Universidad Nacional Autónoma de México

posgrado en ciencia e ingeniería de la computación

"EMPUJANDO FICHAS"

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:

MICHEL ABDUL MASSIH SAID

DIRECTOR DE TESIS: DR. JORGE URRUTIA GALICIA

MÉXICO D.F. 2009

Agradecimientos

Agradezco a mis padres Joseph y Nadia, a mi hermana Marsel y mi novia Argelia. Su apoyo y paciencia hicieron posible este trabajo.

También agradezco de todo corazón al director de esta tesis, el Dr. Jorge Urrutia, quien no sólo ha sido un maestro excepcional, sino un guía para el futuro y una inspiración.

A mis compañeros y amigos les agradezco los consejos y palabras de aliento, especialmente (en orden cronológico) a: Chelo, Quique, Wally, Gus, Beavis, Tania, Berna, Vic, Ale, Lalo y Miguel.

Estoy muy agradecido con los Doctores José de Jesús Galaviz Casas, Francisco Hernández Quiroz, Juan José Montellano Ballesteros y Sergio Rajsbaum Gorodezky por el tiempo dedicado a este trabajo y sus valiosos comentarios.

Finalmente agradezco a la Universidad Nacional Autónoma de México por haberme dado la oportunidad de llevar a cabo mis estudios de posgrado en tan magnífica institución y al Consejo Nacional de Ciencia y Tecnología por el apoyo económico durante este tiempo.

Índice general

Índice de figuras

Capítulo 1

Prefacio

Los juegos (o rompecabezas) en los que se tienen que deslizar o empujar fichas rectangulares, bloques convexos o monedas han sido desde hace más de un siglo de mucho interés en el área de las matemáticas ya que por un lado, presentan un reto y una forma de entretenimiento para cualquier persona dado que no se necesitan conocimientos matemáticos para resolverlos; por otro lado cuentan con una alta complejidad que llama la atención, especialmente a la gente en el área de computación ya que son muy interesantes por su complejidad computacional y por sus espacios de búsqueda tan amplios.

Uno de los juegos de este estilo más importantes sin duda es *El juego de 15* (*The fifteen puzzle*), atribuido a Sam Lloyd, ver [4, 5, 6, 8, 9, 10], que consiste en quince bloques unitarios cuadrados movibles, numerados del 1 al 15 en un tablero de cuatro por cuatro y cuyo objetivo es, partiendo de una configuración válida, ordenar los bloques del uno al quince dentro del tablero. Existe una generalización del juego de 15 que se llama el juego de $n^2 - 1$. Un detalle interesante de este juego es que no todas las configuraciones iniciales son válidas para llegar a una configuración final, de hecho, de las $n^2!$ permutaciones posibles de inicio, sólo $\frac{n^2!}{2}$ tienen solución ya que, como lo demostró Storey [9] en 1879, una configuración inicial sólo puede ser resuelta si es una permutación par de la configuración final.

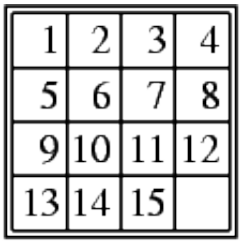


Figura 1.1: El juego de 15.

Richard M. Wilson [10] generaliza el juego de 15 a un rompecabezas basado en una gráfica arbitraria. Después, Kornhauser [5] retoma el problema tomando los bloques como canicas que se trasladan entre vértices de una gráfica.

Ratner y Warmuth [8] demostraron que el problema de encontrar el mínimo número de movimientos necesarios para llegar de una configuración inicial a una final en el juego es NP-difícil por medio de la reducción del problema $2/2/4-SAT$ al problema REL y ayudándose de ésta para llevar a cabo la reducción de $2/2/4-SAT$ al juego n^2-1. REL consiste en trasladar elementos que se encuentran en vértices de una gráfica hacia sus vértices destino por medio de una trayectoria euleriana de esta gráfica sobre aristas con capacidad cero o dos. Es también NP-difícil encontrar una solución que esté a una constante aditiva de la solución óptima. Ellos proponen un algoritmo que está a una constante multiplicativa de la solución óptima, aunque dicha constante es muy grande. Después, Parberry [6] propone un algoritmo *greedy* y *divide y vencerás* que da una solución con una constante multiplicativa menor para la cota superior que es $5n^3$. También propone una cota inferior calculada con la *distancia Manhattan* entre las posiciones iniciales y finales de los bloques pero esta cota puede ser mejorada.

El juego n^2-1 es muy utilizado para probar algoritmos de búsqueda. Östergård [4] encuentra soluciones completas a varias configuraciones pequeñas del juego mediante el algoritmo BFS. Primero obtiene la gráfica del juego asociando cada posición posible a un vértice e insertando aristas entre los vértices si las posiciones correspondientes se pueden obtener en un solo movimiento. Es necesario después obtener los puntos a máxima distancia δ para conocer el mínimo número de movimientos requeridos en el peor caso. También da una cota superior

para el juego de 24 que es $\delta \leq 210$. Con esto podemos ver que incluso para tamaños de n pequeños, es muy difícil encontrar el mínimo número de movimientos requerido para resolver el juego. De hecho, si se usan técnicas de búsqueda como esta, para resolver el juego de 15, se necesitarían aproximadamente 1300 GB de memoria.

Este juego puede ser generalizado a monedas marcadas. Esto puede ser interesante desde el punto de vista de *motion planning* en robótica. Ejemplos de trabajo en esta área son el de Dumitrescu y Pach [3] con sus discos disjuntos o congruentes en el plano que se deben mover de una configuración inicial a una final teniendo la libertad de seguir trayectorias curvas. El número de movimientos suficiente para llevar una configuración inicial de discos congruentes a una final es $\frac{3n}{2} + O(n \ln n)$ y en caso de que no sean congruentes, $2n$ movimientos son suficientes. Nótese que el área en la que se mueven las monedas no es confinada. Abellanas et all [1] propone de igual manera, con discos disjuntos o congruentes, buscar el mínimo número de movimientos necesarios para llegar de una configuración a otra, pero con trayectorias rectas y en algunos casos dentro de un área definida. Cuando los discos no están confinados a ningún área, $2n - 1$ movimientos son suficientes cuando los discos son congruentes y $2n$ movimientos cuando no lo son y cuando están confinados en un área *muy pequeña*, 6n movimientos son suficientes cuando los discos son congruentes. En este tipo de problemas es común que se utilice el recurso de enviar las monedas una por una al infinito y después regresarlas a su posición final, de esta forma se puede ver muy fácilmente por qué son suficientes $2n - 1$ movimientos en el caso en que los discos no están confinados a un área específica.

Dumitrescu y Pach [3] también proponen algoritmos para la solución de sistemas metamórficos modulares por medio de deslizar bloques desde una configuración inicial conectada (no existen bloques que no sean adyacentes a algún otro) hasta llegar a una configuración final conectada tal que todas las configuraciones intermedias estén conectadas.

En este trabajo se presenta un algoritmo con un mejor resultado que el descrito por Parberry [6] en cuanto al máximo número de movimientos necesarios para llegar de una configuración inicial a una final en el juego $n^2 - 1$. Este algoritmo es más complejo que el de

Parberry ya que contiene bastantes casos especiales dada la técnica para llevar a cabo los movimientos de las fichas. Se implementó este algoritmo utilizando el lenguaje C para poder hacer simulaciones.

Por otro lado, en este trabajo se investiga una variación del juego $n^2 - 1$ en la que el número de celdas vacías en el tablero no está limitado a una sola. En este problema se puede pensar el tablero de $n \times n$ como una bodega cuadrada en la que existen "corredores" y "estantes". Cada estante será un bloque de fichas de tamaño $\sqrt{n} \times \sqrt{n}$ y se trabajará sobre el caso en el que se pueden generar corredores alrededor de cada bloque en el tablero para que las fichas dentro de cada uno de éstos tengan el camino libre para llegar a su destino llevando a cabo un número relativamente pequeño de movimientos.

Se presenta un algoritmo cuya entrada es el tablero de $n \times n$ con $n^2 - 2n\sqrt{n} + n$ fichas acomodadas en bloques y $2n\sqrt{n} - n$ celdas vacías acomodadas en corredores y cuya salida es el tablero con cada ficha en la celda que le corresponde dentro de algún bloque y se analiza el máximo número de movimientos en el peor caso.

Lo interesante de este algoritmo es que acomoda un 99 % de las fichas totales de un tablero grande en menos de la *mitad* de los movimientos en los que el algoritmo del juego $n^2 - 1$ acomoda el total del tablero.

Este tipo de resultados podría ser utilizado en bodegas que necesiten hacer una permutación de los productos que contengan, sabiendo desde el inicio cuántos movimientos son necesarios para ponerla en práctica.

Capítulo 2

Introducción

2.1. El juego de 15 y el juego $n^2 - 1$

El Juego de 15 (*The fifteen puzzle* en inglés) es uno de los rompecabezas más populares del mundo. Consiste de un tablero de 4×4 celdas que contiene 15 fichas numeradas del 1 al 15 y una celda vacía. El objetivo del juego es ordenar las fichas de manera ascendente (como se muestra en la Figura 1.1) a partir de una configuración válida de tal forma que en cada paso, se desliza una ficha del tablero hacia una celda vacía adyacente a la misma. En el resto de este trabajo dicha operación será llamada un movimiento básico o simplemente un *movimiento*. Una configuración válida es una permutación par de la configuración final. En la siguiente sección se analiza a detalle este concepto.

El juego de 15 puede ser generalizado al juego $n^2 - 1$ donde en lugar de tener un tablero de 4×4, se tiene uno de $n \times n$. De la misma manera se tienen que acomodar las $n^2 - 1$ fichas en orden mediante movimientos iguales a los utilizados en el juego de 15.

El problema de encontrar el mínimo número de movimientos necesarios para pasar de una configuración inicial válida a una configuración final es NP-difícil, como lo describen Ratner y Warmuth [8]. Gracias a esto se buscan soluciones que se encuentren a una constante

multiplicativa de la solución óptima.

Como ya se mencionó en el capítulo anterior, existen muchos trabajos en torno al juego $n^2 - 1$. Para nuestro trabajo analizaremos con más detalle el de Parberry [6]. Su algoritmo para resolver el problema en un tablero de $n \times n$ consiste en una técnica *greedy* y *divide y vencerás* en la cuál se ordenan la primera fila y columna del tablero de manera *greedy* y se continúa con las siguientes filas y columnas de manera recursiva hasta que se llega a un tablero de 3×3, el cual se resuelve por fuerza bruta [7] ya que en el peor caso, 30 movimientos son necesarios para resolverlo.

En este trabajo se desarrolla un algoritmo con base en el de Parberry utilizando las técnicas *greedy* y *divide y vencerás* que él utiliza pero con algunas variaciones, de tal manera que se necesitan menos movimientos para llegar a una configuración final en el peor caso. Así mismo se lleva a cabo una implementación del algoritmo con fines de simulación.

2.2. Permutaciones pares

En esta sección, un movimiento (como fue definido en la sección anterior), tendrá el nombre de giro. Demostraremos que no exite una secuencia de giros para obtener la configuración en la Figura 2.1(b) de la configuración en la Figura 2.1(a).

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(a) Juego de 15 con las fichas ordenadas.

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

(b) Juego de 15 con el 14 y 15 invertidos.

Figura 2.1: Configuración del bloque después de introducir la ficha destino y subirla un nivel.Juego de 15 con las fichas ordenadas y 14 y 15 invertidas.

A cada configuración C_i, le podemos asociar una permutación σ_i biunívocamente. σ_i se obtiene leyendo las celdas de C_i por filas.

Cualquier giro en C_i resulta en una configuración con diferente paridad, donde la paridad de una configuración es la paridad de su permutación asociada.

La paridad de una permutación es la paridad de su número de *inversiones*. Una inversión en una permutación ocurre cuando hay un elemento mayor antes de uno menor, como en el siguiente caso:

1 2 3 4 5 6
1 2 3 4 5 6

Número de inversiones = 0.

1 2 3 4 5 6
6 5 4 3 2 1

Número de inversiones = $\binom{6}{2}$

Es bien sabido que aplicar una trasposición a una permutación, resulta en una permutación con paridad diferente.

Una *gráfica de giros* G de las configuraciones en el juego de $n^2 - 1$ es aquella que tiene como vértices al conjunto de todas las configuraciones ($n^2!$), donde dos de ellas son adyacentes si una puede ser obtenida de la otra mediante un giro.

El problema del juego de $n^2 - 1$ puede ser replanteado en términos de las propiedades de una gráfica de giros. En particular mostraremos que no es conexa.

Para esto usamos los siguientes hechos:

a) G es bipartita. Esto es porque dos configuraciones adyacentes están asociadas a dos

permutaciones con diferente paridad.

b) Ir de una configuración con la celda vacía en un lugar a otra con esta celda en el mismo lugar requiere un número par de movimientos. Esto es porque el número de movimientos hacia la izquierda tiene que ser igual al número de movimientos a la derecha (de otra manera la celda vacía terminaría en una columna diferente). De igual manera el número de movimientos hacia arriba es igual al número de movimientos hacia abajo.

Teorema 1 G *no es conexa.*

Por *a)* y *b)*, el conjunto de las configuraciones alcanzables desde I_N tal que la celda vacía se encuentre en la posición N, tiene solamente permutaciones "pares". Si trasponemos los últimos dos elementos de dicha configuración, tendremos una permutación impar $I_N \times (N-1, N-2)$, que no puede ser alcanzada desde I_N.

□

2.3. El problema de la bodega

En este trabajo estudiaremos variantes del juego de 15, en las cuales permitimos que el número de celdas vacías sea mayor que uno. Los movimientos permitidos son iguales a los del juego de 15, i.e. deslizar fichas del tablero a celdas adyacentes vacías.

Es evidente que entre más celdas vacías tenga nuestro tablero, es más fácil desplazar las fichas del mismo para llevarlas de una configuración inicial, a alguna configuración final. En aplicaciones de la vida real, tales como bodegas, es claro que nunca están llenas a tope y con frecuencia es necesario rearreglar la mercancía almacenada en ellas para, por ejemplo, asegurarse que la mercancía vieja sea más accesible que la nueva y de esta manera, asegurar

que la mercancía sea vendida antes de su fecha de expiración. Siempre se dejan corredores que faciliten dichos cambios. En nuestro problema, estos corredores serán representados por familias de celdas vacías en el tablero.

Se presenta un algoritmo que utiliza el concepto de "corredores" para poder transportar las fichas a sus posiciones finales de manera más eficiente dentro de un tablero de $n \times n$, tomando en cuenta que es necesario sacrificar el número total de fichas para poder utilizar estos corredores.

Un corredor será un conjunto de espacios en blanco acomodados de manera vertical u horizontal, adyacente a determinado número de fichas.

El total de fichas se divide en bloques de $\sqrt{n} \times \sqrt{n}$ distribuidos uniformemente dentro del tablero. En este trabajo se estudia el caso en el que cada bloque está rodeado por corredores.

Cuando n es grande, la suma de todas las fichas en los bloques se aproxima mucho al número total de fichas que puede haber en un tablero de $n \times n$, por ejemplo, cuando $n > 40000$, se tiene más del 99 % de la capacidad del tablero ocupada por fichas.

Como se menciona en el capítulo anterior, el algoritmo que utiliza los bloques y corredores acomoda un 99 % de las fichas totales de un tablero grande en menos de la *mitad* de los movimientos en los que el algoritmo del juego $n^2 - 1$ acomoda el total del tablero.

Si se piensa en los bloques como estantes en una bodega o almacén, el algoritmo podría funcionar para hacer permutaciones de los artículos entre estantes y dentro de ellos de manera eficiente.

Capítulo 3

Algoritmo del juego $n^2 - 1$

Para comenzar a describir el algoritmo, se necesita primero definir el concepto de máxima distancia en una gráfica construida mediante permutaciones de fichas en un tablero que se puede encontrar en [4]. A continuación la definición.

Existen $n^2!$ configuraciones de fichas posibles inicialmente en el juego $n^2 - 1$. Una gráfica G puede ser construida asociando cada configuración posible con un vértice y una arista entre dos vértices si y sólo si las configuraciones correspondientes pueden ser obtenidas una de otra en un solo movimiento.

Una posición inicial puede ser llevada a una final si éstas se encuentran en la misma componente conexa. Como ya se vio, el número total de configuraciones válidas para llegar de cualquiera de éstas a cualquier otra es de $\frac{n^2!}{2}$. Östergård en [4] describe varios métodos para encontrar los vértices a distancia máxima δ de la configuración final. En este trabajo utilizamos únicamente la δ de un tablero de 2×2 como base del algoritmo del juego $n^2 - 1$.

Östergård demuestra que la distancia máxima δ en un tablero de 2×2 es 6 de la siguiente manera: Este tablero tiene $\frac{4!}{2} = 12$ posiciones posibles, la celda vacía siempre está en una esquina, así que siempre hay dos movimientos posibles. Ya que la gráfica del juego es conexa y es regular de grado 2, es una gráfica cíclica. Entonces, $\delta = \frac{12}{2} = 6$.

En el Cuadro 3.1 podemos ver las $\frac{4!}{2}$ posibles configuraciones (válidas) de un tablero de 2×2. En la Figura 3.1 se muestran las relaciones entre cada configuración (los vértices) y cuándo existe un sólo movimiento entre dos de ellas (las aristas). Podemos rápidamente notar que la máxima distancia entre cualesquiera dos vértices es 6.

1 2 3	1 3 2	1 3 2	3 1 2
2 3 1	2 1 3	2 1 3	1 2 3
3 1 2	3 2 1	3 2 1	2 3 1

Cuadro 3.1: $\frac{4!}{2}$ configuraciones posibles en el tablero de 2×2

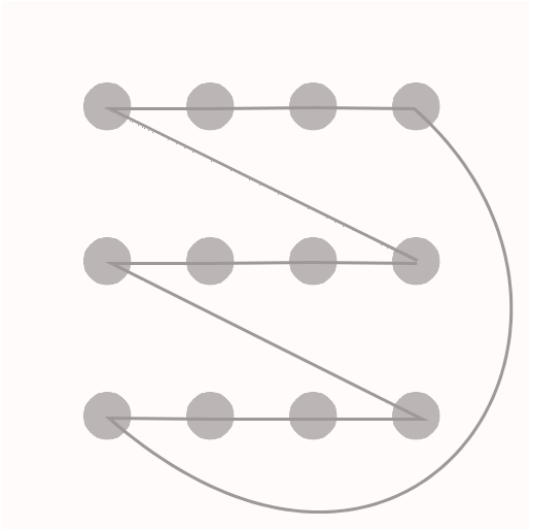


Figura 3.1: Gráfica de configuraciones del tablero de 2×2.

Antes de continuar, es conveniente definir unos cuantos conceptos. Ya se ha hablado del *tablero* que se define como un área cuadrada de $n \times n$ celdas que contiene $n^2 - 1$ fichas acomodadas en n filas y n columnas. La *localidad destino* es la celda que corresponde a una ficha en particular en la configuración final deseada. Una *ficha destino* es aquella que se pretende mover o se está moviendo hacia su localidad destino. La *celda vacía* es es la celda restante en el tablero (Figura 3.2).

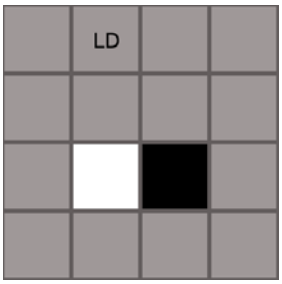


Figura 3.2: Ficha destino. La ficha destino está pintada de negro, la celda vacía de blanco y 'LD' indica la localidad destino.

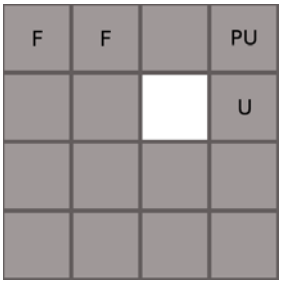


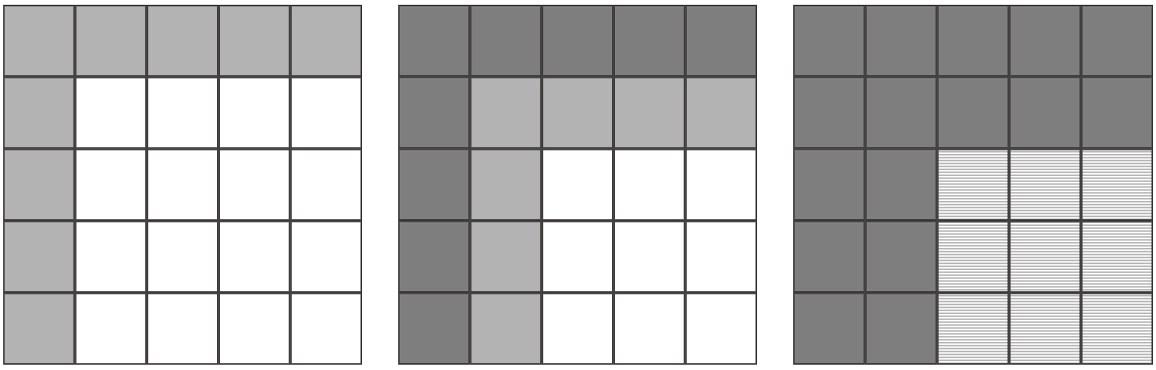
Figura 3.3: Última ficha (U), penúltima ficha (PU), fichas fijas (F).

Se va a dar el nombre de *adyacencia fuerte* al tipo de adyacencia entre fichas horizontal o vertical y *adyacencia débil* al tipo de adyacencia diagonal. Nótese que los movimientos de las fichas sólo pueden ser horizontales o verticales pero la adyacencia débil servirá para saber si alguna esquina de la celda vacía es adyacente a alguna esquina de la ficha destino u otra.

El algoritmo del juego de $n^2 - 1$ como se ha mencionado, se va a llevar a cabo de una manera similar al de Parberry, resolviendo de manera recursiva solamente la fila y columna n. Estando éstas acomodadas, se procede a acomodar la fila y columna $n - 1$ y se continúa de esta manera hasta que n sea igual a 3, momento desde el cual se utiliza fuerza bruta para llevar a cabo los movimientos siguientes. En la Figura 3.4(a), (b) y (c) se ilustra lo anterior.

Una ficha en estado *fijo* es aquella que se encuentra en su localidad destino.

La *última ficha* de una fila/columna es la ficha cuya localidad destino es n de dicha fila/columna y la *penúltima ficha* es aquella cuya localidad destino es $n - 1$. En el algoritmo, la ficha penúltima de cada fila/columna se va a llevar a la posición de la última y va a ser



(a) Primera fila y columna del juego de 24.

(b) Segunda fila y columna del juego de 24 (Juego de 15).

(c) Juego de 8. Se resuelve por fuerza bruta.

Figura 3.4: En (a) se están rellenando la fila y columna 5, en (b) se están rellenando la fila y columna 4 y la fila y columna 5 ya han sido calculadas, en (c) se está rellenando el tablero de 3×3 y y las filas y columnas 4 y 5 ya han sido calculadas.

marcada como fija para poder llevar a cabo una secuencia de movimientos específica para acomodar a ésta y a la última ficha de su respectiva fila/columna como se muestra en la Figura 3.3.

Se va a *rellenar* una fila/columna cuando se estén posicionando las fichas correspondientes a esa fila/columna en sus localidades destino.

3.1. El tablero de 2×2

La función del tablero de 2×2 fichas (al cual nos referiremos como *tabla* en el futuro) en este trabajo es darnos la posibilidad de saber en la mayoría de los casos, qué secuencia de movimientos es la óptima para acercar la ficha destino a la localidad destino cuando la celda vacía es adyacente a esta ficha (sea por adyacencia fuerte o débil).

Definimos como d la distancia desde la ficha destino a su localidad destino en cualquier momento. Para calcular esta distancia se utiliza la *distancia Manhattan* [6]. Llamemos a la coordenada de la localidad destino (i, j) y a la coordenada de la ficha destino (k, l). La distancia Manhattan se obtiene de la siguiente manera: $|i - k| + |j - l|$.

Podemos definir para mayor claridad, subtipos de las adyacencias fuerte y débil. Para la adyacencia fuerte existen cuatro subtipos: Norte (N), Sur (S), Este (E) y Oeste (O). Si la posición de la ficha es (i, j), la celda vacía se encuentra en la posición $(i-1, j)$ en el caso de adyacencia N, si la posición de la ficha es (i, j), la celda vacía se encuentra en la posición $(i, j+1)$ en el caso de adyacencia E y así sucesivamente.

De igual manera existen cuatro subtipos para la adyacencia débil: Noreste (NE), Noroeste (NO), Sureste (SE) y Suroeste (SO). Si la posición de la ficha es (i, j), la celda vacía se encuentra en la posición $(i-1, j+1)$ en el caso de adyacencia NE y se procede de manera similar para los demás subtipos.

Cuando en este trabajo se habla de que hay un "movimiento" de la celda vacía, significa que alguna ficha es trasladada a la posición actual de la celda vacía y la celda en la que se encontraba la ficha es la nueva posición de la celda vacía.

Se pueden utilizar las letras N, S, E y O para indicar que la celda vacía se va a mover hacia la posición $(i-1, j)$ en el caso de N, $(i+1, j)$ en el caso de S, $(i, j+1)$ en el caso de E y $(i, j-1)$ en el caso de O, estando ésta en la posición (i, j).

Se va a *superponer* la tabla al tablero cuando se haga coincidir la ficha uno de alguna configuración de la tabla con la ficha destino del tablero y la celda vacía de la misma configuración de la tabla con la celda vacía del tablero. Se puede ver la relación de una configuración con área 2×2 del tablero con una configuración de la tabla en la Figura 3.5. Lo anterior se hace siempre y cuando el área de la tabla no rebase los límites del tablero. En adelante a la acción de superponer alguna configuración de la tabla al tablero la llamaremos simplemente *superponer la tabla*.

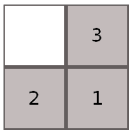


Figura 3.5: Configuración de la tabla con respecto a adyacencia de ficha destino y celda vacía de la Figura 3.6.

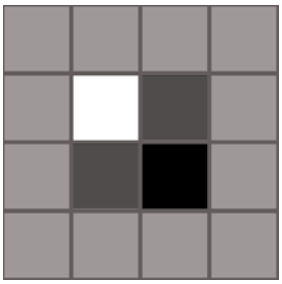


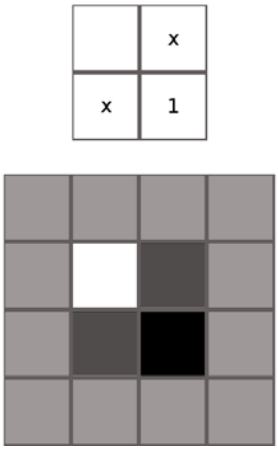
Figura 3.6: Tablero de 4×4 con la ficha destino en la posición (3,3) y la celda vacía en la posición (2,2).

En la tabla hay dos configuraciones distintas por cada subtipo de adyacencia fuerte y una configuración para cada subtipo de adyacencia débil (Cuadro 3.1). El conjunto de estas configuraciones es el total de configuraciones válidas en la tabla. Así que al superponer la tabla teniendo adyacencia fuerte, se puede utilizar cualquiera de estas dos configuraciones. Dado lo anterior, podemos ver que siempre es posible superponer la tabla.

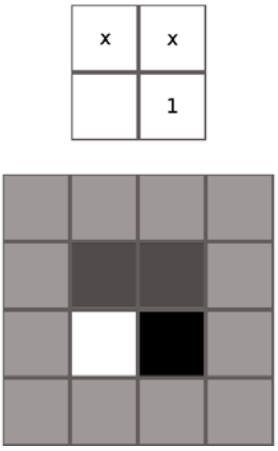
Habiendo superpuesto la tabla, un movimiento con respecto a ésta será trasponer alguna de las fichas en el área del tablero donde dicha tabla se ha superpuesto con la celda vacía de éste, imitando el movimiento que se tendría que hacer en la tabla para llegar a la configuración deseada dentro del área de 2×2 (a esta configuración la llamaremos simplemente *configuración deseada*). Cuando se hable de una configuración deseada, únicamente se dará importancia a la posición de la ficha uno en dicha configuración.

Una secuencia de movimientos con respecto a la tabla se lleva a cabo cuando la configuración deseada está a $\delta > 1$ de la configuración actual. Podemos ver un ejemplo de una serie de movimientos con respecto a la tabla en la Figura 3.7. En la Figura 3.7(a) la ficha destino se encuentra en la posición $(3, 3)$ del tablero. Llevando a cabo la secuencia de movimientos de las Figuras 3.7(b), 3.7(c), 3.7(d), 3.7(e) y 3.7(f), y suponiendo que la localidad destino se encuentra en la posición $(1, 1)$ del tablero, la distancia d se reduce de $d = 4$ a $d = 2$.

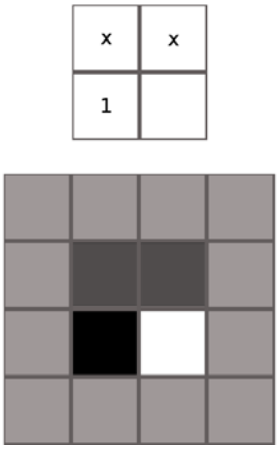
Lema 1 *Una secuencia de cinco movimientos con respecto a la tabla son suficientes y a veces necesarios para llegar a la configuración deseada.*



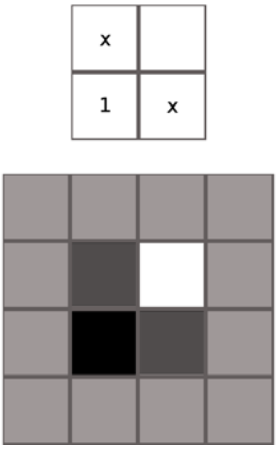
(a) Posición de ficha destino y celda vacía en el tablero y configuración de tabla a superponer.



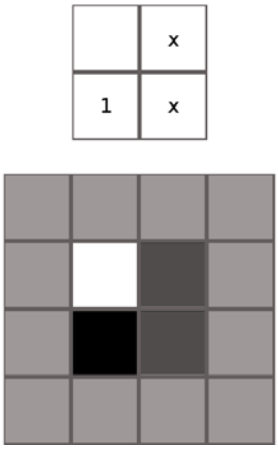
(b) Primer movimiento respecto a la tabla.



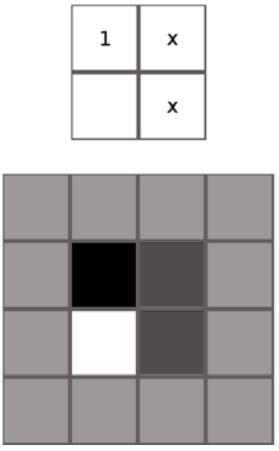
(c) Segundo movimiento respecto a la tabla.



(d) Tercer movimiento respecto a la tabla.



(e) Cuarto movimiento respecto a la tabla.



(f) Quinto movimiento respecto a la tabla.

Figura 3.7: Secuencia de cinco movimientos con respecto a la tabla.

Se puede ver en el Cuadro 3.1 que la ficha uno puede ocupar cualquiera de las cuatro posiciones de la tabla. Existen tres configuraciones diferentes que corresponden a cada una de estas posiciones. Al llevar a cabo una secuencia de movimientos, no es necesario conocer las posiciones de las fichas dos y tres, ya que solamente las posiciones de la ficha uno y la celda vacía son de interés.

Se ha mencionado previamente que δ es 6. Esto significa que llegar de cualquier configuración de la tabla a cualquier otra requiere de una secuencia de a lo más seis movimientos. Ya que nuestro interés reside en la posición final de la ficha 1, en cualquier caso se puede tomar cualquiera de las tres configuraciones antes mencionadas como configuración deseada y el hecho de que δ sea 6 significa que alguna de estas tres configuraciones se encuentra a lo más a cinco movimientos de la configuración actual.

□

A la configuración que se seleccione de las tres posibles mencionadas anteriormente se le llamará *configuración de menor distancia*.

Para llevar a cabo la secuencia más corta de movimientos de la tabla entre la configración actual y la configuración de menor distancia se insertan en una lista circular doblemente ligada cada una de las configuraciones de la tabla. Los valores anterior y siguiente en cada nodo serán las configuraciones que estén a un movimiento de distancia de éste. Se asigna a cada uno de los nodos un índice comenzando por 0 hasta el 11 (sin importar cuál sea el cero, el uno se asignará al nodo apuntado como siguiente del nodo cero y así sucesivamente). Teniendo estos datos, se procede de la siguiente manera:

```
1: si índice de configuración de menor distancia < índice de configuración actual entonces
2:    si abs(índice de menor distancia - índice de configuración actual) <= 6 entonces
3:       Mover a siguiente configuración.
4:    si no
5:       Mover a configuración anterior.
6:    fin si
7: si no
8:    si abs(índice de menor distancia - índice de configuración actual) <= 6 entonces
9:       Mover a configuración anterior.
10:   si no
11:      Mover a siguiente configuración.
12:   fin si
13: fin si
```

Lema 2 *Existen casos en los que superponer la tabla servirá para calcular la secuencia más corta de movimientos para llevar la ficha destino de una posición en el tablero a otra en un área de* 2×2 *tal que* d *sea reducida.*

Habiendo decidido qué configuraciones de la tabla se pueden superponer, se calcula con la distancia Manhattan qué posición de cualquiera de las configuraciones seleccionadas es la más cercana a la localidad destino. Un ejemplo de lo anterior se puede ver en la Figura 3.8. Si coincide la posición más cercana con la posición de la ficha destino, se utiliza algún otro método de los que se hablará más adelante. Si no, se lleva a cabo la secuencia de movimientos necesarios con respecto a la tabla para llevar la ficha destino a la posición más cercana a la localidad destino dentro del área de 2×2 en el tablero correspondiente a la configuración seleccionada de la tabla tal que d sea reducida (ver Figura 3.9) en a lo más cinco movimientos (Lema 1).

□

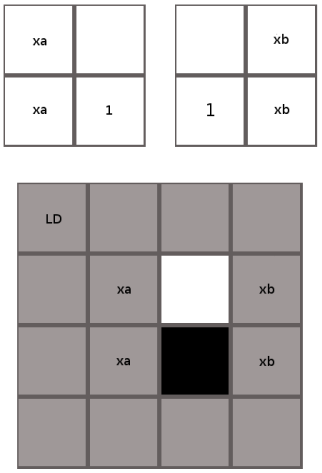
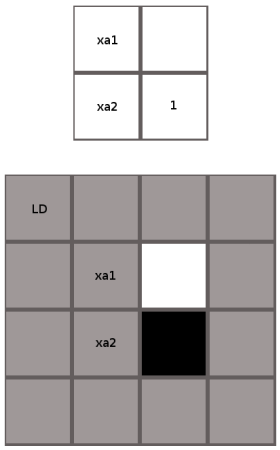
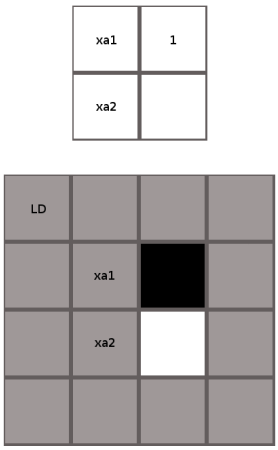


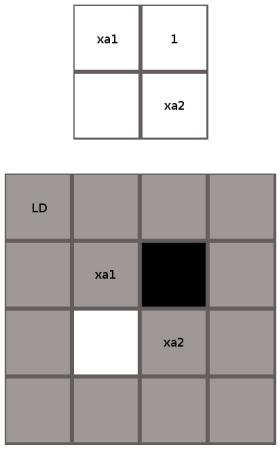
Figura 3.8: Dos configuraciones posibles. Si LD es la localidad destino, la configuración a será seleccionada.



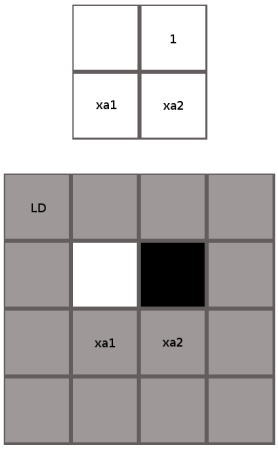
(a) La posición *xa1* es la más cercana a LD en el área de 2×2 sobre la que la tabla se ha sobrepuesto. $d = 4$



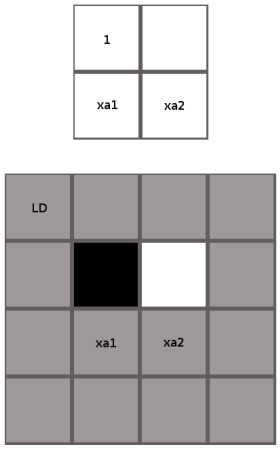
(b) Primer movimiento respecto a la tabla. $d = 3$



(c) Segundo movimiento respecto a la tabla.



(d) Tercer movimiento respecto a la tabla.



(e) Cuarto movimiento respecto a la tabla. $d = 2$

Figura 3.9: Secuencia de movimientos con respecto a la tabla tal que d se reduce de 4 a 2.

3.2. Jerarquía de movimientos

En el Lema 2 se menciona que existen casos en los que no se puede llevar a cabo una secuencia de movimientos con respecto a la tabla. En estos casos van a existir dos opciones, hacer una secuencia de movimientos preestablecida de acuerdo a una jerarquía que se describe a continuación, o de acuerdo a una serie de casos especiales que se describen en la sección siguiente.

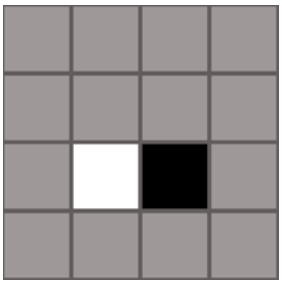
Vamos a definir dos tipos de movimientos que se van a utilizar en casos en los que se necesite la jerarquía. A favor de las manecillas del reloj (CW) o en contra de las manecillas del reloj (CCW). En los movimientos CW, la celda vacía se va a "mover" a favor de las manecillas del reloj *alrededor* de la ficha destino mientras que en los movimientos CCW, la celda vacía se va a "mover" en contra de las manecillas del reloj *alrededor* de la ficha destino. Se puede ver un ejemplo de lo anterior en la Figura 3.10.

Cuando la ficha destino se encuentra en la posición más cercana de la tabla superpuesta a la localidad destino, ya no es posible llevar a cabo movimientos con respecto a la tabla (Figura 3.11), entonces, salvo cuando existe un caso especial, será necesario cambiar la celda vacía de posición para poder seguir utilizando la tabla en los movimientos siguientes o en su defecto, volver a utilizar casos especiales o la jerarquía de movimientos.

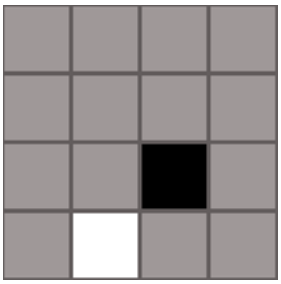
Existen diversas posibles posiciones de la localidad destino con respecto a la ficha destino y para cada una de ellas se va a ejecutar un tipo de movimiento de la jerarquía (ya sea CW o CCW).

Sea (i, j) la posición de la ficha destino y (k, l) la localidad destino, entonces si:

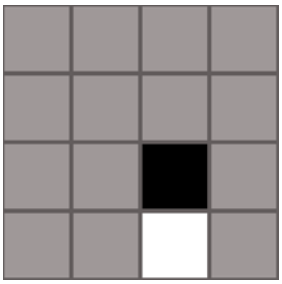
$k < i$ y $l < j$, tipo de movimiento: CCW. (arriba-izquierda)
$k < i$ y $l > j$, tipo de movimiento: CCW. (arriba-derecha)
$k > i$ y $l < j$, tipo de movimiento: CCW. (abajo-izquierda)
$k > i$ y $l > j$, caso especial. (abajo-derecha)



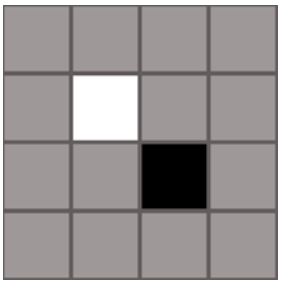
(a) Posición inicial de la celda vacía.



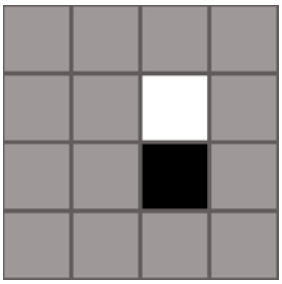
(b) Primer movimiento CCW.



(c) Segundo movimiento CCW.



(d) Primer movimiento CW.



(e) Segundo movimiento CW.

Figura 3.10: Ejemplo de movimientos CW y CCW.

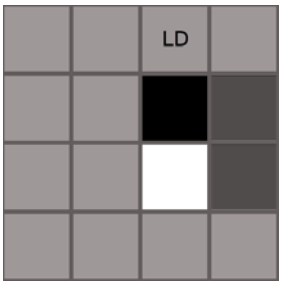


Figura 3.11: Ficha destino (pintada de negro) en la posición más cercana de la tabla superpuesta a localidad destino (LD).

$k < i$ y $l = j$, tipo de movimiento: CCW. (arriba)

$k = i$ y $l > j$, tipo de movimiento: CCW. (derecha)

$k = i$ y $l < j$, tipo de movimiento: CW. (izquierda)

$k > i$ y $l = j$, tipo de movimiento: CW. (abajo)

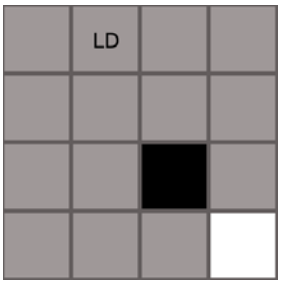
Cada uno de estos casos se ilustra en la Figura 3.12 (excepto *abajo-derecha* que se ilustra en los casos especiales).

Ya que en el algoritmo se revisa la posición de la celda vacía después de ejecutar cada movimiento para decidir qué movimiento será el siguiente, sólo será necesario evaluar los casos en los que la localidad destino se encuentra *arriba*, *abajo*, a la *izquierda* o a la *derecha* de la ficha destino, ya que por ejemplo, si la localidad destino se encontrara *arriba-izquierda* de la ficha destino, basta con llevar a cabo el movimiento de *arriba* y el estado del tablero se volverá a evaluar resultando en un caso completamente distinto.

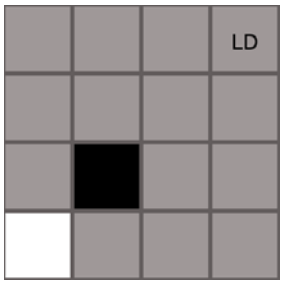
Cuando se está rellenando una fila, es normal que se suscite el caso *arriba* (nótese que cuando se rellena la fila n, la columna n aún no puede tener fichas fijas salvo la primera que es a su vez la primera de la fila n). Se ha seleccionado el tipo de movimiento CCW para este caso de manera arbitraria. Sólo va a existir un caso en el que se necesite hacer el movimiento CW con *arriba* y es cuando la localidad destino se encuentra en la columna n y será tratado como un caso especial. Se puede ver que si se hubiera seleccionado CW para el caso *arriba*, sería igual de eficiente y el caso especial ejecutaría el tipo CCW.

La misma situación se va a dar con el caso de *izquierda* sólo que esta vez CCW y CW están invertidos, es decir, en la situación normal se llevará a cabo CW y en el caso especial, CCW.

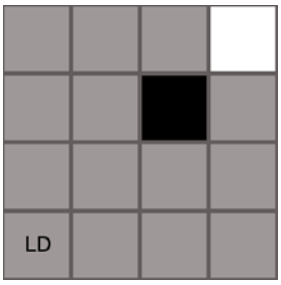
La selección del tipo de movimiento CCW para *derecha* no es arbitraria. Este caso se va a dar únicamente en la fila que se esté rellenando para llevar a la penúltima ficha a la última posición de la fila (en la siguiente sección se explica el por qué). La razón por la que el tipo de movimiento CCW es utilizado para este caso es que si se seleccionara CW, y la celda vacía tuviera adyacencia O con respecto a la ficha destino, se le estaría indicando a la celda



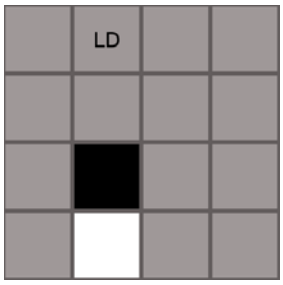
(a) Arriba-izquierda.



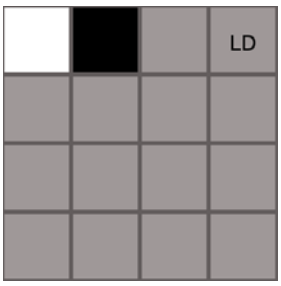
(b) Arriba-derecha.



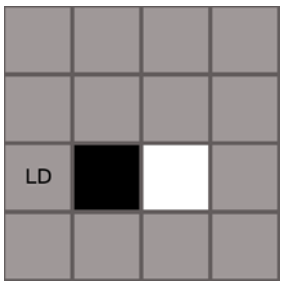
(c) Abajo-izquierda.



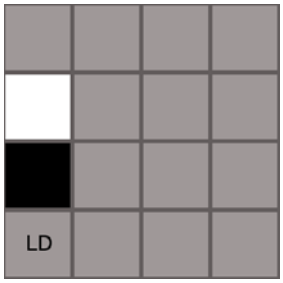
(d) Arriba.



(e) Derecha.



(f) Izquierda.



(g) Abajo.

Figura 3.12: Ejemplo de casos de jerarquía. 'LD' es la localidad destino, ficha destino de negro y celda vacía de blanco

vacía que se moviera fuera del tablero.

Existe un caso especial (*derecha-abajo*) en el que se lleva a cabo una secuencia de movimientos

específica.

El caso *abajo* es similar a *derecha*, sólo que en vez de tratar con filas, lo hace con columnas.

3.3. Última ficha

Llevar a la última ficha a su posición final es tratado de diferente manera a las demás fichas de la fila/columna a la que corresponde. Como se había mencionado, es necesario primero mover la penúltima ficha a la posición final de la última y a la última a una fila abajo de la penúltima en el caso de que se esté rellenando una fila o a la columna de la derecha cuando se esté rellenando una columna (Figura 3.3). Después de esto se debe llevar a cabo una serie de movimientos específicos para llevar a éstas fichas a sus respectivos destinos.

Si se está rellenando una fila:

Si la celda vacía tiene adyacencia O con respecto a la última ficha, moverla de la siguiente manera: N, E, S.
Si la celda vacía tiene adyacencia S con respecto a la última ficha, moverla de la siguiente manera: O, N, N, E, S.

Cuando se dan casos especiales para posicionar la última ficha, éstos llevan a cabo una serie de movimientos que deja la celda vacía con adyacencia NO con respecto a la última ficha. En estos casos basta con mover la celda vacía de la siguiente manera: E, S y la penúltima y última fichas quedarán en sus localidades finales.

Si se está rellenando una columna:

Si la celda vacía tiene adyacencia N con respecto a la última ficha, moverla de la siguiente manera: O, S, E.
Si la celda vacía tiene adyacencia E con respecto a la última ficha, moverla de la siguiente manera: N, O, O, S, E.

Y si como en el caso de las filas, se da un caso especial para posicionar la última ficha, la celda vacía quedará con adyacencia *NO* con respecto a la última ficha. Se debe mover la celda vacía de la siguiente manera: *S*, *E* y la penúltima y última fichas quedarán en sus localidades finales.

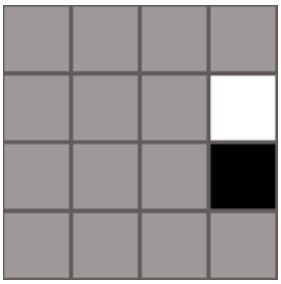
Con lo anterior se puede ver que se necesitan a lo más *cinco movimientos* para colocar las fichas penúltima y última en sus localidades destino suponiendo que la penúltima ficha se encuentra en la localidad final de la última ficha y que la última una fila abajo en caso de que se esté rellenando una fila o una columna a la derecha en caso de que se esté rellenando una columna y que la celda vacía es adyacente a la última ficha.

3.4. Casos especiales

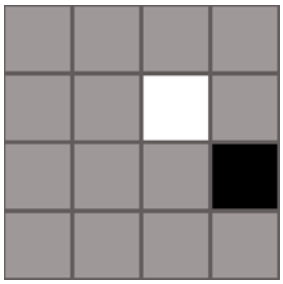
Un caso especial va a ocurrir cuando no se pueda utilizar la tabla o cuando no haya algún movimiento de la jerarquía que lo contemple.

Van a existir dos situaciones en las que va a ser necesario utilizar los casos especiales, cuando haya alguna o varias fichas fijas adyacentes a la ficha destino y no exista algún caso de jerarquía que contemple el movimiento que se necesita llevar a cabo o que la ficha destino se encuentre en el límite del tablero (en la posición 1 o n ya sea de fila o de columna) y ya sea que esté en la posición de la tabla superpuesta más cercana a la localidad destino o que no exista algún caso de jerarquía que contemple el movimiento.

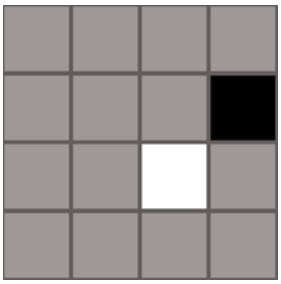
Las Figuras 3.13, 3.14, 3.15, 3.16 y 3.17 ilustran los casos especiales posibles y a continuación, la serie de movimientos para solucionar cada caso.



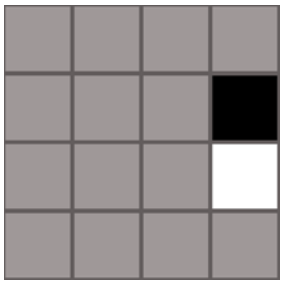
(a) Caso especial 1.



(b) Caso especial 2.

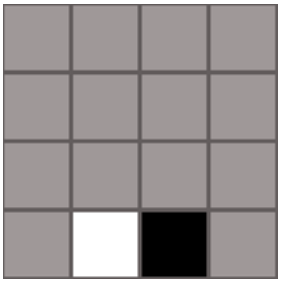


(c) Caso especial 3.

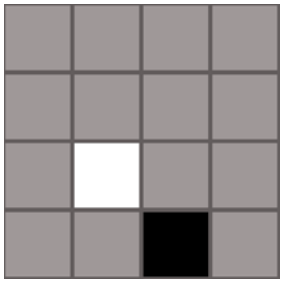


(d) Caso especial 4.

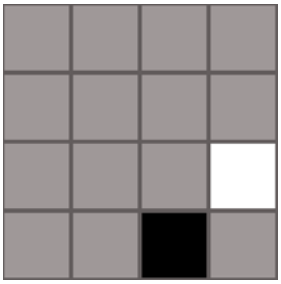
Figura 3.13: Casos especiales 1 a 4.



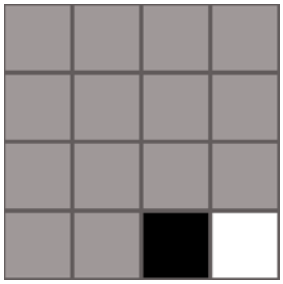
(a) Caso especial 5.



(b) Caso especial 6.

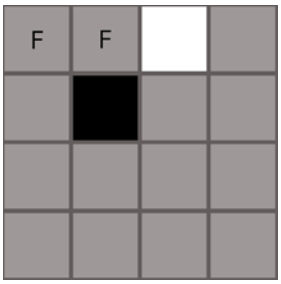


(c) Caso especial 7.

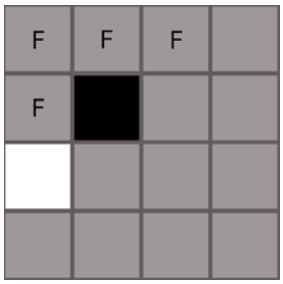


(d) Caso especial 8.

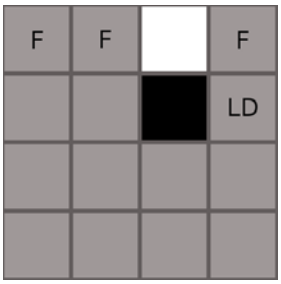
Figura 3.14: Casos especiales 5 a 8.



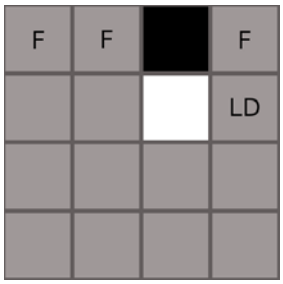
(a) Caso especial 9.



(b) Caso especial 10.

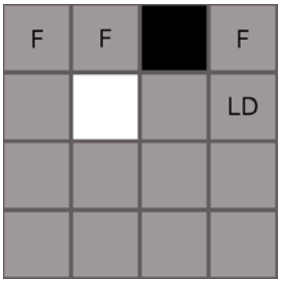


(c) Caso especial 11.

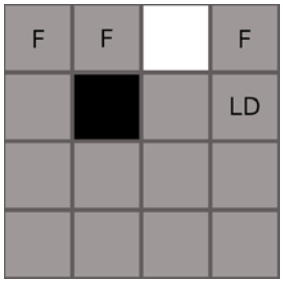


(d) Caso especial 12.

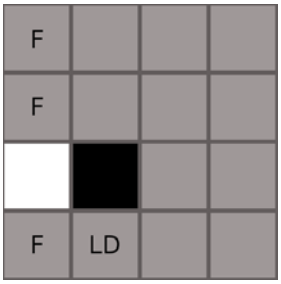
Figura 3.15: Casos especiales 9 a 12.



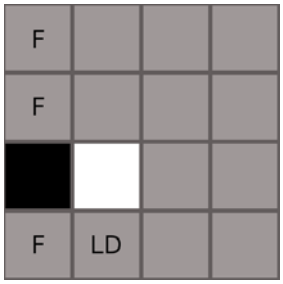
(a) Caso especial 13.



(b) Caso especial 14.

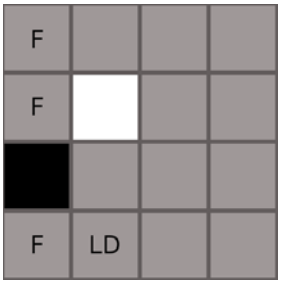


(c) Caso especial 15.

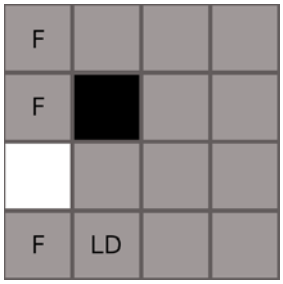


(d) Caso especial 16.

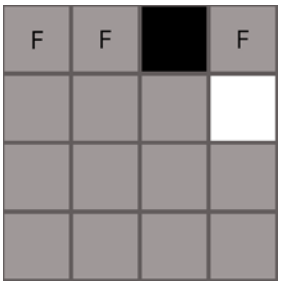
Figura 3.16: Casos especiales 13 a 16.



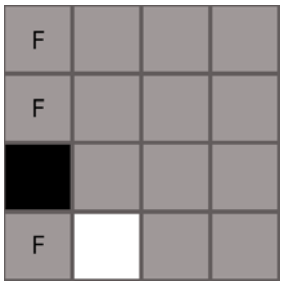
(a) Caso especial 17.



(b) Caso especial 18.



(c) Caso especial 19.



(d) Caso especial 20.

Figura 3.17: Casos especiales 17 a 20.

A continuación se describen los movimientos a llevar a cabo cuando se presenta un caso especial:

Para el caso 1: CCW

Para el caso 2: CCW

Para el caso 3: CW

Para el caso 4: CW

Para el caso 5: CW

Para el caso 6: CW

Para el caso 7: CCW

Para el caso 8: CCW

Para el caso 9: CW

Para el caso 10: CCW

Para el caso 11: O, S, S, E, E, N, O, O, N, E

Para el caso 12: N, O, S, S, E, E, N, O, O, N, E

Para el caso 13: E, N, O, S, S, E, E, N, O, O,N, E

Para el caso 14: S

Para el caso 15: N, E, E, S, S, O, N, N

Para el caso 16: O, N, E, E, S, S, O, N, N, O,S

Para el caso 17: S, O, N, E, E, S, S, O, N, N,O, S

Para el caso 18: E

Para el caso 19: O, N, O, S, S, E, E, N, O, O,N, E

Para el caso 20: N, O, N, E, E, S, S, O, N, N,O, S

3.5. Descripción de la estructura general del algoritmo

A continuación se enlista el algoritmo del juego de $n^2 - 1$ en su forma general.

En el siguiente pseudocódigo las localidades finales de las fichas última y penúltima no serán las reales sino las que se mencionan previamente para poder llevar a cabo la serie de movimientos que permiten moverlas a sus localidades destino reales.

$n^2 - 1$_Puzzle(n)
si ($n > 3$) **entonces**
  (Generar fila n)
  Calcular localidades destino de la última y penúltima fichas de la fila n
  repetir
    Pasar a la siguiente ficha destino
    Calcular localidad destino de ficha destino
    si la ficha destino no está en su localidad destino **entonces**
      Mover la celda vacía a adyacencia (débil o fuerte)
      repetir
        Sobreponer *tabla* a la ficha destino
        Verificar si la *tabla* sobrepuesta se encuentra sobre alguna ficha fija o si existe secuencia en *tabla* que acerque a la ficha destino a su localidad destino
        si se encuentra sobre alguna ficha fija o no existe secuencia en *tabla* **entonces**
          Buscar *caso especial* en el que exista una secuencia de movimientos que lleve la ficha destino más cerca a su localidad destino
          si existe dicho caso especial **entonces**
            Llevar a cabo serie de movimientos dictados por dicho caso
          si no
            Mover la celda vacía con respecto a *jerarquía*
          fin si
        si no
          Llevar a cabo la secuencia de movimientos de *tabla* que acerque la ficha a su localidad destino.
        fin si
      hasta que la ficha destino esté en su localidad destino
    fin si
  hasta que la última ficha sea marcada como fija
  (Generar columna n)
  (Se procede de la misma manera que para generar la fila n pero con las operaciones específicas para manejo de las columnas)
  $n^2 - 1$_Puzzle($n - 1$)
fin si

3.6. Peor caso

En el peor caso, cada vez que se comience a mover una ficha, ésta se encontrará a la máxima distancia Manhattan de su localidad destino y la celda vacía será adyacente a la última ficha que se posicionó (la ficha uno se encontrará en la posición (n, n) y la celda vacía en la posición $(1, 1)$). Un ejemplo de lo anterior se puede ver en la Figura 3.18.

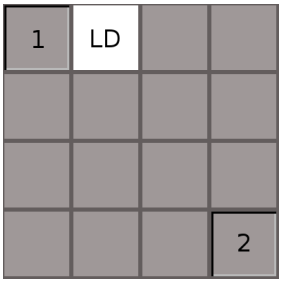
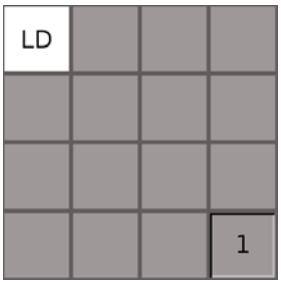


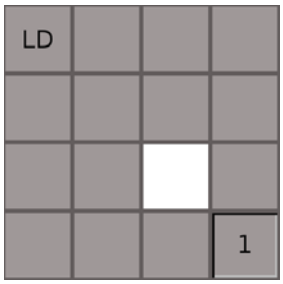
Figura 3.18: Peor caso ficha 2.

Teorema 2 *El procedimiento* $n^2 - 1$*_Puzzle(n) resuleve el juego juego* $n^2 - 1$ *en a lo más* $5n^3 - \frac{17n^2}{2} + \frac{31n}{2} - 71$ *movimientos.*

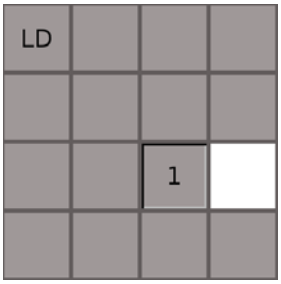
Suponga que la celda vacía se encuentra en la posición $(1, 1)$ y que la ficha 1 se encuentra en la posición (n, n) como se describe en el peor caso. El número de movimientos para que la celda vacía sea adyacente a la ficha 1 será $n-2$ hacia E y $n-2$ hacia S. Si denotamos la localidad destino de la ficha como k este número será de $2n-k-3$ movimientos. Para mover a la ficha 1 al lugar más cercano a la localidad destino dentro del área de la tabla superpuesta se necesitan cinco movimientos (Lema 1). Después de eso, la celda vacía tendrá adyacencia E con respecto a la ficha 1. A partir de ese momento la posición más cercana a la localidad destino al superponer la tabla se encontrará a tres movimientos. La celda vacía tendrá adyacencia S con respecto a la ficha uno y de nuevo la posición más cercana a la localidad destino se encontrará a tres movimientos. Se puede continuar de esta manera hasta que se la ficha 1 se encuentre en la localidad 1 y la celda vacía tenga adyacencia E con respecto a ésta (Figura 3.19).



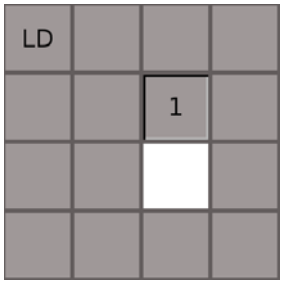
(a) Posición inicial de ficha uno y celda vacía.



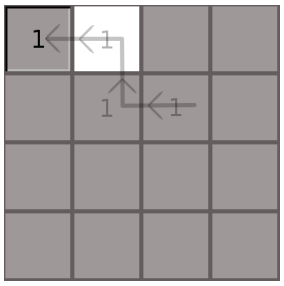
(b) Celda vacía adyacente a ficha uno.



(c) Ficha uno y celda vacía después de cinco movimientos de la tabla superpuesta.



(d) Ficha uno y celda vacía después de tres movimientos más.



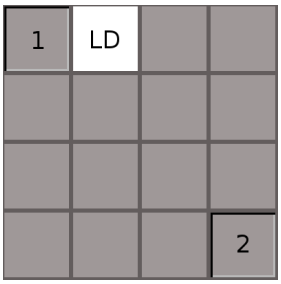
(e) Ruta que tomará la ficha uno para llegar a su localidad destino después de 3.19(d).

Figura 3.19: Movimiento de la ficha uno a su localidad destino en el peor de los casos.

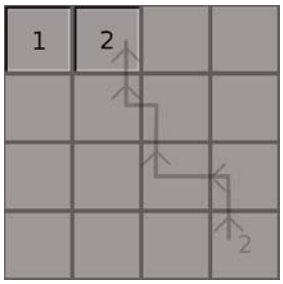
Después de los primeros cinco movimientos, por cada fila restante se harán 6 movimientos. Existen filas en las que no se ocupan esos seis movimientos. En este caso son las filas n y $n-1$. Para cada ficha en el rango $k = 1 \leq \frac{n}{2}$ habrán $k-1$ filas en las que no se ocupen los seis movimientos descritos anteriormente. Se necesitan entonces para llevar a la ficha uno a su localidad destino $5 + 6(n-k-1) + 2n - k - 3$ movimientos, o sea $8n - 11$ movimientos.

Para llevar a la ficha dos a la localidad $k = 2$ desde la localidad (n, n) se llevará a cabo la misma serie de movimientos que con la ficha uno y llegará a la posición con adyacencia S con respecto a la localidad $k = 2$, por lo que se necesitarán tres movimientos más para poder llevar a la ficha dos a su localidad destino (Figura 3.20). Se puede ver que el número de filas en las que la ficha dos utiliza la serie de seis movimientos descritos en el párrafo anterior es $(n - k - 1)$. Así que se puede llevar a la ficha dos a su localidad final en a lo más $5 + 6(n - k - 1) + 3 + 2n - k - 3$ movimientos. El caso de la ficha tres será similar a los anteriores; la serie de seis movimientos se llevará a cabo $(n - k - 1)$ veces y se necesitan tres movimientos más para llevarla a adyacencia S con respecto a su localidad destino y la celda vacía tendrá a su vez, adyacencia S con respecto a la ficha tres. Para llevar a esta ficha a su localidad destino es necesario mover la celda vacía hasta que tenga adyacencia N con respecto a ella, lo cual toma al menos cuatro movimientos y uno más para llevarla a su localidad destino. Podemos ver que la serie de seis movimientos que se lleva a cabo en $(n - k - 1)$ filas, se hace hasta que la ficha k queda $k - 1$ filas abajo de la localidad $(1, k)$ y la celda vacía con adyacencia E con respecto a ésta. Entonces se necesitan tres movimientos para que la ficha quede $k - 2$ filas abajo de la posición $(1, k)$ y la celda vacía con adyacencia S. A continuación se necesitan $5(k - 2)$ movimientos para llevar la ficha destino a su localidad destino, así que las fichas $3 \leq k \leq \frac{k}{2}$ serán llevadas a sus localidades destino en $2n - k - 3 + 5 + 6(n - k - 1) + 3 + 5(k - 2)$ o $8n - 2k - 11$ movimientos, donde $2n - k - 3$ es el número de movimientos necesarios para llevar la celda vacía a adyacencia con la ficha destino desde la columna k en la fila uno, 5 es el número de movimientos para llevar a la ficha destino a la posición $(n - 1, n - 1)$, $6(n - k - 1)$ movimientos son necesarios para llevar a la ficha a la columna k, 3 movimientos para subir una fila y $5(k - 2)$ movimientos para llegar a la localidad destino. En la Figura 3.21 se puede ver que la ficha cuatro ya ha sido llevada hasta el punto en el que se necesita mover la celda vacía alrededor de ella con movimientos CCW de manera repetida hasta que llegue a su localidad destino.

Así que para la primera mitad de la primera fila tenemos que el número de movimientos en el peor caso será de:



(a) Posición inicial de ficha dos y celda vacía.



(b) Ruta que tomará la ficha uno para llegar a su localidad destino.

Figura 3.20: Movimiento de la ficha dos a su localidad destino en el peor de los casos.

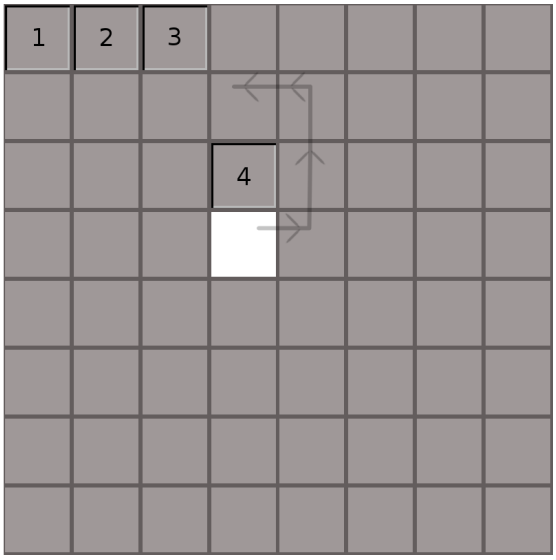


Figura 3.21: Las flechas indican la serie de cuatro movimientos que tiene que hacer la celda vacía para poder continuar llevando a la ficha cuatro a su localidad destino.

$$\begin{aligned}
\sum_{k=2}^{\frac{n}{2}}(8n-2k-11) &= 8\sum_{k=2}^{\frac{n}{2}}n-2\sum_{k=2}^{\frac{n}{2}}k-11\sum_{k=2}^{\frac{n}{2}}1 \\
&= 8n\left(\frac{n}{2}-1\right)-\left(\frac{n}{2}+2\right)\left(\frac{n}{2}-2+1\right)-11\left(\frac{n}{2}-1\right) \\
&= \frac{8n^2}{2}-8n-\left(\frac{n^2}{4}-n+\frac{n}{2}+n-4+2\right)-11\frac{n}{2}+11 \\
&= 15\frac{n^2}{4}+\frac{-28n}{2}+13.
\end{aligned}$$

Si añadimos a lo anterior los $8n - 11$ movimientos de la primera ficha:

$$\begin{aligned} 15\frac{n^2}{4} + \frac{-28n}{2} + 13 + 8n - 11 &= 15\frac{n^2}{4} + \frac{-28n + 16n}{2} + 2 \\ &= 15\frac{n^2}{4} + \frac{-12n}{2} + 2 \\ &= 15\frac{n^2}{4} + 6n + 2. \end{aligned}$$

Ahora, si reflejamos las operaciones anteriores para completar la segunda mitad de la fila, tomando en cuenta que para llevar a las fichas última y penúltima se necesitan cinco movimientos más tenemos:

$$2\left(15\frac{n^2}{4} - 6n + 2\right) + 5 = 15\frac{n^2}{2} - 12n + 9.$$

El cálculo de la primer columna es una reflexión de todas las operaciones anteriores pero sin contar a la primera ficha, lo que resulta en:

$$\begin{aligned} 2\left(15\frac{n^2}{2} - 12n + 9\right) - 8n + 11 & \\ &= 15n^2 - 24n + 18 - 8n + 11 \\ &= 15n^2 - 32n + 29. \end{aligned}$$

Este número de movimientos es necesario para resolver las filas/columnas $4 \leq k \leq n$ ya que como se ha mencionado, cuando $n = 3$ se utilizará fuerza bruta. Así que en total, para resolver el juego de $n^2 - 1$ tenemos:

$$\sum_{k=4}^{n}(15k^2-32k+29) \;=\; 15\sum_{k=4}^{n}k^2-32\sum_{k=4}^{n}k+29\sum_{k=4}^{n}1.$$

Resolviendo cada sumatoria por separado:

$$\begin{aligned}
15\sum_{k=4}^{n}k^2 &= 15\sum_{k=1}^{n-3}(k+3)^2\\
&= 15\sum_{k=1}^{n-3}k^2+15(6)\sum_{k=1}^{n-3}k+15(9)\sum_{k=1}^{n-3}1\\
&= \frac{15(n-3)(n-3+1)(2(n-3)+1)}{6}+\frac{15(6)(n-3)(n-3+1)}{2}+15(9)(n-3)\\
&= \frac{15(2n^3-15n^2+37n-30)}{6}+\frac{15(6)n^2-15(6)(5)n+15(6)(6)}{2}+15(9)n-15(9)(3)\\
&= \frac{30n^3-225n^2+555n-450+270n^2-1350n+1620+810n-2430}{6}\\
&= \frac{30n^3+45n^2+15n-1260}{6}\\
&= 5n^3+\frac{45}{6}n^2+\frac{15}{6}n-210.
\end{aligned}$$

La segunda sumatoria:

$$\begin{aligned}
32\sum_{k=4}^{n}k &= 32\frac{(n-4+1)(n+4)}{2}\\
&= 32\frac{n^2+n-12}{2}\\
&= 16(n^2+n-12)\\
&= 16n^2+16n-192.
\end{aligned}$$

La tercer sumatoria:

$$\begin{aligned} 29\sum_{k=4}^{n} 1 &= 29(n-3) \\ &= 29n - 87. \end{aligned}$$

Sumando el resultado de las tres sumatorias tenemos:

$$\begin{aligned} & 5n^3 + \frac{45}{6}n^2 + \frac{15}{6}n - 210 + 16n^2 + 16n - 192 + 29n - 87 \\ = \; & \frac{30n^3 + 45n^2 + 15n - 1260 - 96n^2 - 96n + 1152 + 174n - 522}{6} \\ = \; & \frac{30n^3 - 51n^2 + 93n - 630}{6} \\ = \; & 5n^3 - \frac{17n^2}{2} + \frac{31n}{2} - 105. \end{aligned}$$

Como se ha mencionado, en [7] se hace un estudio exhaustivo del juego de 3×3 y se concluye que el peor caso para este tablero es de 30 movimientos. Ya que $n^2 - 1$_Puzzle(3) se resuelve por fuerza bruta, al resultado anterior se le sumarán 34 movimientos y se obtiene:

$$5n^3 - \frac{17n^2}{2} + \frac{31n}{2} - 105 + 34 = \boxed{5n^3 - \frac{17n^2}{2} + \frac{31n}{2} - 71.}$$

□

Este resultado mejora aquel obtenido por Parberry en [6] primordialmente porque en su método es necesario llevar la celda vacía a adyacencia fuerte con la ficha destino mientras que en el método presentado en este trabajo, la ficha destino puede comenzarse a mover

teniendo adyacencia fuerte o débil con la celda vacía. El número de movimientos para el peor caso en el método propuesto por Parberry es de $5n^3 - \frac{9n^2}{2} + \frac{19n}{2} - 89$.

Capítulo 4

El problema de la bodega

Supongamos que se tiene un tablero con $n \times n$ celdas. Algunas de estas celdas contienen fichas y otras están vacías a diferencia del caso que se manejaba en el capítulo 3, en donde existía una sola celda vacía. Es lógico pensar que si se tienen varias celdas vacías, puede existir alguna manera más eficiente de ordenar las fichas que la descrita en el capítulo anterior. En este capítulo se propone un acercamiento basado en bloques y corredores.

Sea T un tablero de $n \times n$, un bloque dentro de T será un conjunto de fichas que formen un polígono ortogonal simple, cuyo perímetro será la frontera entre dicho bloque y los corredores que lo rodean. Un corredor dentro de T será un conjunto de celdas vacías acomodadas de manera horizontal o vertical entre los bloques. Un ejemplo de lo anterior se puede ver en la Figura 4.1.

Se va a estudiar el caso específico en el que los bloques contengan $\sqrt{n} \times \sqrt{n}$ fichas y cada uno de ellos esté rodeado de corredores. El haber tomado $\sqrt{n} \times \sqrt{n}$ como el tamaño de los bloques tiene como meta que éstos no sean ni muy grandes ni muy pequeños, asegurar que todos estén rodeados por corredores (si n tiene raíz entera) y que cuando n sea grande, la proporción de fichas con respecto a localidades del tablero sea alta.

El total de bloques para un tablero de $n \times n$ será $(\sqrt{n}-1)^2$. El total de fichas será $n^2 -$

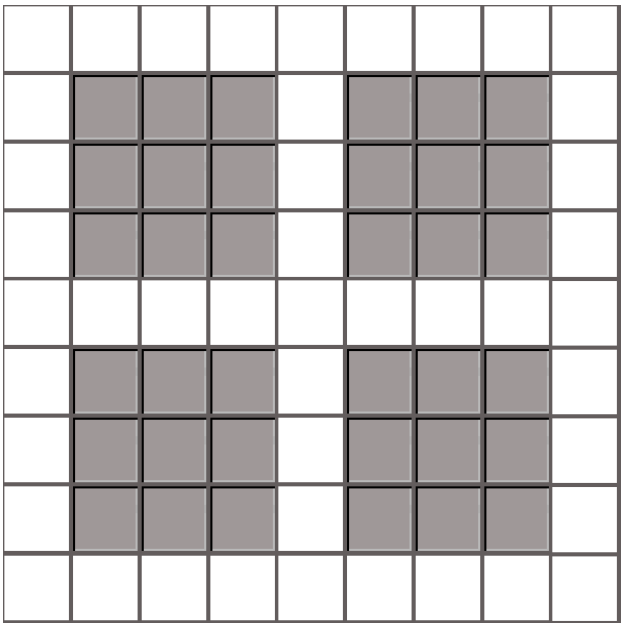


Figura 4.1: Bloques y corredores en un tablero de 9×9.

$2n\sqrt{n} + n$ y $2n\sqrt{n} - n$ celdas vacías.

Como se menciona en el capítulo 2, la motivación para el análisis de este problema es encontrar una forma eficiente para hacer permutaciones entre los elementos contenidos en estantes en una bodega con una configuración como la que se presenta en los siguientes párrafos.

Otra razón para estudiar este problema es comparar el número de movimientos que se deben llevar a cabo cuando se tienen varias celdas vacías y cuando se tiene sólo una.

Cuando en las siguientes secciones se usen los términos bodega, estantes o elementos contenidos en los estantes, entiéndase que la bodega tendrá un área de $n \times n$ celdas de tamaño unitario y cada elemento será del área de una celda.

4.1. Estructura general del algoritmo

Se tiene un tablero con $n \times n$ y $n^2 - 2n\sqrt{n} + n$ fichas dentro de éste acomodadas en $n - 2\sqrt{n} + 1$ bloques, con $2n\sqrt{n} - n$ celdas vacías organizadas en corredores. Entonces se debe llevar cada ficha a su bloque correspondiente y dentro de éste, a su localidad destino.

Un bloque *fuente* será aquel en el que se encuentra la *ficha destino* en determinado momento y el bloque *objetivo* será al que se tiene que mover esa ficha.

El algoritmo basado en bloques y corredores consta de los siguientes pasos generales:

1: Sacar la primer ficha destino de su bloque fuente mediante una serie predefinida de movimientos (puede ser cualquier ficha que no se encuentre en su bloque objetivo).
2: Trasladar esta ficha a adyacencia con su bloque objetivo.
3: Insertarla en su bloque objetivo utilizando movimientos previamente definidos.
4: Sacar del bloque la ficha que se encontraba en la localidad destino de la ficha destino, llevando a cabo los movimientos inversos del paso anterior.
5: Repetir los pasos 2, 3 y 4 hasta que todas las fichas se encuentren en su localidad destino.

Llevando a cabo estos pasos, será posible hacer una permutación de los elementos contenidos en estantes de una bodega con las características previamente mencionadas. Nótese que al finalizar estos pasos sobre un tablero, las fichas se encontrarán no sólo dentro de sus bloques objetivo sino en sus localidades destino. En la siguientes secciones se desarrollan los pasos del algoritmo.

4.2. Trasladar fichas entre bloques

Llamaremos *celda de menor distancia* de una ficha a la celda que se encuentre a menor distancia de la ficha y sobre un corredor. El término *ficha destino* se seguirá utilizando como se ha hecho durante todo el trabajo y la *siguiente ficha destino* será la ficha que se comenzará a mover a su bloque objetivo después de haber concluido los movimientos de la ficha destino actual.

Cuando llega el momento de trasladar cada ficha a su bloque objetivo, éstas se deben mover en primer lugar hasta el corredor más cercano y de ahí trasladarse por los corredores sin encontrarse obstáculo alguno (otras fichas). Cuando la ficha sea adyacente al bloque objetivo, se trasladará a la *celda de menor distancia* de la siguiente ficha destino. Mediante una serie de movimientos que se describen en la siguiente subsección, se introducirá la ficha destino al bloque y se sacará de éste la siguiente ficha destino. Este proceso se repite hasta que cada ficha se encuentre en su localidad destino. Un ejemplo de lo anterior se puede ver en la Figura 4.2.

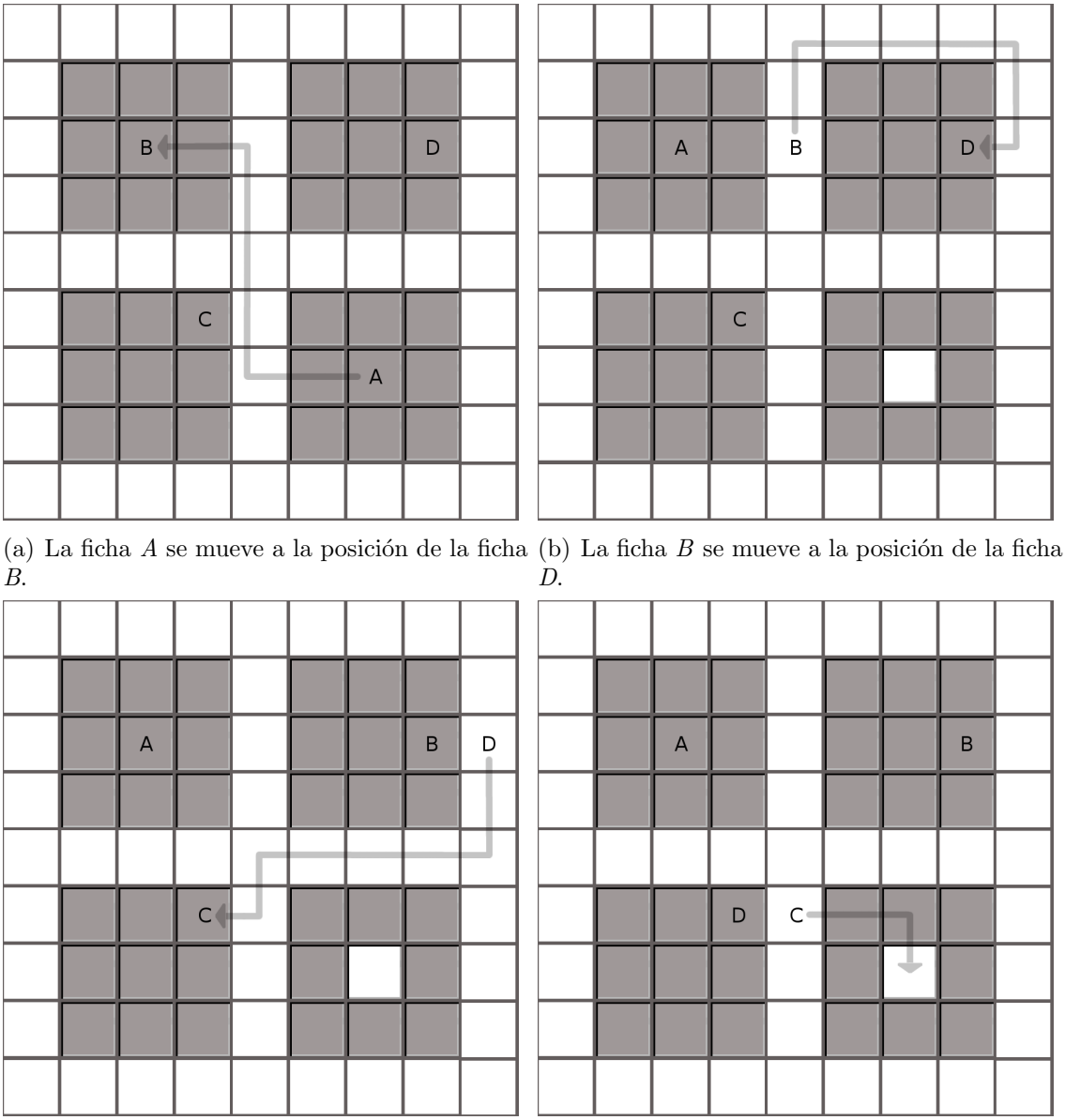
En la siguiente subsección se describe el método para sacar fichas de bloques fuente e introducirlas en bloques objetivo. Se analiza también el número de movimientos necesarios para llevar a cabo estas acciones.

4.2.1. Sacar fichas de bloques fuente e introducirlas en bloques objetivo

Llamaremos un *nivel* dentro de un bloque al conjunto de fichas que se encuentra a determinada distancia de su corredor más cercano. Así, el nivel 1 será el conjunto de fichas a distancia 1 del corredor, el nivel 2 será el conjunto de fichas a distancia 2 y así sucesivamente. Una *capa* se numerará exactamente del modo contrario a los niveles, o sea que la capa 1 será la capa del centro, la 2 será la siguiente y así sucesivamente hasta llegar al conjunto de fichas que hacen frontera con los corredores.

Si un bloque contiene n fichas ($\sqrt{n} \times \sqrt{n}$) y $\sqrt{n}$ es par, entonces el bloque está compuesto por $\frac{\sqrt{n}}{2}$ niveles; si $\sqrt{n}$ es impar, el bloque está compuesto por $\frac{\sqrt{n}+1}{2}$ niveles. Esto se debe a que el número de niveles es igual a la distancia máxima de cualquier ficha dentro del bloque a algún corredor. Esta distancia es la misma que la del "centro" del bloque a algún corredor.

Al introducir una ficha en el bloque y llevarla a su localidad destino, varias fichas tendrán que ser movidas en el proceso, así que al sacar la siguiente ficha, se deben regresar las fichas que



(a) La ficha A se mueve a la posición de la ficha B.

(b) La ficha B se mueve a la posición de la ficha D.

(c) La ficha D se mueve a la posición de la ficha C.

(d) La ficha C se mueve a la posición original de la ficha A.

Figura 4.2: Ejemplo de movimiento de fichas entre bloques.

se movieron a sus posiciones previas a insertar la ficha destino. El método que se presenta a continuación cuenta con esta propiedad. Así que al finalizar de trasladar todas las fichas, éstas estarán en sus posiciones finales.

Es fácil ver que en un tablero de 2×2 con dos celdas vacías y dos fichas, se pueden llevar las dos fichas de cualquier posición a cualquier otra; esto no es cierto cuando se tiene una

sola celda vacía como se explica en la sección 2.2. Utilizaremos este hecho para introducir y sacar fichas de los bloques sin cambiar su estado (salvo por la ficha introducida y la que se saca).

Para introducir una ficha a su bloque objetivo, lo primero que se debe hacer es introducir tres celdas vacías al bloque. Esto se hace para poder formar un tablero de 2×2 dentro del bloque con dos celdas vacías, la restante se utilizará para introducir la ficha destino al tablero. Para introducir estas celdas al tablero, es necesario sacar temporalmente tres fichas. Etiquetaremos estas fichas como a, b y c. La ficha que es adyacente a la *celda de menor distancia* de la siguiente ficha destino será la a, una ficha en el primer nivel adyacente a a, será b y otra más adyacente a b pero en el segundo nivel, será c (Figura 4.3). Estas tres fichas serán movidas al corredor adyacente a ellas de manera que no obstruyan la entrada de la ficha destino al bloque en 8 movimientos. Las celdas correspondientes a estas tres fichas antes de ser sacadas del bloque serán las celdas A, B y C, respectivamente.

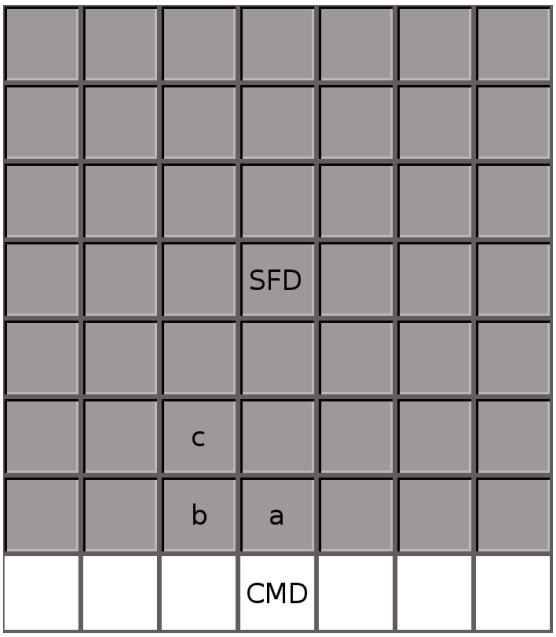
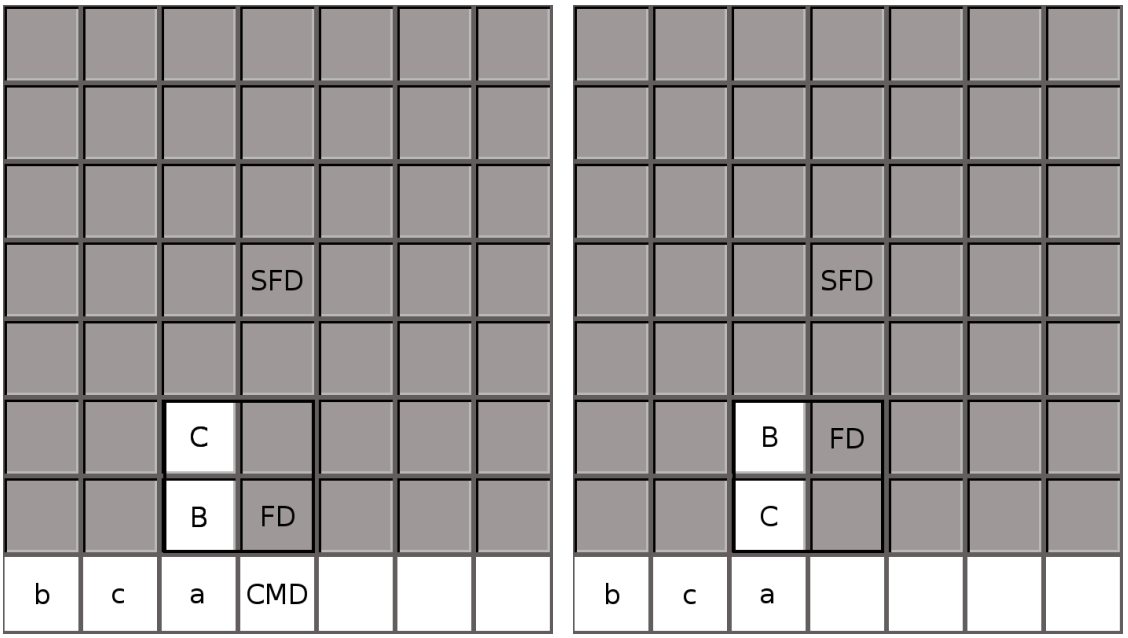


Figura 4.3: Bloque de 7×7. *SFD* es la siguiente ficha destino, *CMD* es la celda de menor distancia de la siguiente ficha destino. a, b y c representan las fichas a ser sacadas del bloque

El siguiente paso será llevar a la ficha destino a la celda A en un movimiento. De esta manera, se habrá formado un tablero de 2×2 dentro del bloque con las celdas B y C, la ficha destino

y la ficha del nivel 2 adyacente a la ficha destino. La Figura 4.4(a) ilustra la configuración del bloque hasta este momento.



(a) *FD* es la ficha destino introducida en el bloque. *a*, *b* y *c* han sido sacadas del bloque y *C* y *B* son las celdas vacías que quedan dentro del bloque. El cuadro negro representa el tablero de 2×2 formado.

(b) Configuración actual del bloque.

Figura 4.4: Configuración del bloque después de introducir la ficha destino y subirla un nivel.

En adelante cuando hablemos de la *ficha del siguiente nivel*, nos estaremos refiriendo a la ficha del siguiente nivel adyacente a la ficha destino.

En este momento podemos intercambiar la ficha destino con la ficha del siguiente nivel (si ésta fuera la posición final de la ficha destino, los pasos para introducirla estarían completos). Intercambiar estas dos fichas toma 4 movimientos y las celdas B y C también quedarán intercambiadas. Podemos ver en la Figura 4.4(b) que ahora es necesario subir un nivel las dos celdas vacías para que se vuelva a formar un talero de 2×2 con la ficha del siguiente nivel. Este procedimiento se repetirá hasta que la ficha destino se encuentre en su localidad destino. Por cada nivel desde el 3, se necesitan 6 movimientos para llevar la ficha destino a la posición de la ficha del siguiente nivel; 2 movimientos para subir las celdas vacías y cuatro más para intercambiar la ficha destino y la ficha del siguiente nivel.

Con lo anterior podemos ver que se necesitan 13 movimientos para meter una ficha al nivel 2 y 6 movimientos por cada nivel hasta llegar a la localidad destino de la ficha, así que para llevar una ficha al nivel r, son necesarios:

$$6(r-2)+13 \;\; = \;\; 6r+1$$

movimientos.

El procedimiento para sacar del tablero la siguiente ficha destino es llevar a cabo los movimientos para introducir al bloque la ficha anterior, pero de manera inversa. El orden de los movimientos será invertido también. Para ejemplificar, supongamos que tenemos tres fichas, X, Y y Z y que llevan a cabo los siguientes movimientos en orden, X hace S, S, Y: O, N, Z: S y Y: E, entonces, el inverso de los movimientos anteriores será: Y: O, Z: N, Y: S, E y X: N, N. Entonces para sacar la siguiente ficha destino, cada ficha movida en el paso anterior hará los movimientos que ejecutó pero en orden inverso. Se puede ver que al ejecutar movimientos, la siguiente ficha destino irá siendo trasladada hasta el nivel uno. Cuando llegue a este nivel, se debe sacar del tablero con un movimiento y llevar a cabo los movimientos inversos de las fichas que se sacaron al principio (las que permitieron que se insertaran las celdas vacías al bloque).

Ya que sacar una ficha del nivel r se lleva a cabo en el mismo número de movimientos que los necesarios para introducir la anterior (menos cuatro movimientos del intercambio entre estas dos), tenemos que se necesitan $6r-3$ movimientos para esta operación.

Para introducir una ficha al nivel uno, lo único que se tiene que hacer es sacar la siguiente ficha destino al corredor (lo que toma un movimiento) y moverla una posición sobre el corredor de tal manera que no obstaculice la inserción de la ficha destino (dos movimientos en total). Insertar una ficha en el primer nivel, habiendo sacado la siguiente ficha destino toma únicamente un movimiento.

4.2.2. Promedio del número de movimientos necesario para sacar e introducir fichas en un bloque

En esta sección se presenta el cálculo del promedio del número de movimientos necesario para sacar e introducir las fichas de un bloque cuando $\sqrt{n}$ es impar; el caso par será muy similar. Esto será útil para calcular el número de movimientos total que se lleva a cabo en el peor caso.

Para poder obtener el promedio, se necesita saber cuántas fichas existen en cada *capa*. Salvo en la capa central (la capa 1), cada capa contendrá $(2i-1)^2-(2i-3)^2$ fichas, donde i es el número de capa. La capa uno contendrá una sola ficha.

El número de movimientos necesarios para introducir una ficha a la capa uno será $6r+1$ y para sacar una ficha de la capa uno se necesitan $6r-3$ movimientos, donde r será el número de nivel de la capa uno. Ya que un bloque de $\sqrt{n}\times\sqrt{n}$ tiene $\frac{\sqrt{n}+1}{2}$ niveles, el número total de movimientos para la capa uno será $6\sqrt{n}+4$.

Recordemos que para sacar e introducir fichas en la misma posición de la última capa (nivel 1) se necesitan 3 movimientos.

Ya que la última capa es igual al número de niveles, ésta estará compuesta por:

$$\left(2\frac{\sqrt{n}+1}{2}-1\right)^2-\left(2\frac{\sqrt{n}+1}{2}-3\right)^2 \;=\; 4\sqrt{n}-4.$$

fichas.

Entonces se necesitan $12\sqrt{n}-12$ movimientos para sacar e introducir fichas a esta capa.

Para introducir y sacar fichas en cualquier otra capa se necesitan $12r-2$ movimientos, donde r será el número de capa.

Así, el promedio de movimientos para sacar e introducir las n fichas del bloque será:

$$\frac{\left(\sum_{i=2}^{\frac{\sqrt{n}+1}{2}-1}(12i-2)\left((2(\frac{\sqrt{n}+1}{2}+1-i)-1)^2-(2(\frac{\sqrt{n}+1}{2}+1-i)-3)^2\right)\right)}{n} + \frac{6\sqrt{n}+4+(12\sqrt{n}-12)}{n}$$

$$\begin{aligned}
&= \frac{\left(\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}(12i+10)\left((2(\frac{\sqrt{n}+1}{2}-i)-1)^2-(2(\frac{\sqrt{n}+1}{2}-i)-3)^2\right)\right)}{n} \\
&+ \frac{18\sqrt{n}-8}{n} \\
&= \frac{\left(\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}(12i+10)\left(4\sqrt{n}-8i-4\right)\right)+18\sqrt{n}-8}{n} \\
&= \frac{\left(\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}-96i^2+48\sqrt{n}i-128i+40\sqrt{n}-40\right)+18\sqrt{n}-8}{n} \\
&= \frac{8\left(-12\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}i^2+6\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}\sqrt{n}i-16\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}i+5\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}\sqrt{n}-5\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}1\right)}{n} \\
&+ \frac{18\sqrt{n}-8}{n}.
\end{aligned}$$

Resolviendo cada sumatoria:

$$-12\sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2}i^2 = \frac{-n\sqrt{n}+6n-11\sqrt{n}+6}{2}.$$

$$6 \sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2} \sqrt{n}i \;=\; \frac{3n\sqrt{n}-12n+9\sqrt{n}}{4}.$$

$$16 \sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2} i \;=\; 2n-8\sqrt{n}+6.$$

$$5 \sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2} \sqrt{n} \;=\; \frac{5n-15\sqrt{n}}{2}.$$

$$5 \sum_{i=1}^{\frac{\sqrt{n}+1}{2}-2} 1 \;=\; \frac{5\sqrt{n}-15}{2}.$$

Entonces, el número de movimientos para las n fichas del bloque será:

$$2n\sqrt{n}+4n-42\sqrt{n}+36.$$

Entonces, el promedio del número de movimientos necesario para sacar e introducir las n fichas de un bloque es:

$$\frac{2n\sqrt{n}+4n-24\sqrt{n}+28}{n} \;\approx\; \boxed{2\sqrt{n}.}$$

4.3. Distancia promedio en el peor caso

En esta sección se analiza la distancia (Manhattan) promedio que tiene que recorrer cada ficha desde el bloque fuente al bloque objetivo en el peor caso.

El peor caso en distancia se dará cuando todas las fichas, ya estando colocadas en bloques (ninguna en su bloque objetivo), se encuentren a la máxima distancia posible de su posición dentro del bloque objetivo. Se va a analizar el caso en el que el número de bloques es par. El caso impar es muy similar.

Es fácil ver que en el peor caso, las posiciones más cercanas a las esquinas van a ser las que estén a mayor distancia, ya que la distancia más grande en el tablero será de una esquina cualquiera a la opuesta. Así que habrán ocho posiciones a distancia $2n - 4$. Estas son las posiciones con adyacencia fuerte con respecto a las cuatro esquinas. Éstas pertenecen a los corredores adyacentes a los bloques más cercanos a las esquinas como se muestra en la Figura 4.5.

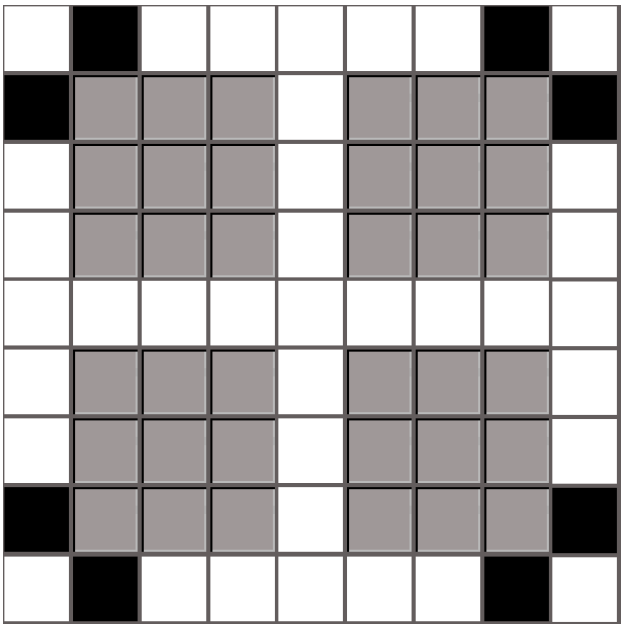


Figura 4.5: Las celdas pintadas de negro representan aquellas que se encuentran a mayor distancia.

Si se recorren las posiciones adyacentes a los bloques "de afuera hacia adentro", se puede

ver cómo la distancia máxima va disminuyendo hasta que la distancia máxima es dos (ya que las posiciones que se van visitando no se vuelven a tomar en cuenta). En la Figura 4.6 se representan los bloques y los corredores y se señalan varias distancias para ejemplificar lo explicado anteriormente. Nótese que entre más cerca del centro están las posiciones, menor es la distancia máxima.

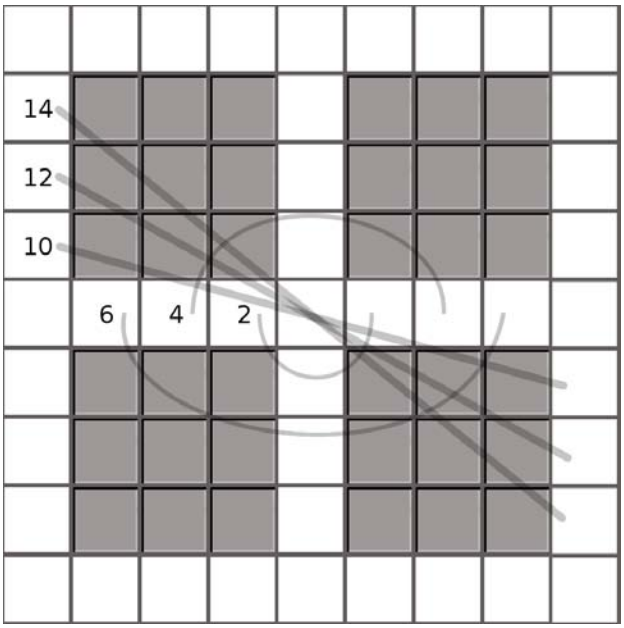


Figura 4.6: Las cantidades representan la distancia máxima de las celdas y las líneas representan las celdas hacia las que se mide la distancia.

Otro detalle a notar es cómo se comportan las distancias en las posiciones adyacentes entre sí y a un bloque, o sea, las posiciones a lo largo de uno de los *lados* de un bloque, como se muestra en la Figura 4.6. Estas distancias van disminuyendo en 2 conforme se van acercando al centro del tablero. La suma de estos valores será:

$$suma = \frac{(p+u)(\sqrt{n})}{2}$$

donde p es la máxima distancia en la primera posición del *lado* del bloque, u es la máxima

distancia de la última posición del *lado* del bloque.

Por lo anterior, van a existir en el tablero varios lados con la misma suma de distancias y aparte de esto, el tablero será simétrico horizontal y verticalmente si se toma la suma de distancias por lado como medida de simetría. Gracias a esto, podemos en adelante analizar tan sólo un cuarto del tablero y el resto tendrá las mismas propiedades.

En la Figura 4.7 los lados del mismo color tendrán la misma suma. Cada uno de los bloques, a su vez, presentará dos lados *l1* y *l2* del mismo color y otros dos lados *l3* y *l4* de un color distinto a *l1* y *l2* pero igual entre ellos. La distancia más grande de *l3* y *l4* será la distancia más pequeña de *l1* y *l2* menos cuatro como se puede ver en la Figura 4.6. Un bloque *vecino* de un bloque b, será aquel que se encuentre horizontal o verticalmente separado del bloque b por un corredor. El lado de un bloque vecino a b que comparte corredor con éste estará representado con el mismo color que dicho lado de b ya que las distancias se miden entre posiciones de corredores y no entre bloques.

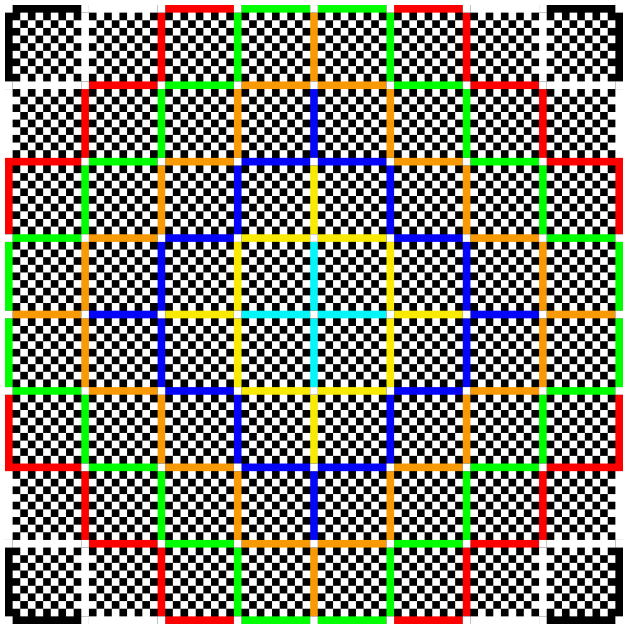


Figura 4.7: Las áreas con el patrón ajedrezado representan cada bloque en un tablero de 81×81 fichas. Cada color representa una suma de distancias entre conjuntos de celdas.

Un cuarto del tablero contendrá $k \times k$ bloques, donde $k = \sqrt{\frac{n-2\sqrt{n}+1}{4}}$. Tómese como refe-

rencia el bloque de la esquina superior izquierda, éste contendrá dos colores. El primer color se encontrará solamente en este bloque y el segundo color, en este bloque y sus dos vecinos. El tercer color estará contenido en los dos vecinos y en los tres vecinos de éstos y así sucesivamente hasta que el número de bloques que contengan el color c sea $2k-1$. Entonces, el color $c+1$ existirá de nuevo en $2k-1$ bloques, el color $c+2$, en $2k-3$ bloques y así hasta que el último color se encuentre solamente en un bloque.

En el Cuadro 4.1 se puede ver la relación de cada color con el número de bloques que lo contienen en un cuarto de tablero donde $k=4$.

Color	Número de bloques que lo contienen
1	1
2	2+1
3	3+2
4	4+3
5	4+3
6	3+2
7	2+1
8	1

Cuadro 4.1: Relación de bloques y colores

La Figura 4.7 ilustra este caso.

En este momento se tiene la información suficiente para obtener el promedio de la distancia que tiene que recorrer cada ficha en el peor caso.

En adelante a cada suma de distancia la llamaremos *color* o *suma*.

Como ya se sabe cuántas veces se repite cada color, se calcularán dos sumas por bloque y el resultado se multiplicará por 8 (ya que se está trabajando con un cuarto del tablero) para conocer el total de distancias en el tablero.

Se necesita multiplicar cada suma por el número de bloques en los que aparecen. Recordar que la distancia más grande en el tablero es $2n-4$, así que la suma total de distancias en un cuarto del tablero es:

$$\begin{aligned}
&\sum_{i=1}^{\frac{\sqrt{n}-1}{2}}((2n-2i-2\sqrt{n}i+2\sqrt{n}-2)+(2n-2i-2\sqrt{n}i))\sqrt{n}/2(2i-1)\\
&+\quad\sum_{i=1}^{\frac{\sqrt{n}-1}{2}}((2n-2(\sqrt{n}-i)-2\sqrt{n}(\sqrt{n}-i)+2\sqrt{n}-2)\\
&+\quad(2n-2(\sqrt{n}-i)-2\sqrt{n}(\sqrt{n}-i)))\sqrt{n}/2(2i-1).
\end{aligned}$$

Analicemos la ecuación anterior:

La sumatoria va a iterar sobre el número de colores y como ya se vio, el número de colores es igual a $2k=\sqrt{n}-1$, donde $k=\sqrt{\frac{n-2\sqrt{n}+1}{4}}$.

Ahora, $2n-4-2(i-1)-2\sqrt{n}(i-1)=2n-2i-2\sqrt{n}i+2\sqrt{n}-2$. La distancia más grande en el tablero es $2n-4$ y sabemos que la distancia más grande por cada color es la distancia más chica del color anterior menos cuatro, o lo que es lo mismo, $2n-4-2\sqrt{n}$. Las expresiones $2(i-1)$ y $(i-1)$ tienen la función de ajustar el cálculo en los bloques que siguen al primero.

La siguiente expresión $2n-4-2(i-1)-2\sqrt{n}i=2n-2i-2\sqrt{n}i$ tiene como objetivo calcular la distancia más pequeña del color i. Nótese que es muy parecida a la anterior, sólo que el número resultante es igual a la distancia mayor del color actual menos $\sqrt{n}+2$

El $2(i-1)$ del final de cada sumatoria es el que dicta a cuántos bloques va a pertenecer el color i.

El número de colores está dividido en dos sumatorias, esto se debe a que, como se puede ver en el Cuadro 4.1, conforme el número de color crece, la cantidad de bloques por color va creciendo, llega a un máximo y decrece hasta llegar a uno en el último bloque. Es por

esto que en la segunda sumatoria, en lugar de que aparezca la i en los mismos lugares que en la primera, aparece $\sqrt{n}-i$ para que los valores de distancia más pequeños queden en los últimos colores en vez de en los colores que están en más bloques.

La suma de distancias máximas de un cuarto del tablero será entonces:

$$
\begin{aligned}
& 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} n\sqrt{n}i - 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n}i^2 \\
- \;& 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} ni^2 - 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} ni \\
- \;& 2\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} n\sqrt{n} - \sum_{i=1}^{\frac{\sqrt{n}-1}{2}} n + \sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n} \\
- \;& 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} ni + 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n}i^2 - 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n}i \\
+ \;& 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} ni^2 + \sum_{i=1}^{\frac{\sqrt{n}-1}{2}} n + \sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n} \\
= \;& 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} n\sqrt{n}i - 2\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} n\sqrt{n} - 4\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n}i + 2\sum_{i=1}^{\frac{\sqrt{n}-1}{2}} \sqrt{n} \\
= \;& \frac{n^3 - n^2\sqrt{n} - n^2 + n\sqrt{n}}{4} - n^2 - n\sqrt{n} - \frac{n^2 - n\sqrt{n} - n + \sqrt{n}}{4} + n - \sqrt{n} \\
= \;& \frac{n^3 - 6n^2 - n^2\sqrt{n} - 4n\sqrt{n} + 3n - 3\sqrt{n}}{4}.
\end{aligned}
$$

El promedio de las distancias máximas del tablero completo se obtiene multiplicando el resultado anterior por 8 y dividiéndolo entre el número total de fichas, o sea, $n^2 - 2n\sqrt{n} - n$:

$$\begin{aligned}
&\frac{2n^3 - 2n^2\sqrt{n} - 12n^2 - 8n\sqrt{n} + 6n - 6\sqrt{n}}{n^2 - 2n\sqrt{n} - n} \\
&= \quad 2n + 2\sqrt{n} - 6 - \frac{18n\sqrt{n} - 6\sqrt{n}}{n^2 - 2n\sqrt{n} - n} \\
&\approx \quad \boxed{2n + 2\sqrt{n}.}
\end{aligned}$$

4.4. Peor caso

En la sección anterior se explica cómo se puede dar el peor caso en un tablero de $n \times n$ con bloques de $\sqrt{n} \times \sqrt{n}$ y se calcula el promedio de la distancia de cada ficha a su destino en este caso. En la subsección 4.2.2 se calcula el promedio del número de movimientos necesario para introducir y sacar fichas de los bloques.

En esta sección se muestra el cálculo para conocer el número de movimientos en el peor caso para el problema: Sea T un tablero de $n \times n$ celdas, con $n^2 - 2n\sqrt{n} + n$ fichas acomodadas en bloques de $\sqrt{n} \times \sqrt{n}$ y se quiere llevar cada ficha a su localidad destino dentro de algún bloque.

Recordemos entonces cómo procede el algoritmo descrito al inicio del capítulo. El primer paso es sacar una ficha de algún bloque hasta el corredor más cercano a ésta (dado que se está estudiando el peor caso, ninguna ficha se encontrará en su bloque objetivo al inicio). Estando la ficha sobre el corredor, trasladarla hasta su bloque objetivo, introducirla al bloque hasta su localidad destino y sacar la siguiente ficha. Se debe repetir este procedimiento hasta que cada ficha se encuentre en su localidad destino.

Sabemos que el promedio de la distancia que se tiene que recorrer en el peor caso es aproximadamente $2n + 2\sqrt{n}$ y que el promedio del número de movimientos para introducir y sacar fichas de los bloques es aproximadamente $2\sqrt{n}$. Si sumamos estas cantidades y las multiplicamos por el número de fichas en el tablero, se obtiene el máximo número de movimientos

en el peor caso:

$$\left(n^2 - 2n\sqrt{n} + n\right)\left(2n + 4\sqrt{n}\right) \quad = \quad \boxed{2n^3 - 6n^2 + 4n\sqrt{n}.}$$

4.5. Ordenando la bodega

Supongamos ahora que las fichas en el tablero no se encuentran acomodadas en bloques desde un principio, sino que se deben acomodar en bloques para después llevarlas a sus bloques objetivo. En los siguientes párrafos se analiza este caso y veremos que para tamaños grandes del tablero, el número de movimientos para acomodar las fichas es considerablemente menor al necesario en el juego $n^2 - 1$.

Formar los bloques de $\sqrt{n} \times \sqrt{n}$ toma a lo más $2\sqrt{n}\sqrt[4]{n}$ movimientos por ficha.

Supongamos que las $n^2 - 2n\sqrt{n} + n$ fichas están acomodadas a la distancia más grande posible de la posición $(1, 1)$ del tablero, es decir, la primera ficha se encuentra en la posición (n, n), las siguientes tres fichas serán adyacentes a ésta y así sucesivamente como se muestra en la Figura 4.8. La distancia Manhattan de las posiciones adyacentes a la localidad $(1, 1)$ es 1, la distancia de las posiciones adyacentes a éstas con respecto a $(1, 1)$ es 2 y así sucesivamente. Cada vez que la distancia a $(1, 1)$ aumenta en uno, el número de posiciones del tablero que están a esa distancia aumenta de la siguiente manera:

$$pp = pa + d + 1$$

donde pp es el número de posiciones presente, pa es el número de posiciones correspondientes a la distancia anterior y d es la distancia actual.

Esto sucede mientras $d < n$, pero para efectos de este problema, no es necesario conocer el número de posiciones cuando $d \geq n$ ya que como se ha visto, cuando $n > 9$ y tiene raíz

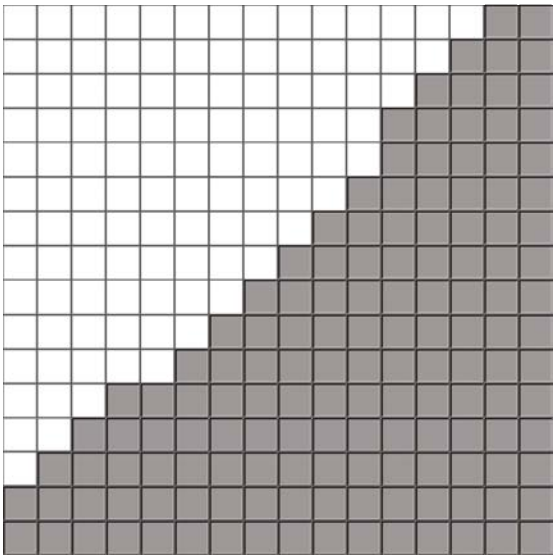


Figura 4.8: Máxima distancia de $n^2 - 2n\sqrt{n} + n$ fichas a la posición (1,1).

entera, el número de fichas que se pueden colocar en el tablero es más de la mitad.

Así que se si se quiere conocer el número de posiciones a determinada distancia de $(1, 1)$ se hace lo siguiente:

$$pp = 1 + \sum_{k=1}^{d} k + 1 \quad = \quad \frac{d^2 + d}{2} + d + 1.$$

Cuando pp es igual al número de celdas vacías del tablero, o sea, $2n\sqrt{n} - n$, cualquier posición a esa distancia de $(1, 1)$ está a su vez a una posición de una de las fichas más cercanas a $(1, 1)$. Por lo anterior, cuando $\frac{d^2+d}{2} = 2n\sqrt{n} - n$, d es la distancia menor de cualquier ficha a la posición $(1, 1)$ del tablero.

Despenjando a d de la ecuación tenemos que:

$$d = \frac{\sqrt{16n\sqrt{n} - 8n + 1} - 1}{2} \quad \approx \quad 2\sqrt{n}\sqrt[4]{n}.$$

Entonces si formamos cada bloque al inicio moviendo cada ficha hacia el más cercano, pro-

cedemos como se hace en las secciones anteriores para llevar las fichas a sus posiciones dentro de los bloques, tenemos que en el peor caso son necesarios:

$$\begin{aligned} & \left(n^2 - 2n\sqrt{n} + n\right)\left(2\sqrt{n}\sqrt[4]{n}\right) \\ \approx & \boxed{2n^3 + 2n^2\sqrt{n}\sqrt[4]{n} - 4n^2\sqrt[4]{n} + 2n\sqrt{n}\sqrt[4]{n} - 6n^2 + 4n\sqrt{n}.} \end{aligned}$$

movimientos.

Capítulo 5

Conclusiones

La motivación de este trabajo en un principio fue la de llevar objetos de una configuración inicial a una final en un espacio confinado, donde una configuración es determinada por la posición de cada uno de los objetos en un plano, como se muestra en el trabajo de Abellanas [1] donde se mueven monedas dentro de un área definida. El problema de tener muchas monedas en un área confinada cuadrada dirigió nuestra atención a uno en el que existen muchos objetos dentro de un área definida y muy poco espacio para moverlos: El juego de 15.

En este trabajo se analizan dos variaciones del juego de 15. En la sección 2.2 se estudia la razón por la que, desde una configuración inicial, sólo se puede llegar a la mitad de las configuraciones posibles en un tablero de $n \times n$. En el capítulo 3 se trata el problema de llevar $n^2 - 1$ fichas de una configuración inicial a una configuración final en un tablero de $n \times n$. En ese capítulo se muestra el desarrollo de un algoritmo para resolver el juego de $n^2 - 1$ y se calcula el máximo número de movimientos en el peor caso. Este número es una mejora al resultado de Parberry [6], que desarrolla un algoritmo que en el peor caso llega de una configuración a otra en $5n^3 - \frac{9n^2}{2} + \frac{19n}{2} - 89$ movimientos, mientras que el algoritmo presentado en este trabajo lo hace, en el peor caso en $5n^3 - \frac{17n^2}{2} + \frac{31n}{2} - 71$ movimientos.

A partir de lo anterior surge la interrogante de cuánto mejora el resultado si hay más espacio para mover las fichas. Específicamente, nos interesa saber si se puede utilizar un método similar al presentado en el capítulo 3 para ordenar una bodega. En el capítulo 4 se presenta un algoritmo basado en bloques de fichas y corredores (celdas vacías) que podrían representar estantes y corredores de una bodega respectivamente. Con el método presentado en ese capítulo, es posible hacer una permutación de los elementos entre estantes en una bodega grande con relativamente pocos movimientos.

Haciendo una comparación del problema del juego de $n^2 - 1$ y un tablero con más celdas vacías, se puede ver que teniendo poco espacio libre en un tablero de $n \times n$ ($2n\sqrt{n} - n$ celdas vacías) es posible llegar de una configuración a otra en muchos menos movimientos (menos de la mitad). Es importante señalar que con el método del capítulo 4 es posible mantener una configuración del tablero y permutar solamente dos fichas, lo cual es imposible cuando se tiene una sola celda vacía (por lo visto en la sección 2.2).

Existe mucho trabajo que se puede realizar en torno a este tema, como buscar una generalización del problema en tableros ortogonales, no necesariamente cuadrados y con un número arbitrario de celdas vacías. Mejorar los métodos presentados aquí para reducir el número de movimientos necesarios para llegar de una configuración a otra, incluso con menos espacio para mover las fichas. Otro problema interesante sería desarrollar algoritmos para bodegas con necesidades específicas y no sólo cuadradas como en el caso presentado en este trabajo.

Bibliografía

[1] M. Abellanas, S. Bereg, F. Hurtado, A. G. Olaverri, D. Rappaport, J. Tejel, Moving coins. *Computational Geometry: Theory and Applications* **34** (2006), 35–48.

[2] S. Bereg, A. Dumitrescu, J. Pach, Sliding disks in the plane. Japan Conference on Discrete and Computational Geometry 2004. *Lecture Notes in Computer Science* 3742, Springer-Verlag, Tokyo, 2005, 37–47.

[3] A. Dumitrescu, J. Pach, Pushing squares around. *Graphs and Combinatorics* **22** (2006), 37–50.

[4] F. Karlemo and P. R.J. Östergård. On sliding block puzzles. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **34** (1) : 97–107, 2000.

[5] D. Kornhauser, G. Miller, P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, 1984, 241–250.

[6] I. Parberry, A real-time algorithm for the $(n^2 - 1)$-puzzle. *Information Processing Letters*, **56**, Number 1, 13 October 1995 , pp. 23–28(6).

[7] P.D.A Schofield, Complete solution of the eight puzzle. In N. L. Collins and D. Michie, editors, *Machine Intelligence 1*, American Elsevier, 1995, pp. 125–133.

[8] D. Ratner, and M. Warmuth, Finding a Shortest Solution for the extension of the 15-Puzzle Is Intractable. *Journal of Symbolic Computation* **10** (1990), 111–137.

[9] W.E.Storey, Notes on the 15 puzzle. *American Journal of Mathematics*, 2(4):399-404, 1879.

[10] R. M. Wilson, Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory Series B* 16 (1974), 86–96.