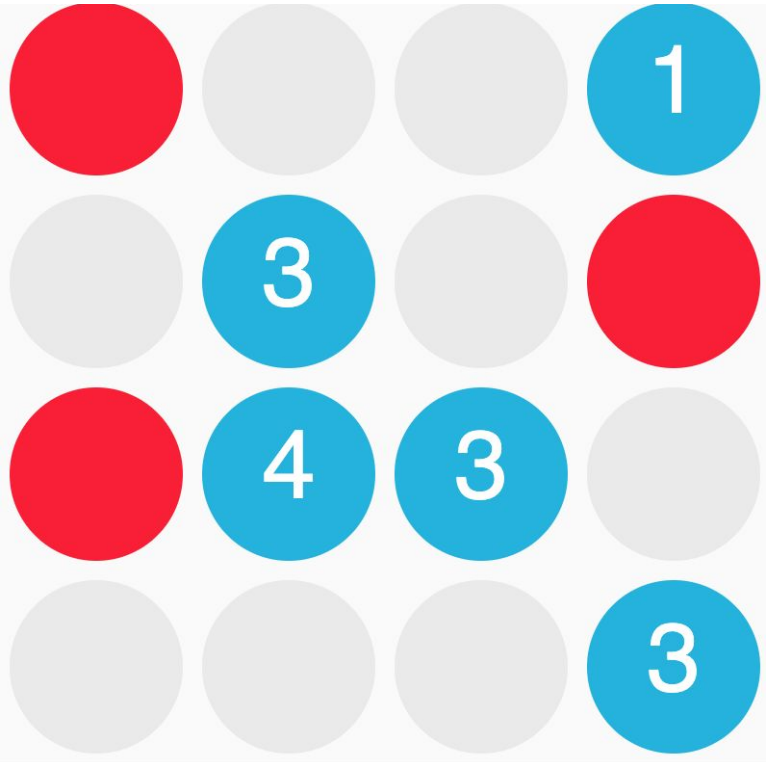
Fila 3 = 1 + 2 = 3

Cuarta Fila: Al 1 azul le falta ver 1

Fila 4 = 1

H6(x) = 3 + 3 + 3 + 1 = 10

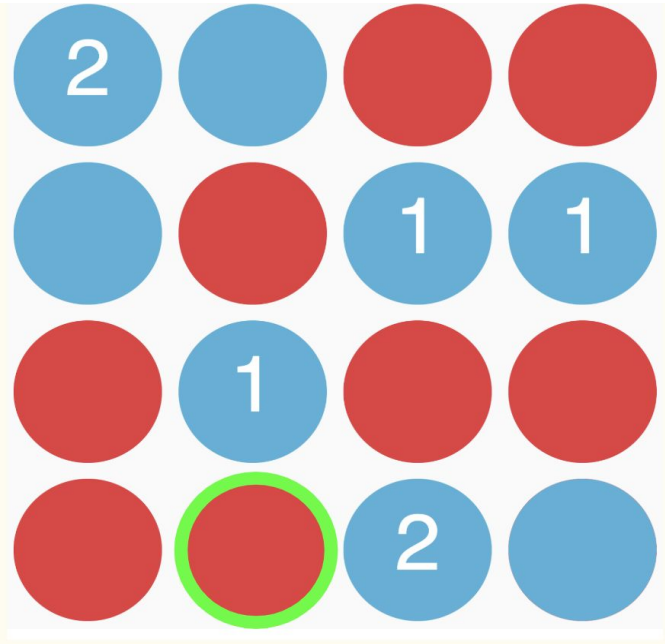
Ejemplo Heurística 7 (Add All Heuristic)



H1 (Fill Blanks) = 8
H3 (Approximate Reds) = 0
H4 (Missing Reds) = 0
H5 (Missing Visible Blues) = 10
H7(x) = 8 + 0 + 0 + 10 = 18

Contraejemplos de Admisibilidad de Heurísticas

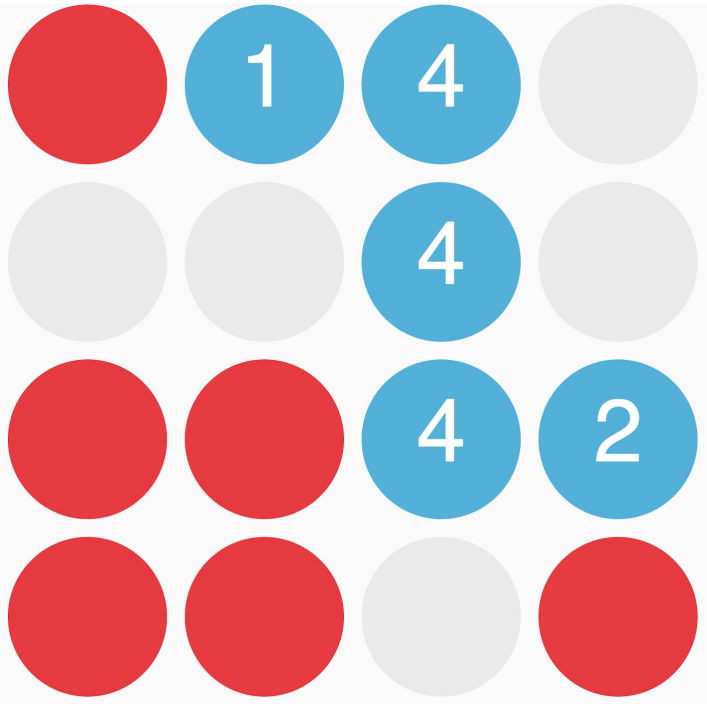
Contraejemplo Heurística 1 (Conflicting Numbers Heuristic)



Como se puede observar hay solo dos números en conflicto por lo que H1(x) = 2 y sin embargo si cambiamos el círculo rojo enmarcado con verde por uno azul llegamos a un estado metacon costo real de 1.

Por lo tanto no es una heurística admisible.

Contraejemplo Heurística 4 (Approximate Reds Heuristic)



Este tablero tiene:

Blancos: 5

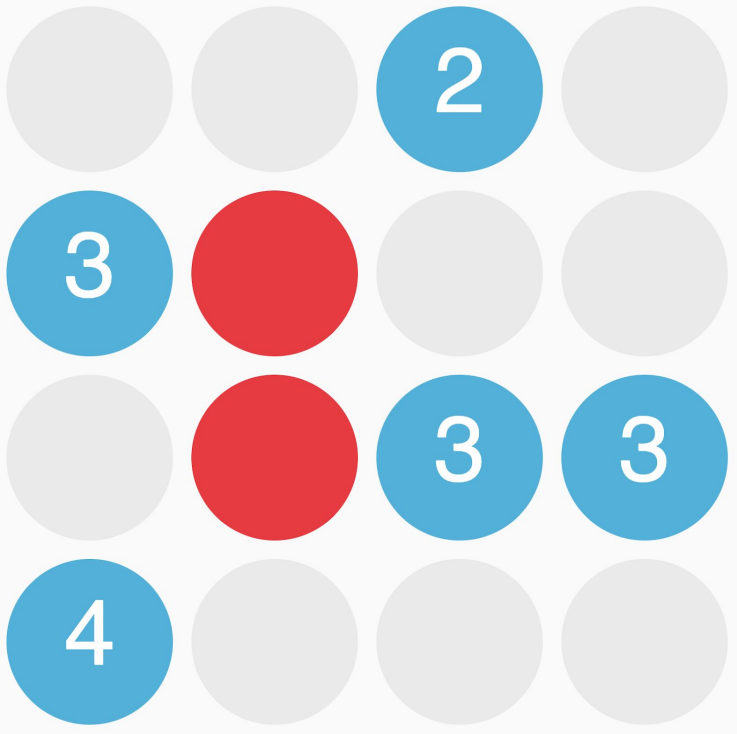
Azules faltantes: 0 + 1 + 2 + 1 + 1 = 5

El valor de la heurística es entonces H4(x) 5 - 5 = 0

y no es un estado meta pues hay restricciones no satisfechas y hay puntos blancos.

Por lo tanto no es una heurística admisible

Contraejemplo Heurística 5 (Missing Reds Heuristic)



La heurística de este tablero es:

Tamaño tablero= 16

Total números azules: 2 + 3 + 3 + 3 + 4 = 15

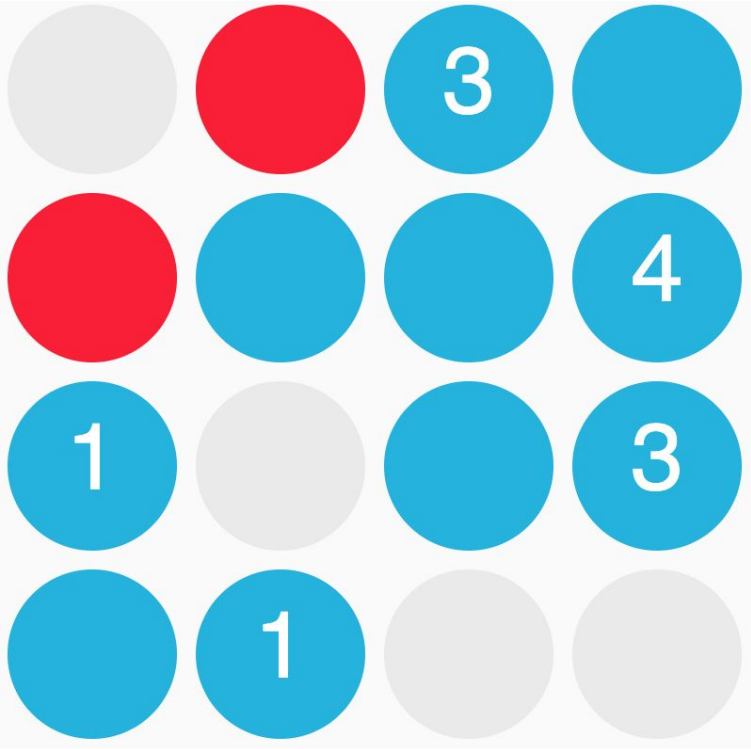
Fijas: 5 (solo los números)

Rojos no Fijos: 2

H5(x) = Max(16 - 15 - 5 - 2, 0) = 0 y no es estado meta pues tiene restricciones sin cumplir y blancos.

Por lo tanto no es una heurística admisible.

Contraejemplo Heurística 6 (Missing Visible Blues Heuristic)



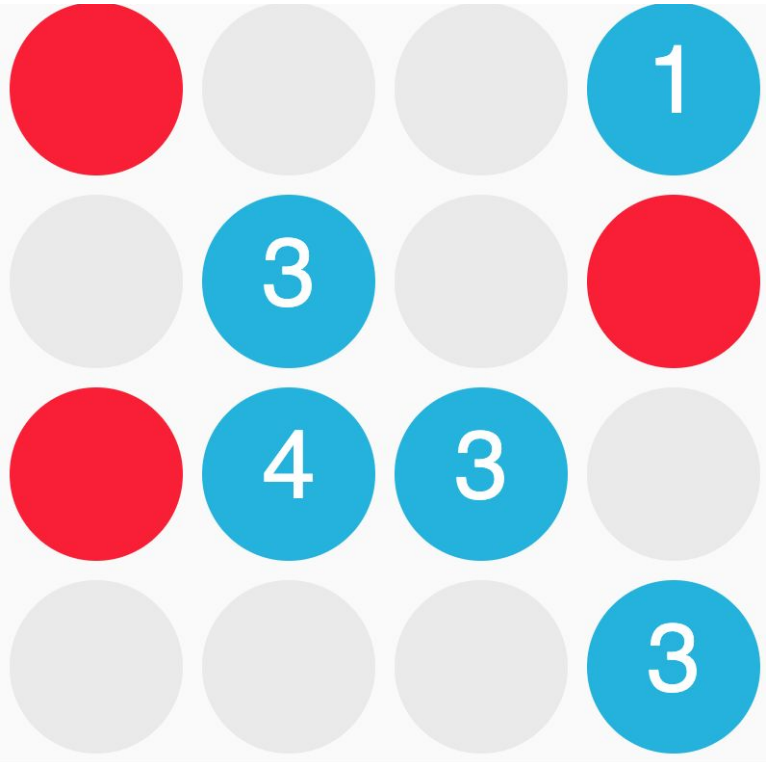
Como se puede ver todos los azules con numero satisfacen su restricción, no les falta ningun azul y por lo tanto

H6(x) = 0

y no es un estado meta ya que tiene blancos.

Por lo tanto no es una heurística admisible

Contraejemplo Heurística 7 (Add All Heuristic)



H1 (Fill Blanks) = 8

H3 (Approximate Reds) = 0

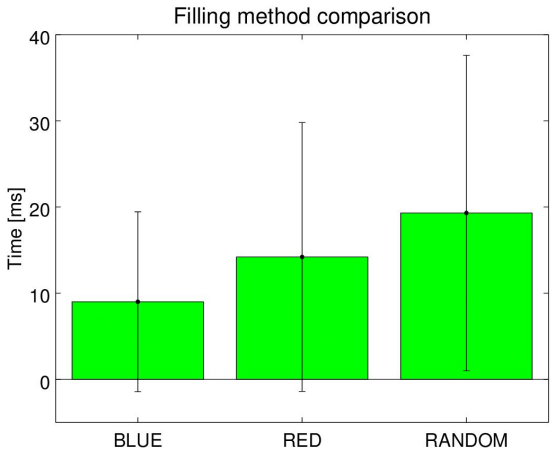
H4 (Missing Reds) = 0

H5 (Missing Visible Blues) = 10

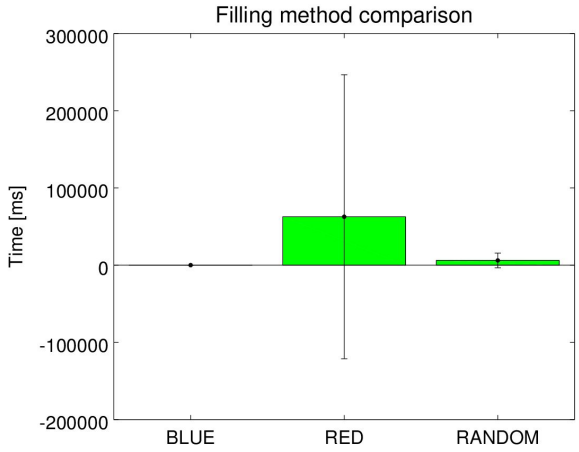
H7(x) = 8 + 0 + 0 + 10 = 18

No es admisible ya que hay 8 blancos, la heurística debería ser ≤ 8

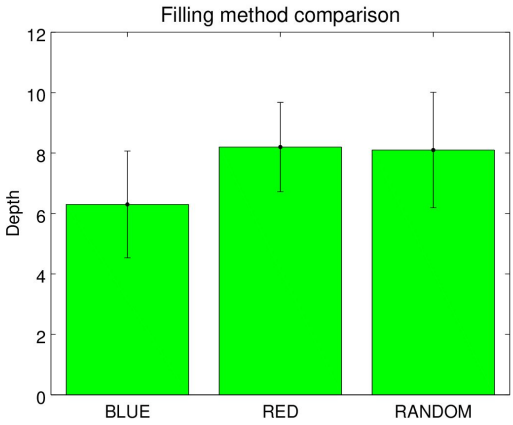
Gráficos



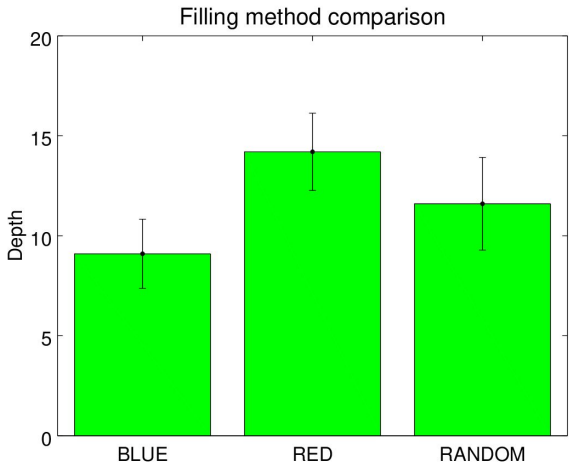
1. Comparación de llenado inicial en tableros 5x5



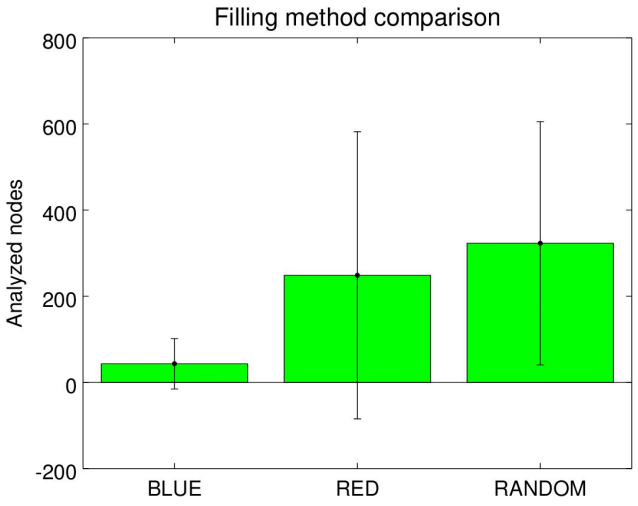
2. Comparación de llenado inicial en tableros 6x6



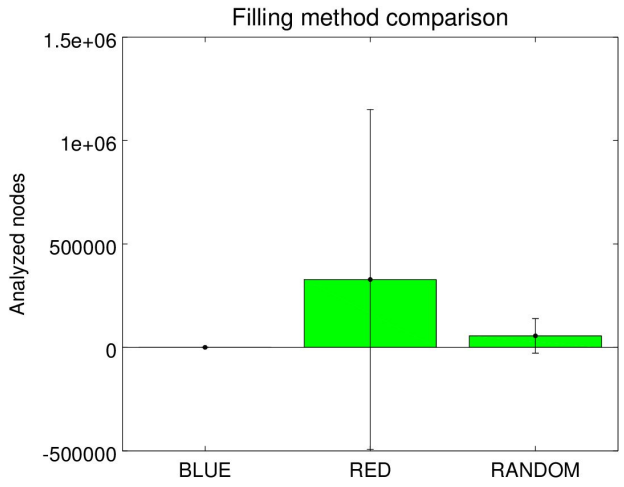
3. Comparación de llenado inicial en tableros 5x5 (profundidad)



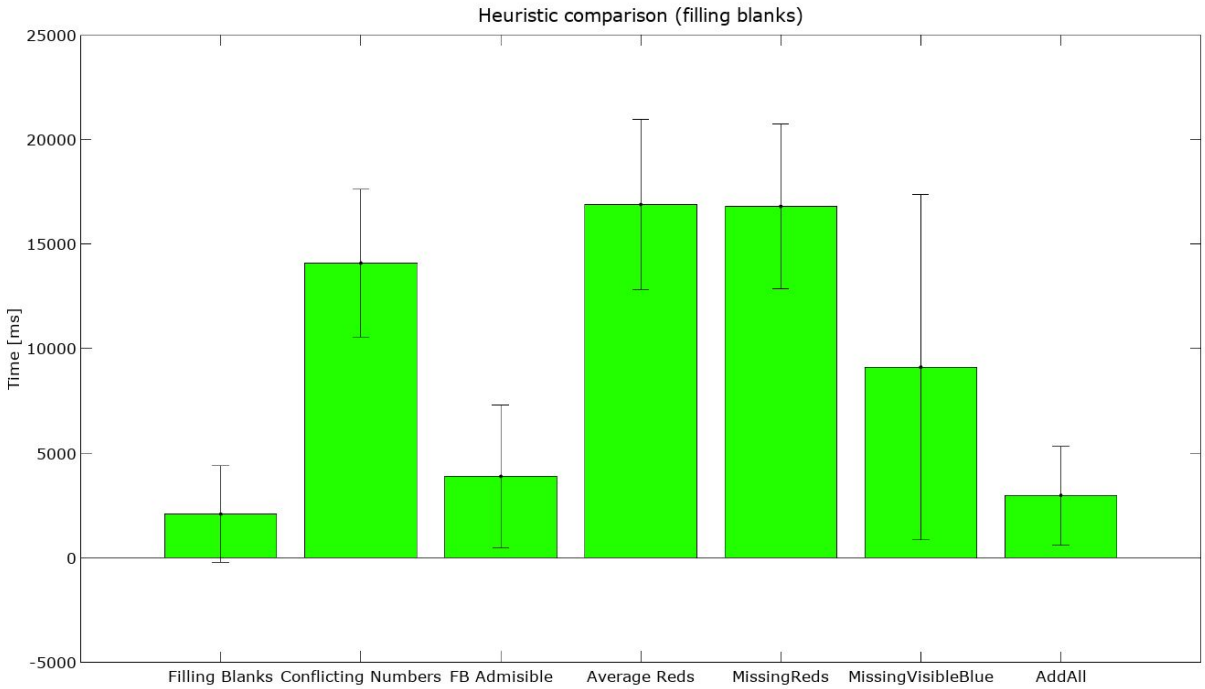
4. Comparación de llenado inicial en tableros 6x6 (profundidad)



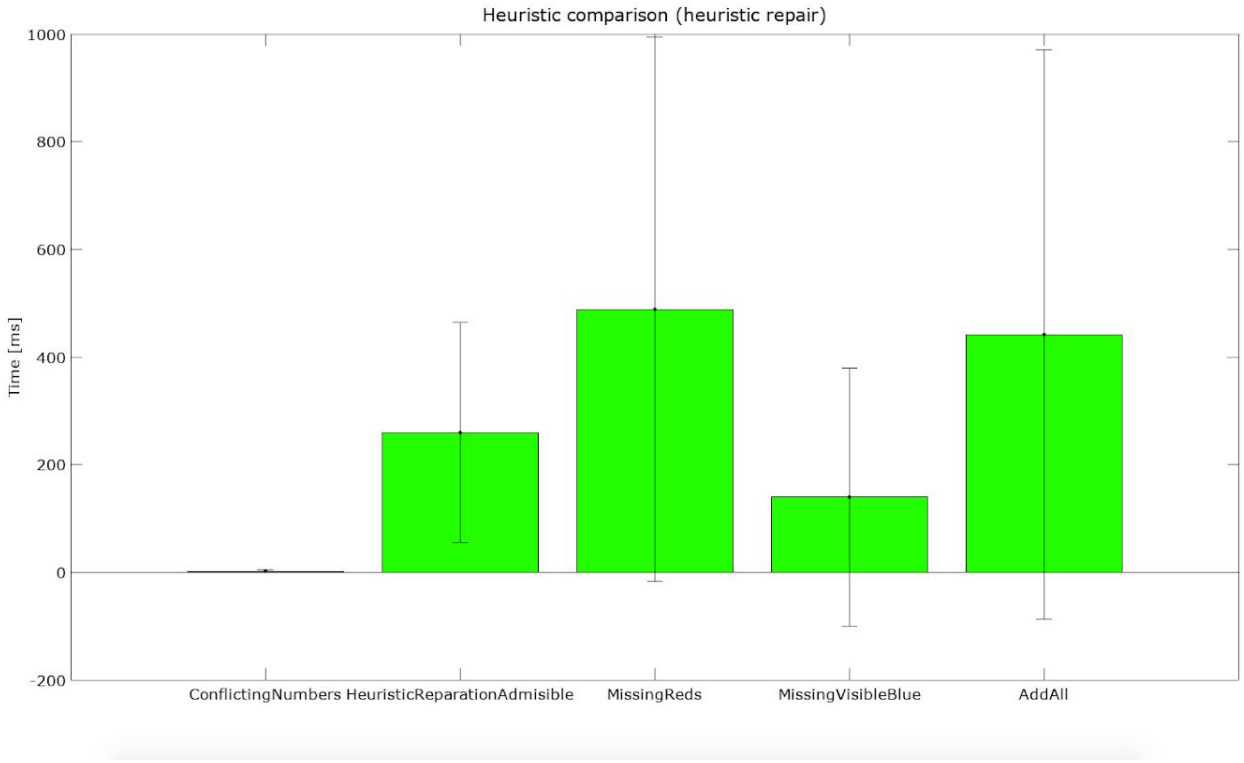
5. Comparación de llenado inicial en tableros 5x5 (nodos analizados)



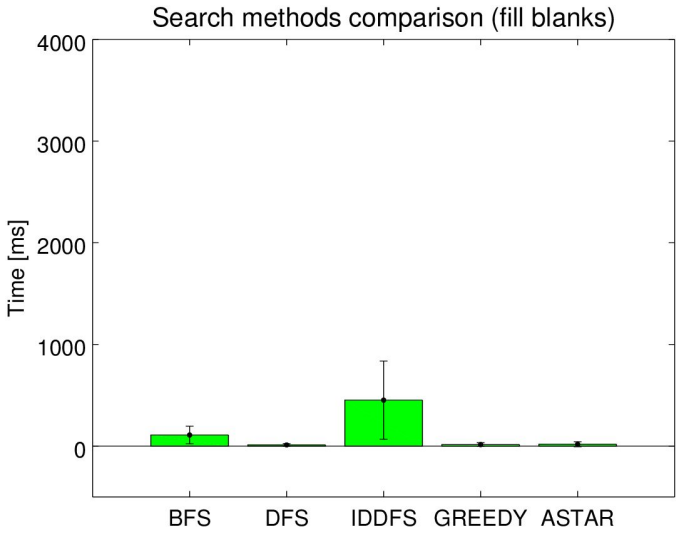
6. Comparación de llenado inicial en tableros 6x6 (nodos analizados)



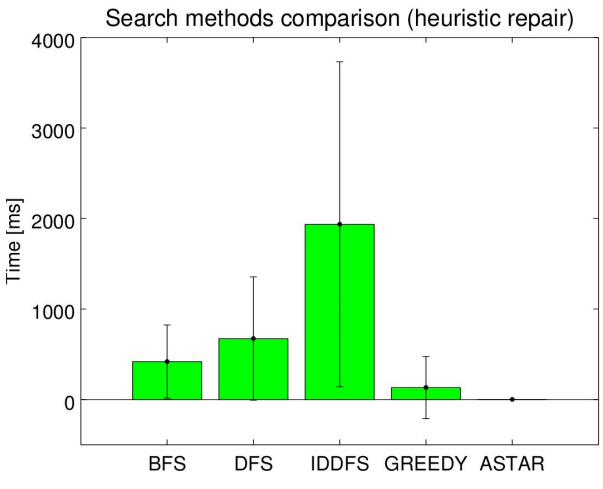
7. Comparación entre heurísticas (formulación incremental)



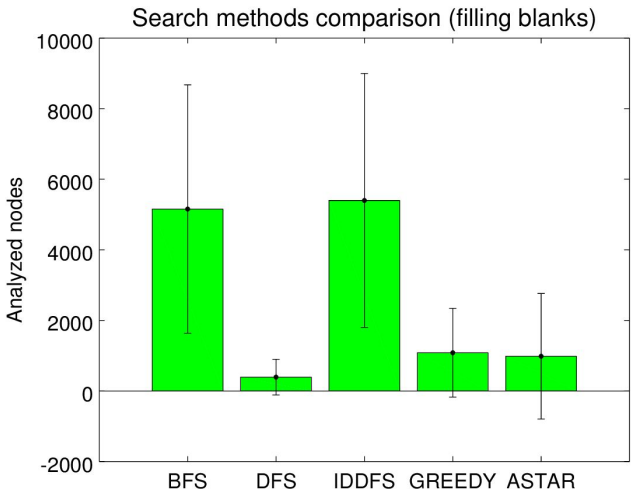
8. Comparación entre heurísticas (reparación heurísticas)



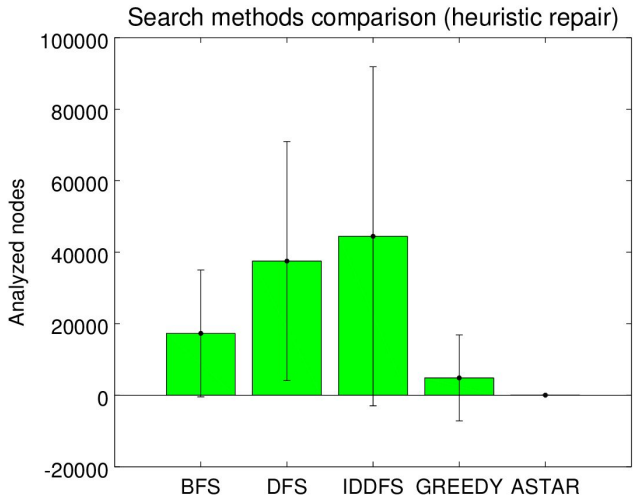
9. Comparación entre métodos de búsqueda (forumlación incremental)



10. Comparación entre métodos de búsqueda (reparación heurística)



11. Comparación entre métodos de búsqueda (nodos analizados con forumlación incremental)



12. Comparación entre métodos de búsqueda (nodos analizados con reparación heurística)