Proyecto final
Simulación Computacional

Johan Sebastian Marulanda A, Manuel Alejandro Victoria, Cristhian Rendon Sanchez

Departamento de Ingeniería, Universidad del Valle, Tuluá
johan.marulanda@correounivall.edu.co
manuel.victoria@correounivalle.edu.co
cristhian.rendon@correounivalle.edu.co

***Abstract*—En este proyecto se realiza un análisis de un juego conocido como Su-Domino-Ku que tiene como objetivo llenar un tablero de un sudoku con fichas de dominó para ello, nos piden implementar tres tipos de algoritmos para solucionar este problema, los cuales son: El algoritmo de búsqueda por amplitud, un algoritmo las vegas tipo 1, y un algoritmo las vegas tipo 2. La información obtenida de dichos resultados la debemos de analizar y con ello llegar a unas conclusiones a lo largo de este articulo.**

Ahora bien, el algoritmo de búsqueda por amplitud usa una estrategia determinística que se encarga de evaluar todos los escenarios posibles, ya que esta genera un árbol con todas las posibles combinaciones de tableros en donde para encontrar la solución se tiene que llenar todo el tablero de forma correcta, lo que nos lleva a depender de la estructura del tablero y de los buenos recursos de la maquina en que se ejecute el algoritmo ya que búsqueda por amplitud es muy costoso tanto en tiempo como en recursos; en cambio el algoritmo las vegas de tipo 1 se aplica en situaciones donde una algoritmo determinístico en su caso promedio tiene una eficiencia más alta que en su caso peor, por lo que con el uso de aleatorización convierte su peor caso en el caso promedio como vamos a apreciar más adelante, en cuanto al algoritmo las vegas de tipo 2 este toma decisiones al azar para resolver el problema (en este problema en específico sus decisión aleatorizada es la ficha que escoge y la casilla en blanco que escoge) aunque este algoritmo fracase reconoce su error y vuelve a aplicar el algoritmo.

***Index Terms*—Búsqueda por amplitud, Algoritmo las vegas tipo 1, Algoritmo las vegas tipo 2, Su-Domino-Ku.**

I. Introduction

Sudoku:
Un Sudoku es un tablero en el cual se intenta representar un rompecabezas el cual tenemos que resolver mediante lógica. Las reglas básicas para representar este sudoku son:

- El sudoku se presenta normalmente como una tabla de 9×9, compuesta por subtablas de 3 × 3 denominadas "sectores".
- Algunas celdas ya contienen números, conocidos como "números dados" (El objetivo es rellenar las celdas vacías, con un número en cada una de ellas, de tal forma que cada columna, fila y región contenga los números 1–9 solo una vez.
- Además, cada número de la solución aparece solo una vez en cada una de las tres "direcciones

Estrategias de Solución:

Técnicas fáciles
La mayor parte de Sudokus se pueden resolver con dos sencillas técnicas:

- Técnica 1: Posición Única.
- Técnica 2: Candidatos Únicos.

Técnicas Medias
Otras técnicas que involucran más que una observación sencilla, son:

- Técnica 3:Líneas de Candidatos
- Técnica 4:Parejas dobles
- Técnica 5:Líneas múltiples

Técnicas Avanzadas
Siguiendo la progresión, se tiene:

- Técnica 6: Parejas o Tripletas desnudas
- Técnica 7: Parejas o Tripletas escondidas

Técnicas de Maestro
Una vez dominadas las anteriores técnicas, se pueden aplicar técnicas más complicadas.

- Técnica 8: X-Wing
- Técnica 9: Swordfish
- Técnica 10: Forcing Chains

Resolución por ordenador:
Como aprendimos en el curso de FADA, la resolución por ordenador es relativamente sencilla de implementar pero con el método de backtracking o “vuelta atrás”. Aunque la

.

complejidad computacional encuentra la solución en determinado tiempo de computación.

Sudominoku
Este juego tiene la forma de un Sudoku normal, como si este no fuera lo suficientemente complejo la revista Games Magazine describió una variante del juego que combina el Sudoku y el Domino llamado Su-Domino-Ku,

Ahora bien, si se desea saber de dónde surgió esta variante del juego, según lo que investigamos fue propuesta en 2006 en un hilo de la página boardgamegeek, por Clark D. Rodeffer, para luego ser probado y publicado, adjunto dicho link https://boardgamegeek.com/thread/122410/sudominoku

Su-Domino-ku es un rompecabezas de dominó solitario donde el objetivo es completar una grilla de sudoku.

Estrategias de juego:
Si quieres tener alguna esperanza de resolver el acertijo, hay dos restricciones para la colocación de los marcadores que debes seguir:

- Dado que cada domino ocupa dos espacios, no coloques los marcadores para que encierren un número impar de espacios vacíos, ya sea por sí mismos o contra un borde o esquina del tablero.

- Si desea colocar los marcadores para que dejen una sola fila de espacios abiertos a lo largo de un borde del tablero que tenga al menos un extremo cerrado, asegúrese de que las ramas cerradas de una longitud impar estén separadas por un número par de espacios / lejos de cualquier extremo cerrado de una sola fila de ancho.

Además dejaré un enlace con una explicación más profunda del juego
http://www.ludism.org/attachments/ppwiki/Sudominoku.pdf

Para resolver el tablero se cuenta con un conjunto de 36 fichas de dominó que se podrán apreciar en la figura 1.2 estas fichas son las suficientes para resolver el problema debido al componente de las rotaciones de estas (0,90,180,270) gracias estas rotaciones fichas que en su posición original no existen , se convierten en otra gracias a su rotación de 180, tomando como ejemplo la primer ficha que es la tupla (1,2) si se aplica la rotación de 180 a esta ficha la tupla que se observa es (2,1)

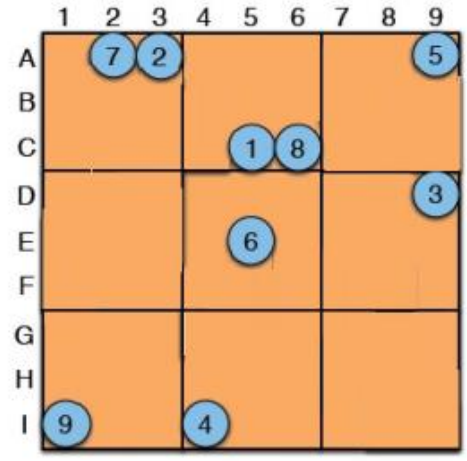

Figura 1: Su-Domino-ku en su estado inicial, que está determinado por los números en círculo azul

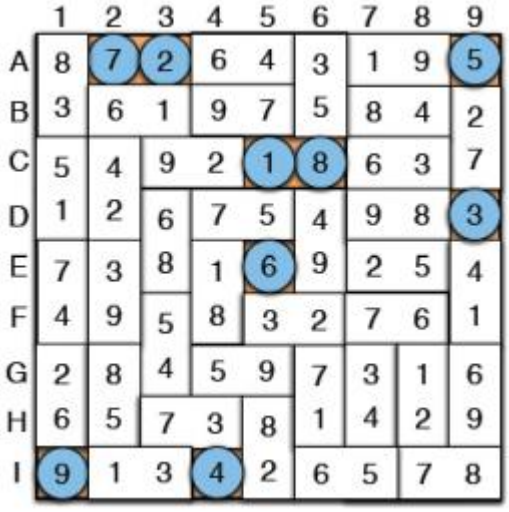

Figura 2: Su-Domino-ku en su estado final

Se espera que a partir de una entrada dada, el programa calcule la solución del Su-Domino-Ku. Las fichas disponibles se muestran en la siguiente figura:

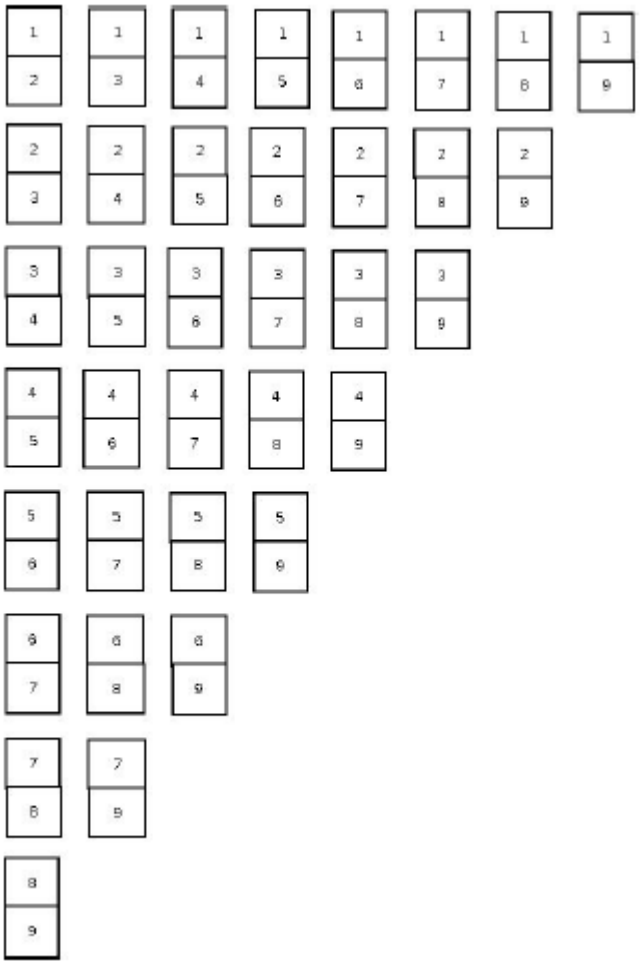

Figura 3: Fichas de dominó

Recuerde la rotación de una ficha:

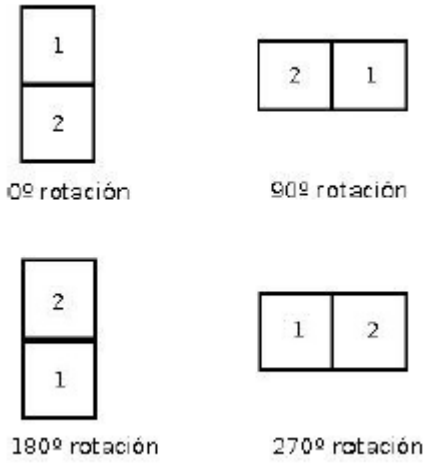

Figura 4: Rotaciones de una ficha de dominó

Se buscan soluciones estocásticas debido a la desagregación del árbol, en el caso de vegas tipo 2 no se genera este (representando un ahorro en la memoria) si no que coloca fichas en posiciones aleatorias, en donde busca una combinación correcta a través de números aleatorios, llegar a esta combinación correcta depende en gran medida de lo que entrega el generador de aleatorios esto quiere decir que no da garantía de encontrar respuesta correcta otra de sus características que tiene una mayor facilidad de implementación, por otro lado a diferencia de vegas tipo 2 búsqueda por amplitud tiene una mayor dificultad de implementación (gracias a la construcción del árbol y su recorrido),pero garantiza encontrar una respuesta ya que este evalúa cada posible combinación de tablero, ficha y posición.

II. METODOLOGIA

En esta sección vamos a describir las 3 técnicas, explicar su funcionamiento y además argumentar los diseños de los algoritmos las vegas:

A. Algoritmo de búsqueda por amplitud

El objetivo de esta técnica es garantizar una respuesta por medio de la exploración de todos los tableros posibles , esto lo logra buscando los espacios en blanco y luego comienza a probar ficha por ficha (además de todas las rotaciones posibles) comprobando cual es la que encaja en esa posición, el algoritmo genera un árbol donde la raíz es el tablero de inicio y se crea un hijo por cada ficha que pone en el tablero , en donde la idea es que cada hijo se expanda hasta donde le es posible, hasta que encuentre solución (por medio de la expansión iterativa de cada hijo)

Diseño e implementación: se deben encontrar los espacios en blanco por cada espacio en blanco se debe probar con cada ficha de la bolsa de fichas si se pueden colocar allí, incluyendo las variantes de las rotaciones (esto quiere decir que toma cada ficha y la prueba a lo largo del tablero para encontrar su lugar correspondiente) estos posibles tablero que se generan del esquema anterior se deben guardar en una estructura que permita su almacenamiento y expansión (simulando el comportamiento de una estructura árbol o también puede ser dicha estructura) estas acciones de crear y expandir nodos dentro de la estructura se harán de forma iterativa

Diagrama de flujo en la Figura 5

B. Algoritmo las vegas tipo 1

El Algoritmo de vegas tipo 1 se centra en convertir en estocásticos métodos deterministas por medio de la aleatorización de ciertos pasos dentro de sus algoritmos, para este caso en específico la idea de la aleatorización se da en el orden de las expansiones de los nodos ya que amplitud en su recorrido original lo realiza de izquierda a derecha, con este factor se permite saltar entre nodos y no realizar la búsqueda de manera consecutiva, se debe contar con un buen generador que prometa no repetir el mismo nodo que ya fue explorado esto se puede apoyar de algún checo antes de expandir

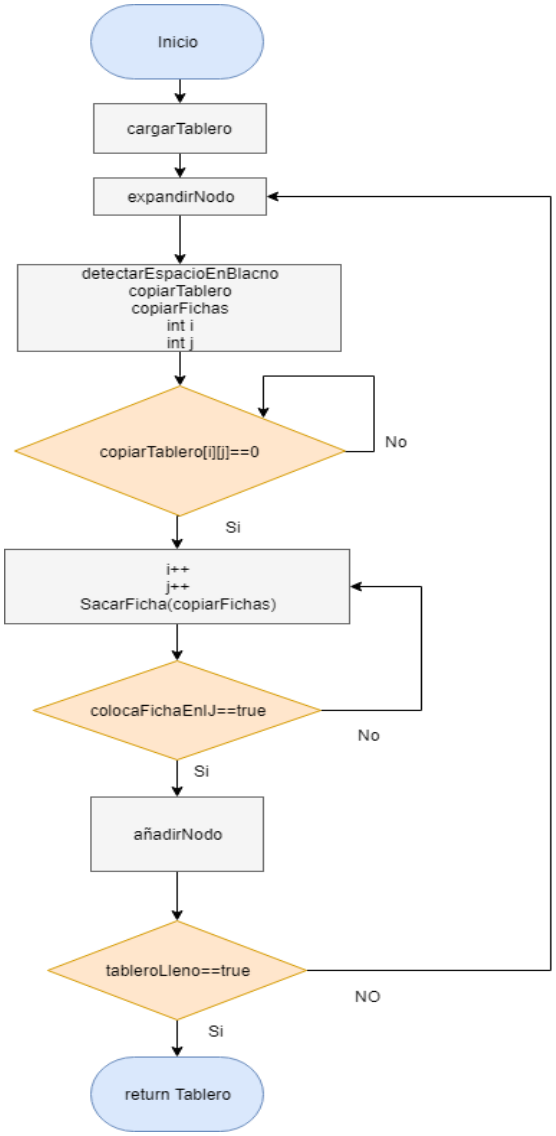

Figura 5: Diagrama de flujo de ejecucion de busqueda por amplitud

C. Algoritmo las vegas tipo II

El objetivo de esta técnica es usar el factor aleatorio para resolver el problema, en nuestro caso, el factor aleatorio lo tomamos como la ficha que escoge, para la cual se implementó una función la cual escoge una ficha aleatoria dados una posición (x,y) en el tablero, y prueba si puede meter la ficha en algún ángulo, si no se puede, vuelve a llamarse recursivamente está función. Además de este método, también se implementó un método para seleccionar una posición aleatoria, el cual genera 2 random (x,y) y valida si estos están vacíos, si no se cumple esta condición se repite hasta que encuentre una posición vacía

Diseño e Implementación:
La solución propuesta inicialmente lee el tablero de entrada, en donde están todas las posiciones del tablero, posterior a esto el programa escoge aleatoriamente una casilla vacía (en otras palabras que tienen un 0 en su posición) y con esa posición, busca aleatoriamente entre la lista de fichas, la primera ficha que pueda poner en esa posición (probando con los ángulos dados), el algoritmo admite que falló cuando saca una ficha y no puede ponerla en ningún Angulo dado. En este caso calcula los promedios (Max y min de fichas usadas) y cantidad de fichas usadas (Como se muestra en la sección de pruebas)

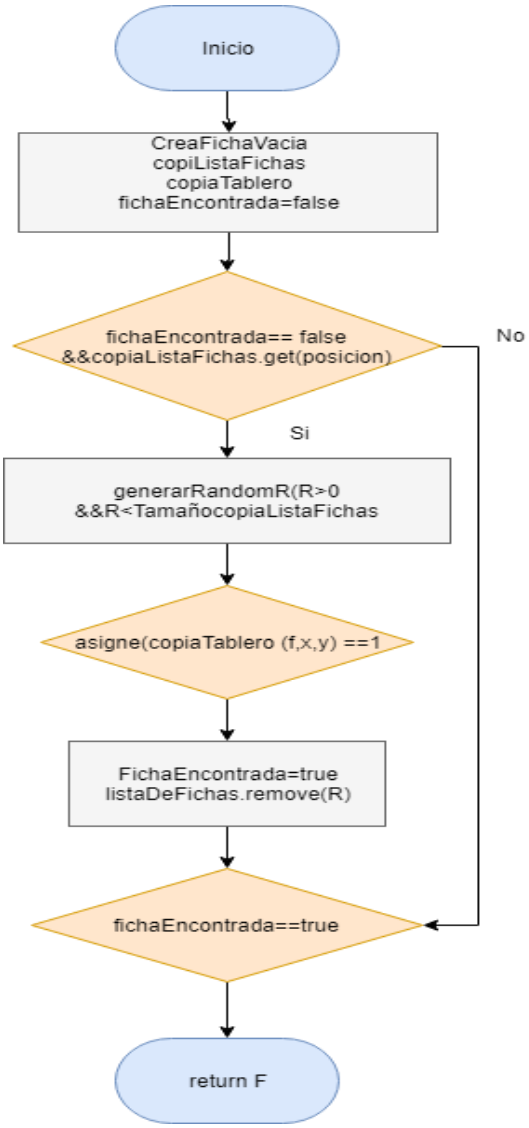

Figura 6: Diagrama de flujo de ejecucion de busqueda por amplitud

III. RESULTADOS

A. *Algoritmo de búsqueda por amplitud*

```
Acabo de generar este muchachon: !
 ??????????????????
0 7 2 0 0 0 0 0 5
0 0 1 2 0 0 0 0 0
0 0 0 0 1 8 0 0 0
0 0 0 0 0 0 0 0 3
0 0 0 0 6 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 4 0 0 0 0 0
No puede poner el dato 2 en [4][4]
BUILD STOPPED (total time: 2 minutes 51 seconds)

Acabo de generar este muchachon: !
 ??????????????????
0 7 2 0 0 0 1 3 5
0 0 1 2 0 0 0 0 0
0 0 0 0 1 8 0 0 0
1 4 0 0 0 0 0 0 3
0 0 0 1 6 0 0 0 0
0 0 0 0 0 0 0 1 6
0 1 7 0 0 0 0 0 0
0 0 0 0 0 1 8 0 0
9 0 0 4 0 0 0 0 0
No puede poner el dato 6 en [2][3]
BUILD STOPPED (total time: 4 minutes 23seconds)

Acabo de generar este muchachon: !
 ??????????????????
0 7 2 0 0 0 1 3 5
0 0 1 2 0 0 0 0 0
0 0 0 0 1 8 0 0 0
0 0 0 0 0 0 0 0 3
0 0 0 1 6 0 0 0 0
1 4 9 0 0 0 0 1 6
0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 8 0 9
9 0 0 4 0 0 0 0 1
No puede poner el dato 7 en [2][3]
BUILD STOPPED (total time: 3 minutes 15seconds)
```

B. *Algoritmo las vegas tipo 1*

```
 ??????????????????
0 7 2 0 0 0 0 0 5
0 0 0 0 0 0 0 2 4
0 0 0 0 1 8 0 0 0
2 9 0 0 0 0 0 0 3
0 5 7 0 6 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
BUILD STOPPED (total time: 1 minute 6 seconds)
```

C. *Algoritmo las vegas tipo II*

Se contemplaron 50 pruebas, la diferencia fue el número de iteraciones que se realizaba por prueba, en dichas pruebas se adjunta el resultado de las simulaciones hechas, como se puede observar esto es muy dependiente del azar, ya que hay algunas simulaciones en donde con un menor número de fichas logró tener un mayor número de fichas colocadas en el tablero (en el documento de la entrega están los resultados de cada una de estas pruebas).

Con 50 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 50
Maximas piezas usadas 10
Minimas piezas usadas 0
Promedio piezas usadas 2.28
BUILD SUCCESSFUL (total time: 4 seconds)
```

Figura 5: Resultados de la primera prueba con 50 iteraciones (Los otros resultados de esta simulación están en la carpeta “Pruebas las Vegas 2”

Con 100 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 100
Maximas piezas usadas 16
Minimas piezas usadas 0
Promedio piezas usadas 1.79
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figura 6: Resultados de la primera prueba con 100 iteraciones (Los otros resultados de esta simulación están en la carpeta “Pruebas las Vegas 2”

Con 150 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 150
Maximas piezas usadas 14
Minimas piezas usadas 0
Promedio piezas usadas 2.4066666666666667
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figura 7: Resultados de la primera prueba con 150 iteraciones (Los otros resultados de esta simulación están en la carpeta “Pruebas las Vegas 2”

Con 200 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 200
Maximas piezas usadas 11
Minimas piezas usadas 0
Promedio piezas usadas 2.205
BUILD SUCCESSFUL (total time: 4 seconds)
```

Figura 8: Resultados de la primera prueba con 200 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Con 500 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 500
Maximas piezas usadas 17
Minimas piezas usadas 0
Promedio piezas usadas 1.964
BUILD SUCCESSFUL (total time: 6 seconds)
```

Figura 9: Resultados de la primera prueba con 500 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Con 600 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 600
Maximas piezas usadas 14
Minimas piezas usadas 0
Promedio piezas usadas 2.0883333333333334
BUILD SUCCESSFUL (total time: 7 seconds)
```

Figura 10: Resultados de la primera prueba con 600 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Con 700 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 700
Maximas piezas usadas 15
Minimas piezas usadas 0
Promedio piezas usadas 2.2342857142857144
BUILD SUCCESSFUL (total time: 9 seconds)
```

Figura 11: Resultados de la primera prueba con 700 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Con 800 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 800
Maximas piezas usadas 15
Minimas piezas usadas 0
Promedio piezas usadas 2.06125
BUILD SUCCESSFUL (total time: 10 seconds)
```

Figura 12: Resultados de la primera prueba con 800 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Con 1000 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 1000
Maximas piezas usadas 14
Minimas piezas usadas 0
Promedio piezas usadas 1.977
BUILD SUCCESSFUL (total time: 11 seconds)
```

Figura 13: Resultados de la primera prueba con 1000 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Con 2000 iteraciones:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 2000
Maximas piezas usadas 21
Minimas piezas usadas 0
Promedio piezas usadas 1.952
BUILD SUCCESSFUL (total time: 23 seconds)
```

Figura 14: Resultados de la primera prueba con 2000 iteraciones (Los otros resultados de esta simulación están en la carpeta "Pruebas las Vegas 2"

Además de las simulaciones anteriores, quisimos contemplar algo aún más amplio, y ya que nos quedó imposible conseguir cuenta en google cloud, dejamos un PC corriendo una simulación con 1millon de iteraciones toda la noche. Este fue el resultado que contemplamos:

```
Ejecuciones Exitosas: 0
Ejecuciones Fracasadas: 1000000
Maximas piezas usadas 24
Minimas piezas usadas 0
Promedio piezas usadas 2.066708
BUILD SUCCESSFUL (total time: 471 minutes 18 seconds)
```

NOTA: Todos los resultados de estas simulaciones los puede observar completos en la carpeta "Pruebas las Vegas 2".

IV. DISCUSIÓN DE RESULTADOS

Comparación con el método determinista:

Ya que en los algoritmos propuestos nos basamos en la información que tenemos (en el caso de las vegas 1 y 2 la búsqueda de los nodos vacíos aleatorios, y por amplitud la generación de los hijos) dado eso se selecciona lo que vamos a

llenar por ende se ve fuertemente afectado el rendimiento del algoritmo ya que se ve afectado por los for y comparaciones que realiza, a diferencia del método determinista. Donde se queda con la primera casilla vacía que encuentre y la primera ficha que se encuentre en la lista de fichas que faltan por usar.

Teniendo en cuenta lo anterior se puede asegurar que el método determinista tiene un 100% de acierto mientras que en los métodos de las vegas este porcentaje se ve reducido.

V. CONCLUSIONES DEL EJERCICIO

- En el caso de las vegas, se podría maximizar los valores aleatorios para que la búsqueda por conocimiento de una matriz se reduzca. En otras palabras, considerar un valor intermedio de conocimiento y con este encontrar el primer elemento con este valor, lo que puede ayudar a disminuir el número de cálculos y mejorar el tiempo de respuesta.
- El método de amplitud al generar un hijo por cada posición que puede ocupar una ficha, hace que su expansión sea regresiva es decir que la cantidad de nodos hijos por nivel del árbol se reducirá a mayor profundidad, siendo así que del ultimo nodo padre se desprenderá máximo 4 hijos (ya que solo falta una casilla por asignar)
- Una forma de reducir el consumo de memoria es realizar la comprobación de éxito solo cuando falte una ficha
- En vegas tipo uno la idea de aleatorizar la selección del nodo se debe a que el recorrido en amplitud va de izquierda a derecha lo que significa crear y luego expandir los nodos en este orden, si se aleatoriza el nodo que va expandir se tiene una la ventaja de que si la solución está en el final de los nodos creados no se tiene que recorrer los n nodos

VI. BIBLIOGRAFIA

[1]http://dis.um.es/~domingo/apuntes/AlgProPar/0607/AlgoritmosProbabilistas.pdf

[2]https://es.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/breadth-first-search-and-its-uses

[3]https://campusvirtual.univalle.edu.co/moodle/course/view.php?id=34482

[4]Enunciado Proyecto Simulación Computacional, Universidad del Valle 2018

REFERENCES

FALTA ESTA MIERDA

Basic format for books:

J. K. Author, “Title of chapter in the book,” in *Title of His Published Book*, *x*th ed. City of Publisher, (only U.S. State), Country: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx.*

Examples:

[1] G. O. Young, “Synthetic structure of industrial plastics,” in *Plastics*, 2nd ed., vol. 3, J. Peters, Ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15–64.

[2] W.-K. Chen, *Linear Networks and Systems*. Belmont, CA, USA: Wadsworth, 1993, pp. 123–135.

Basic format for periodicals:

J. K. Author, “Name of paper,” *Abbrev. Title of Periodical*, vol. *x*, *no*. *x*, pp. *xxx-xxx,* Abbrev. Month, year, DOI. 10.1109.*XXX*.123456.

Examples:

[3] J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility,” *IEEE Trans. Electron Devices*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959, 10.1109/TED.2016.2628402.

[4] E. P. Wigner, “Theory of traveling-wave optical laser,” *Phys. Rev*., vol. 134, pp. A635–A646, Dec. 1965.

[5] E. H. Miller, “A note on reflector arrays,” *IEEE Trans. Antennas Propagat.*, to be published.