Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias en Ciencias de la Computación

Optimización del conjunto de mosaicos en el auto-ensamblado de ADN de patrones en 2D

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Felipe Leza Álvarez

Ensenada, Baja California, México

2016

Tesis defendida por

Felipe Leza Álvarez

y aprobada por el siguiente Comité

Dr. Israel Marck Martínez Pérez

Director de tesis

Miembros del comité

Dr. Carlos Alberto Brizuela Rodríguez

Dr. Hugo Homero Hidalgo Silva

Dr. Miguel Ángel Alonso Arevalo

Dr. Jesús Favela Vara

Coordinador del posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez

Directora de estudios de Posgrado

Resumen de la tesis que presenta **Felipe Leza Álvarez** como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Optimización del conjunto de mosaicos en el auto-ensamblado de ADN de patrones en 2D

Resumen aprobado por:

Dr. Israel Marck Martínez Pérez
Director de tesis

El problema de sintetizar un conjunto de mosaicos para el auto-ensamblado de patrones (PATS, por sus siglas en inglés), consiste en encontrar un conjunto mínimo de tipos de mosaicos para que auto-ensamblen un patrón P de dos dimensiones previamente definido. El auto-ensamblado se realiza con el modelo abstracto de ensamblado de mosaicos, cuyas unidades fundamentales son mosaicos cuadrados que representan una molécula de ADN con extremos pegajosos. Este modelo provee el comportamiento combinatorio del auto-ensamblado de ADN y puede utilizarse como modelo de cómputo y como método de construcción para nanoestructuras de ADN. En este último, el objetivo es diseñar secuencias de ADN que automáticamente ensamblen una estructura deseada. El problema PATS pertenece al conjunto de problemas NP-difícil. Los métodos propuestos en el estado del arte abordan el problema aplicando diferentes heurísticas. Sin embargo, estos métodos no obtienen los conjuntos óptimos para todos los casos de prueba. En este trabajo de investigación se propone un algoritmo de recocido simulado para resolver el problema. Se utilizan dos tipos de representación: con grafos y matricial. El diseño incluye operaciones para generar el vecindario de soluciones, así como la utilización de tres perfiles de enfriamiento. Los experimentos realizados con los patrones de prueba obtienen los conjuntos óptimos para patrones pequeños (menores a 8×8), aunque para patrones de mayor tamaño solamente se logra superar a los métodos voraces del estado del arte, pero no así a los métodos basados en procesamiento paralelo.

Palabras clave: Problema PATS, aTAM, Recocido Simulado, Auto-ensamblado de ADN, Sistemas de Ensamblado de Mosaicos.

Abstract of the thesis presented by **Felipe Leza Álvarez** as a partial requirement to obtain the Master of Science degree in Computer Science.

Optimization tile set in DNA self-assembly of 2D patterns

Abstract approved by:

Dr. Israel Marck Martínez Pérez
Thesis Director

The Patterned self-Assembly Tile set Synthesis (PATS) problem consists in finding a minimal set of unit square tiles that self-assemble a predefined two-dimensional pattern. DNA self-assembly is performed by using the abstract Tile Assembly Model (aTAM), where a tile represents a DNA molecule with four sticky ends protruding from it. This model can be viewed from two different angles: as a computational model, on one hand, and as a construction method for nanoscale structures, on the other. In the latter case, the goal is to design DNA sequences that automatically assemble into a target structure. However, the task of finding a minimum-size tile set is known to be NP-hard. To date, four different approaches have been proposed for the resolution of this problem. In this thesis, a simulated annealing algorithm to solve the PATS problem is presented. In total, five neighborhood operations were developed on a matrix structure and three cooling schedules were considered. Computational experiments were performed on 14 patterns to find an optimal solution. Experimental results show that simulated annealing algorithm performs very well when considering small-size instances. Nevertheless, for instances greater than 8×8, obtained results cannot improve those of the state of art.

Keywords: PATS problem, aTAM, Simulated Annealing, DNA self-assembly, Tile Assembly Systems.

Dedicatoria

A mis padres Sofia y Felipe.

A mi familia.

Agradecimientos

A mis padres Felipe Leza Rodríguez y Sofía Álvarez Rosas por su apoyo incondicional y su cariño. Por ser mi inspiración para seguir aprendiendo y motivarme a hacerlo.

A mis amigos por escucharme y ofrecerme su compañía a lo largo de mi estancia en Ensenada.

A mi asesor de tesis, por su paciencia y por guiarme en el trabajo de investigación.

A mi comité de tesis, por sus observaciones y comentarios para mejorar mi investigación.

A los profesores del Departamento de Ciencias de la Computación de CICESE, por compartir sus conocimientos.

A cada uno de mis compañeros del posgrado en Computación, por compartir sus ideas y experiencias.

A todas las personas que compartieron su misión y su visión del mundo.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría.

Tabla de contenido

Página

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

El auto-ensamblado es el proceso por el cuál pequeños componentes automáticamente se organizan entre ellos en estructuras más grandes y complejas. En la naturaleza existen ejemplos de tal proceso: los lípidos se auto-ensamblan para formar la membrana celular o el virus bacteriófago auto-ensambla una cápside para invadir a las bacterias. En ingeniería, dichos procesos implementan una serie de tareas simples de crecimiento que son controladas de manera autónoma y son llamados sistemas de auto-ensamblado algorítmico. La principal motivación del estudio de estos sistemas está relacionada con la fabricación de estructuras artificiales a escala nanométrica para la manipulación de la materia (Doty, 2012).

El ácido desoxirribonucleico (ADN) se utiliza en muchos laboratorios en el diseño e implementación de sistemas de auto-ensamblado para la construcción de complejas nanoestructuras de ADN, debido a que es fácil de sintetizar y sus propiedades físicas son bien conocidas (Reif y Labean, 2012). La investigación del auto-ensamblado con ADN tiene dos enfoques importantes: por un lado, como modelo de computación; por el otro, como un método para construir estructuras a nanoescala (Lempiäinen, 2015). En el primer enfoque, el objetivo es codificar la entrada de un problema computacional a cadenas de ADN, de tal manera que las interacciones entre ellas produzcan una codificación de salida que solucione el problema (Doty, 2012). En el segundo, el objetivo es diseñar secuencias de ADN que automáticamente ensamblen una estructura deseada. Doty (2012) expone que se debe aprender a programar moléculas para que se manipulen entre ellas mismas, ya que muchas de las formas tradicionales de control manual no son posibles a pequeña escala.

Algunos autores describen aplicaciones potenciales del auto-ensamblado de ADN, entre las que se encuentran la construcción de estructuras y dispositivos a escala molecular para la industria microelectrónica (Haddow y Gautam, 2013), primitivas computacionales y biosensores (Lempiäinen, Czeizler y Orponen, 2011). En este sentido, los trabajos se han centrado en la construcción de estructuras más complejas a partir de plantillas de dos dimensiones (Reif y Labean, 2012). Una forma de implementar estas plantillas de auto-ensamblado es a partir de unidades cuadradas llamadas mosaicos.

Un mosaico puede describirse como un complejo de ADN que tiene cuatro extremos pegajosos formados por cadenas simples de ADN (Doty, 2012). Los segmentos pegajosos con secuencias complementarias de Watson-Crick se unen por el proceso de hibridación permitiendo pegar mosaicos. Los mosaicos más conocidos para la construcción de nanoestructuras son los mosaicos de doble cruzamiento (Winfree, Liu, Wenzler y Seeman, 1998). El modelo combinatorio que describe el comportamiento de estos mosaicos fue propuesto por Winfree (1998), el cual es conocido como *el modelo abstracto de ensamblado de mosaicos* (aTAM, por sus siglas en inglés). Los elementos básicos de este modelo son mosaicos cuadrados con etiquetas en cada

uno de sus lados. Las etiquetas representan un tipo de pegado molecular que permite unir al mosaico con otros que tengan un pegado compatible en sus lados adyacentes (Winfree, 1998). Debido a que los lados del mosaico tienen una orientación, no se pueden rotar ni reflejar. Hay un número finito de diferentes tipos de mosaicos, cada uno disponible de manera ilimitada. Se produce un ensamble sobre una superficie de dos dimensiones agregando un mosaico a la vez e iniciando a partir de una semilla. Los mosaicos de ADN se van pegando y la estructura crece conforme al ensamble deseado (Lempiäinen, 2015).

Para la construcción de estas nanoestructuras de ADN, es importante definir el conjunto de mosaicos que auto-ensamblan un patrón específico, ya que si se minimiza el número de tipos de mosaicos se reduce también el riesgo de error en la implementación química en laboratorio (Lin, Liu, Rinker y Yan, 2006). Ma y Lombardi (2008) formulan este problema como uno de optimización combinatoria al que denominan *síntesis del conjunto de mosaicos en el auto-ensamblado de patrones* (PATS, por sus siglas en inglés). Este consiste en, dado un patrón coloreado de dos dimensiones, construir un sistema de ensamblado de mosaicos para que auto-ensamble dicho patrón, tal que se minimice el número de tipos de mosaicos necesarios.

Czeizler y Popa (2013) demostraron que el problema PATS es NP-difícil y que la búsqueda de un conjunto mínimo de mosaicos para un patrón, requiere una cantidad de tiempo exponencial. Los algoritmos propuestos en el estado del arte para solucionarlo son: un algoritmos voraz (Ma y Lombardi, 2008), uno exhaustivo (Göös y Orponen, 2011), uno basado en heurísticas (Lempiäinen, 2015) y un algoritmo genético (Lozano, 2014).

1.1. Planteamiento del problema

El problema consiste en la optimización del diseño de un sistema de ensamblado de mosaicos con el modelo aTAM, los cuales se auto-ensamblan para formar patrones coloreados. La entrada del problema es un patrón rectangular de dos colores y la salida es un conjunto de mosaicos aTAM. El objetivo es diseñar un sistema que se encuentre formado por un número mínimo de tipos de mosaicos y que de manera determinista se auto-ensamblen para formar el patrón deseado. Este problema es conocido como PATS (Ma y Lombardi, 2008) y es NP-difícil (Czeizler y Popa, 2013), por lo que para encontrar un conjunto mínimo de mosaicos se requiere una cantidad de tiempo exponencial. Así, el problema puede ser abordado con algoritmos heurísticos para encontrar soluciones óptimas. En este trabajo de investigación se diseñó e implementó un algoritmo de recocido simulado para encontrar un conjunto de mosaicos mínimo para 14 casos de prueba del problema.

1.1.1. Preguntas de investigación

El algoritmo de recocido simulado necesita respuesta a las siguientes preguntas relacionadas al problema que se quiere solucionar:

- ¿Cómo representar el problema para realizar la búsqueda de una solución?
- ¿Cuáles serán los vecinos de una solución?
- ¿Cuál es la función de evaluación de una solución?
- ¿Cómo generar una solución inicial?
- ¿Cómo debe ser el enfriamiento?
- ¿Cuál es es número de iteraciones para el algoritmo?
- ¿Cuál es la mejor configuración de los parámetros del algoritmo con los que se obtienen mejores soluciones?
- ¿En general, qué tan bueno es el desempeño de recocido simulado con respecto a otros algoritmos?

1.2. Objetivos

1.2.1. Objetivo general

El objetivo es diseñar e implementar un algoritmo de recocido simulado que minimice la cardinalidad del conjunto de tipos de mosaicos necesarios para la construcción de un sistema de auto-ensamblado de patrones de dos colores y de tamaño $m \times n$. El sistema debe estar formado por un número mínimo de tipos de mosaicos aTAM y que de manera determinista se auto-ensamblen para formar un patrón de entrada.

1.2.2. Objetivos específicos

- Seleccionar una representación del problema.
- Diseñar operaciones para generar un vecindario de soluciones.
- Seleccionar perfiles de enfriamiento para el algoritmo.
- Codificación del algoritmo de recocido simulado.
- Realizar experimentos con 14 patrones de prueba.

- Ejecutar el algoritmo con diferentes parámetros de configuración.
- Seleccionar la configuración de parámetros con la que se obtenga los mejores resultados para todos los patrones de prueba.
- Realizar la simulación de los resultados obtenidos con el software ISU TAS para validar que el conjunto de mosaicos obtenido auto-ensamblan correctamente los patrones de prueba.
- Comparar los resultados obtenidos por el algoritmo diseñado con los del estado del arte.

1.3. Metodología

Para el cumplimiento de los objetivos, se planteó el problema como uno de optimización mono-objetivo. Se realizó una revisión del estado del arte y se seleccionó el algoritmo de recocido simulado, tomando en cuenta que en dicha revisión no se encontraron trabajos relacionados con este algoritmo. Así mismo, se revisaron las representaciones del problema utilizadas en los algoritmos propuestos en el estado del arte y se eligió una representación con grafos (Ma y Lombardi, 2008) y una matricial (Lozano, 2014).

La solución inicial se construyó como en el trabajo propuesto por Göös y Orponen (2011), la cual consiste en considerar que todos los mosaicos de un patrón de tamaño $m \times n$ son todos diferentes. El patrón de entrada se representó con dos grafos y se diseñó una operación para reducir el número de vértices. Como una segunda propuesta, se representó al patrón de entrada con una matriz de mosaicos aTAM de tamaño $m \times n$ y se diseñaron cinco operaciones para generar el vecindario de soluciones.

Por otro lado, se seleccionaron valores para cada uno de los parámetros del algoritmo de recocido simulado y se ejecutaron experimentos con 14 patrones de prueba. Con el resultado de los experimentos realizados se seleccionó aquella configuración de parámetros con la que se obtuvo los mejores resultados para todos los patrones de prueba. Para verificar la validez de dichos resultados, se construyeron los sistemas de ensamblado de los patrones de prueba con el software ISU TAS. Finalmente, se realizó la comparación de los resultados del algoritmo propuesto con los del estado del arte.

1.4. Organización de la tesis

El resto de la tesis se organiza de la siguiente manera: el Capítulo 2 introduce los conceptos relacionados con este trabajo de investigación. Se describe qué es el ADN y su estructura, cómo está formado un mosaico de ADN y en qué consiste el auto-ensamblado de mosaicos. Se explica el modelo aTAM, utilizado en la mayoría

de las ideas propuestas en esta investigación. Se presenta también el estado del arte y la definición formal del problema PATS.

El Capítulo 3 detalla la representación del problema así como los algoritmos para la construcción de la solución inicial. Se enuncian las propiedades que deben cumplir las soluciones vecinas y se presentan las operaciones diseñadas para generarlas. Además, se enuncia el algoritmo diseñado para evaluar las soluciones generadas y se realiza una breve explicación de la plataforma de programación OPT4J utilizada para la codificación del algoritmo de optimización.

El Capítulo 4 presenta los experimentos realizados a lo largo de este trabajo de investigación. Se describen las características de los patrones de prueba utilizados, se especifican los parámetros con los que se ejecutaron los experimentos y se presentan los resultados obtenidos. Así mismo, se muestran las simulaciones realizadas con el software ISU TAS para validar los conjuntos de mosaicos obtenidos. Finalmente, en el Capítulo 5 se enuncian las conclusiones del trabajo de investigación y algunas ideas de trabajo a futuro. El Apéndice A contiene los mejores resultados obtenidos para los 14 patrones de prueba con el algoritmo de recocido simulado. Se desglosan detalladamente los elementos de cada uno de los conjuntos obtenidos para cada patrón de prueba.

Capítulo 2. Marco teórico

2.1. El ADN, su estructura y manipulación

2.1.1. Estructura del ADN

Una molécula de ADN es un polímero lineal formado por una serie de monómeros denominados *nucleótidos* (Pérez Jiménez y Sancho Caparrini, 2003). A su vez, cada nucleótido consta de:

- un azúcar (desoxirribosa) que tiene cinco átomos de carbono enumerados del 1' al 5', y que en el carbono 3' tiene un grupo hidroxilo (OH).
- Un grupo fosfato (P), unido al azúcar por el carbono 5'.
- Una base nitrogenada, unida al azúcar por el carbono 1'.

Cada nucleótido se identifica de acuerdo a la base nitrogenada que contiene. Existen cuatro tipos de bases nitrogenadas: adenina, citosina, guanina y timina, que se representan por A, C, G y T, respectivamente. La adenina y guanina pertenecen al grupo de las purinas, mientras que la citosina y la timina son del grupo de las pirimidinas. Otro aspecto importante de los nucleótidos es la forma en que pueden enlazarse entre sí. Existen dos maneras diferentes:

- Mediante un enlace fosfodiester (covalente): el grupo fosfato $5'$ de un nucleótido se une al grupo hidroxilo $3'$ de otro.
- Mediante un enlace de hidrógeno que se realiza a través de las bases nitrogenadas.

Los enlaces fosfodiester entre nucleótidos permite la construcción de cadenas simples de ADN. Estas poseen dos extremos de comportamientos muy diferentes; uno de ellos con un grupo fosfato dispuesto a nuevos enlaces, y el otro con un grupo hidroxilo. La diferencia que existe entre ambos extremos proporciona una polaridad o direccionamiento a las moléculas de ADN. De esta manera, puede tener la dirección $5' \rightarrow 3'$ o $3' \rightarrow 5'$.

Por convención, una cadena simple de ADN siempre está escrita en la dirección $5' \rightarrow 3'$, a menos que se indique lo contrario. Por ejemplo, la palabra $ACGTCGACTAC$ representa una cadena simple de ADN con dirección $5' - ACGTCGACTAC - 3'$.

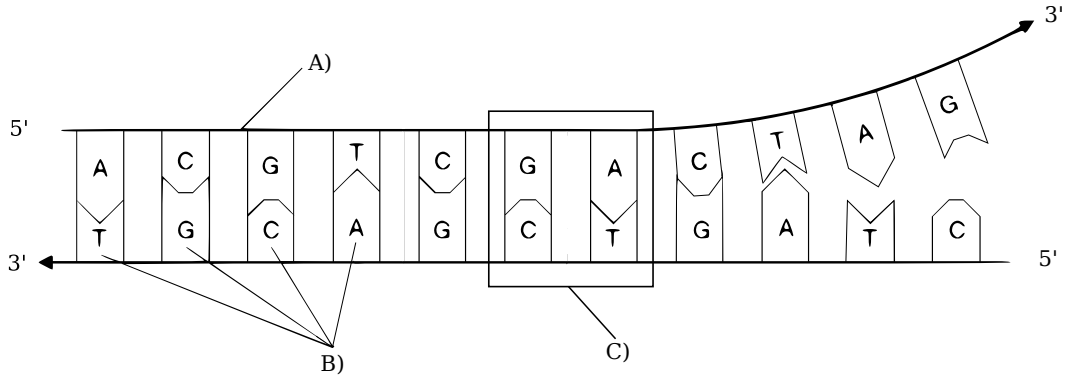


Figura 1. Estructura del ADN: A) Una columna vertebral. B) Nucleótidos. C) Principio de complementariedad entre las bases G y C, y entre A y T. Las cadenas simples de ADN tienen una orientación opuesta entre ellas, formando una cadena doble ADN. Adaptado de (Rozenberg et al., 2012).

El enlace de hidrógeno es más débil que el de fosfodiester y se rige por el principio de complementariedad: la adenina sólo se puede enlazar con la timina, y la citosina sólo con la guanina. Con la combinación de enlaces fosfodiester y de hidrógeno se obtienen cadenas dobles de ADN (Pérez Jiménez y Sancho Caparrini, 2003), como la que se muestra en la Figura 1. Por ejemplo, la cadena simple $5' - ACGTCGACTAC - 3'$ se unirá a otra cadena $5' - GTAGTCGACGT - 3'$ para formar una cadena doble de ADN de la siguiente manera:

$$5' - ACGTCGACTAC - 3'$$
$$3' - TGCAGCTGATG - 5'.$$

Dicha combinación proporciona hebras dobles de ADN, formando la estructura de doble hélice como en la Figura 2, que consiste en dos cadenas simples alineadas de forma antiparalela (una con la dirección $5' \rightarrow 3'$ y la otra con $3' \rightarrow 5'$) y unidas por enlaces de hidrógeno; los nucleótidos están unidos por enlaces fosfodiester.

2.1.2. Manipulación del ADN

A continuación se describen de manera general algunas de las operaciones que se realizan con moléculas de ADN y que constituyen la base tanto de la ingeniería genética como de la computación molecular basada en ADN.

2.1.2.1. Desnaturalización y Renaturalización

La estructura de doble hélice del ADN es especialmente sólida. Su estabilidad proviene de los enlaces de hidrógeno y de fosfodiester, así como de la distribución espacial de sus componentes (Pérez Jiménez y Sancho

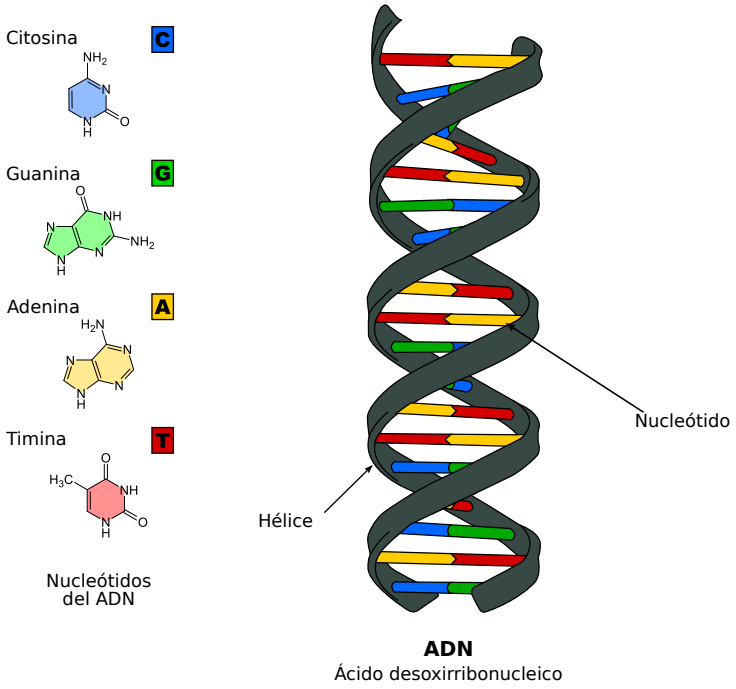


Figura 2. Doble hélice de ADN. Modificado de (Sponk, 2012).

Caparrini, 2003).

El proceso biológico por el cual se rompen los enlaces de hidrógeno de una doble hebra dando como resultado dos cadenas simples de ADN se denomina *desnaturalización*. Este proceso se puede replicar en un laboratorio calentando la solución en la que se encuentren las cadenas doble de ADN, hasta una temperatura entre 85 y 94 grados centígrados, a la que se conoce como temperatura de fusión.

El proceso inverso, llamado *renaturalización* (ó hibridación), consiste en someter la solución con cadenas simples complementarias a un enfriado lento hasta aproximadamente los 55 grados centígrados, para permitir que las bases nitrogenadas complementarias se unan a través de sus enlaces de hidrógeno.

2.1.2.2. Replicación de ADN

La operación de replicación de ADN tiene lugar cuando copias idénticas de ADN son creadas en un proceso iterativo llamado PCR (*Polymerase Chain Reaction*), que se basa en el uso de la enzima ADN polimeraza. Es la técnica más importante y revolucionaria en biología molecular para obtener *in vitro*, millones de copias de un fragmento de ADN a partir de una sola molécula.

Generalmente, la PCR inicia con la desnaturalización o separación de la doble hélice de ADN mediante el calentamiento de la muestra a una temperatura entre 94 y 96 grados centígrados para romper los puentes de hidrógeno que las une. Una vez separadas las cadenas de ADN, se alinean los iniciadores (fragmentos

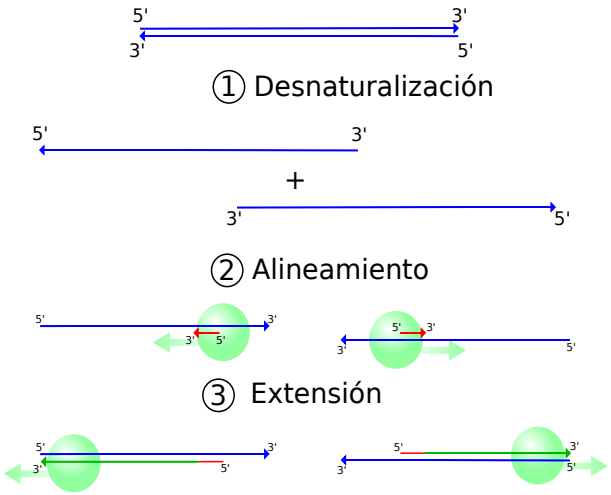


Figura 3. Replicación del ADN con PCR.

de ADN de 15 a 30 nucleótidos alrededor de la región a amplificar y que aportan el extremo $3'$ libre para que inicie la transcripción) a sitios específicos complementarios de las cadenas sencillas de la región que se va a amplificar. Para que esto suceda se disminuye la temperatura entre 40 y 60 grados centígrados lo que permite la unión (alineamiento) de los iniciadores. Finalmente, se sintetiza una nueva cadena en sentido 5' a 3' para lo cual se incrementa la temperatura, por lo general a 72 grados centígrados. Estas tres etapas: 1) desnaturalización, 2) alineamiento y 3) extensión del ADN (Figura 3), se repiten sucesivamente y en cada nuevo ciclo se amplifica simultáneamente la región de interés de las dos cadenas complementarias (Serrato, Flores, Aportela y Sierra, 2016).

2.1.3. Nanoestructuras de ADN

Una nanoestructura de ADN es un complejo multimolecular formado por cadenas simples de ADN que están parcialmente unidas a lo largo de sus segmentos (Rozenberg et al., 2012). El concepto de nanoestructura de ADN fue utilizado por primera vez por Seeman (2004). En particular, los tipos de *motivos* útiles más usados en las nanoestructuras de ADN son:

- Estructura de lazo: está formada por una cadena simple de ADN que se enrolla para hibridar sobre ella misma. Es decir, un segmento de la cadena (cerca del final $5'$), se une con otro segmento (cerca del final $3'$) de la misma cadena. Por ejemplo, la estructura de la Figura 4A) está formada por una región de cadena doble de ocho nucleótidos (secuencia CACGGTGC en la parte inferior) y una región simple formada por la secuencia TTTT.
- Extremo pegajoso: consiste en una cadena simple de ADN que sobresale del final de una doble hélice. En la Figura 4B) el extremo pegajoso mostrado (ATCG) sobresale de la cadena doble en la parte inferior.

Estos extremos pegajosos son usados para unir dos nanoestructuras de ADN a través del proceso de hibridación de sus cadenas simples complementarias.

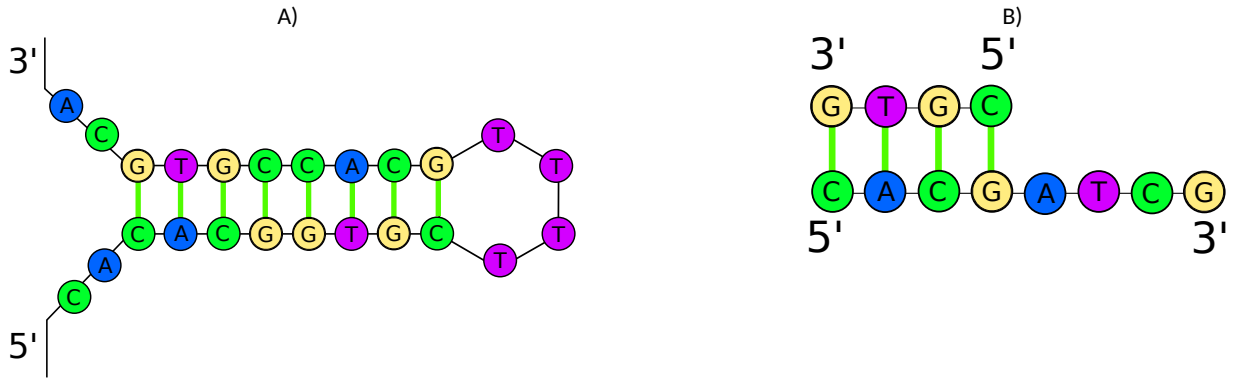


Figura 4. Motivos de ADN: A) Estructura de lazo. B) Extremo pegajoso. Adaptado de (Rozenberg et al., 2012).

2.1.4. Mosaicos de ADN

Un mosaico es una nanoestructura de ADN que tiene un cierto número de extremos pegajosos, tal como se muestra en la Figura 5. Los extremos pegajosos pueden hibridar con secuencias complementarias de otros mosaicos para enlazarlos. Un enrejado de ADN está compuesto por un grupo de mosaicos de ADN unidos a través de sus extremos pegajosos (Figura 6). Winfree et al. (1998), desarrollaron una familia de mosaicos de ADN conocidas como mosaicos DX, formados por cuatro o cinco cadenas simples de ADN (Figura 7A). Así mismo, de manera experimental, demostraron que con estos tipos de mosaicos se pueden formar enrejados en dos dimensiones. Posteriormente, desarrollaron otra familia conocida como TX, que están formados por seis cadenas simples (Figura 7B)).

Los segmentos pegajosos corresponden a secuencias de nucleótidos con una energía especifica de hibridación. Dicha fuerza de hibridación depende principalmente del número de pares de bases y de una temperatura. Es decir, segmentos pegajosos más largos implican una fuerza de pegado mayor (Winfree, 1998). La molécula esta compuesta por cadenas sencillas de ADN formadas por una secuencia de 37 a 42 nucleótidos (Fu y Seeman, 1993).

2.1.5. Moléculas de ADN de doble cruzamiento (DX)

Las moléculas de ADN con doble cruzamiento contienen dos sitios de cruce entre dominios helicoidales, la distancia entre estos dos puntos se especifica con un número múltiplo de dos. La estructura de estas moléculas son altamente estables (Fu y Seeman, 1993). En la literatura, existen dos grandes clases: las paralelas

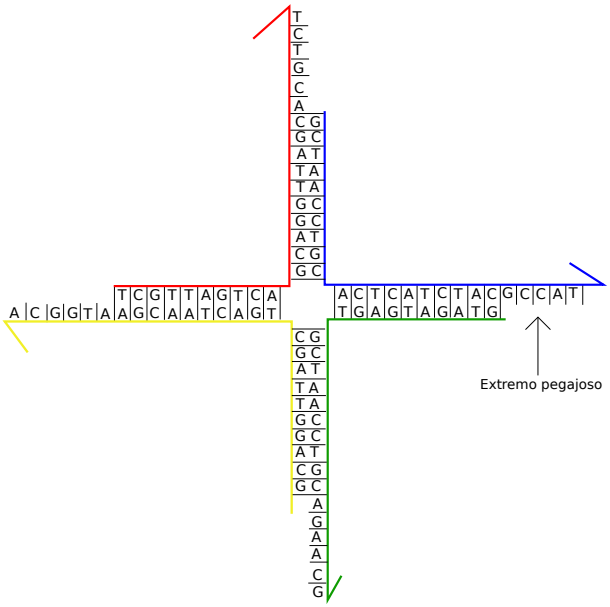


Figura 5. Mosaico de ADN (Carbone y Seeman, 2004).

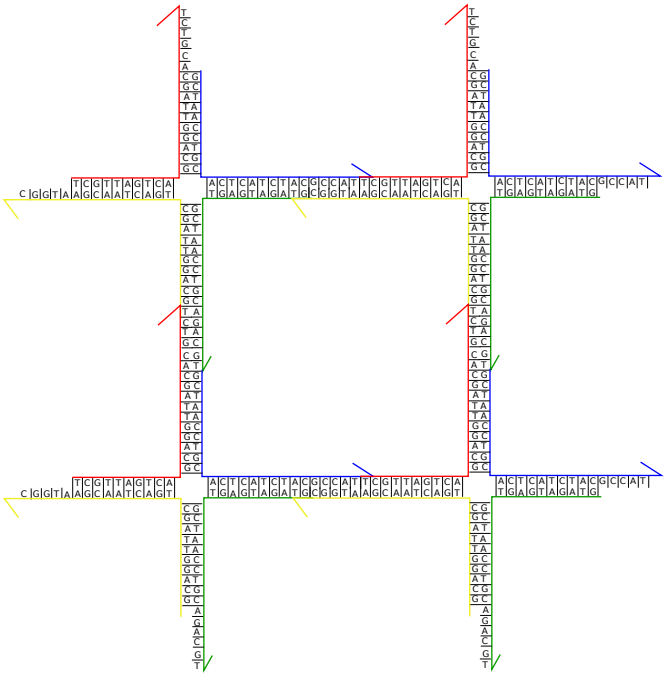


Figura 6. Enrejado de ADN (Carbone y Seeman, 2004).

y las antiparalelas. Las antiparalelas están constituidas por dos tipos de moléculas: par e impar. La molécula par (DAO) está compuesta por cuatro cadenas sencillas de ADN que se complementan entre ellas con una longitud de 12.6 nm y una secuencia de 37 nucleótidos (Figura 8A)). La molécula impar (DAE) contiene cinco fragmentos de cadenas sencillas de ADN que se complementan entre si, con una longitud de 14.3 nm y una secuencia de 42 nucleótidos (Figura 8B)). Las dos moléculas tienen cuatro extremos salientes, formados por una secuencia de cinco nucleótidos que pueden variar de longitud dependiendo del diseño de los mismos.

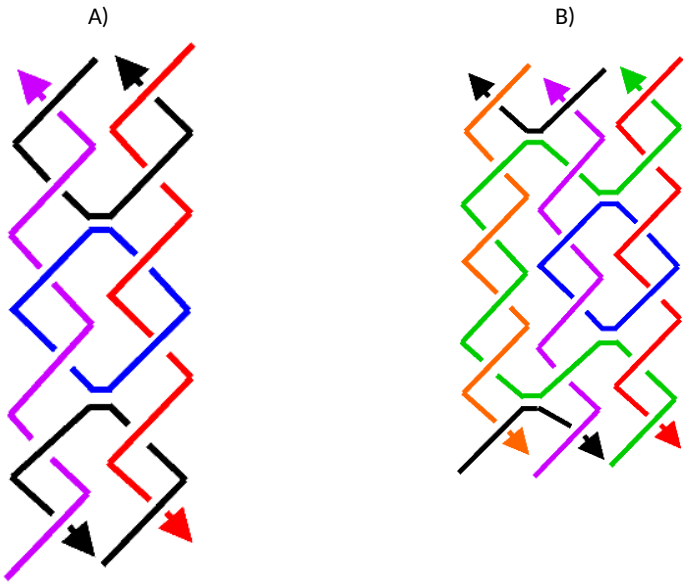


Figura 7. Familias de mosaicos de ADN: A) DX y B) TX. Tomado de (Mao, 2004).

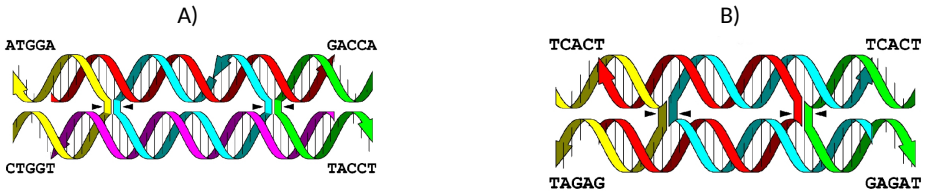


Figura 8. Moléculas de doble cruzamiento: A) DAO y B) DAE. Tomado de (Rothemund et al., 2004).

2.1.6. Computación molecular

La computación molecular es un campo multidisciplinario donde el objetivo es programar moléculas para que realicen algún cómputo, fabriquen un objeto deseado o controlen el funcionamiento de un sistema molecular. La idea central de estas investigaciones es que los datos pueden ser codificados en biomoléculas, por ejemplo, moléculas de ADN. Es decir, se utiliza un programa molecular, que consta de una colección de moléculas, localizadas en un substrato para realizar una función específica.

El surgimiento de la computación molecular está asociado al experimento de Adleman (1994), quien resolvió un caso particular de un problema de grafos. Adleman codificó este problema en cadenas de ADN y las manipuló en tubos de ensayo para obtener una solución. Inicialmente, los avances en esta área estaban enfocados en la solución de problemas computacionales difíciles, sin embargo, uno de los principales aportes ha sido para las nanociencias, donde es notable su contribución en el entendimiento del auto-ensamblado y la aplicación de técnicas de programación molecular para construir objetos a escala nanométrica o dispositivos con funcionalidades específicas.

2.1.7. El experimento de Adleman

Adleman (1994) fue el primero en codificar información en secuencias de ADN para realizar simples bio-operaciones. Su primer experimento con el cómputo con ADN consistió en resolver un caso particular de un problema NP-completo llamado *el problema del camino Hamiltoniano*. Este problema considera un grafo dirigido G, que contenga dos vértices asignados como v_{inicio} y v_{final}. El grafo G contiene un camino Hamiltoniano sí, y sólo si, existe una secuencia de aristas dirigidas $e_1, e_2, \ldots, e_z$ que inicia en v_{inicio}, termina en v_{final} y cada vértice es visitado solamente una vez.

La secuencia de pasos del Algoritmo 1 fue seguida por Adleman para resolver dicho problema para un grafo con siete vértices y 14 aristas, con $v_{inicio} = 0$ y $v_{final} = 6$ (Figura 9). Para codificar los datos de entrada, se asocia en primer lugar a cada nodo i ($0 \leq i \leq 6$) una cadena simple de ADN sintetizada, s_i, de longitud 20. Para cada i se designa por $s_i^{'}$ a los diez primeros caracteres de la cadena s_i, y por $s_i^{''}$ a los últimos diez. Cada arista (i, j) del grafo está codificada por la correspondiente cadena complementaria $s_i^{''} s_j^{'}$ (Pérez Jiménez y Sancho Caparrini, 2003).

Para realizar el paso 1, se introducen inicialmente en el tubo de ensayo, para cada nodo i y para cada arista (i, j), muchas copias de s_i y cadenas complementarias de $s_i^{''} s_j^{'}$. Mediante un proceso de hibridación, se generan cadenas dobles que codifican todos los posibles caminos de G; a través de un mecanismo de filtrado se obtiene en un tubo de ensayo única y exclusivamente todos esos caminos. Enseguida, en el paso 2, mediante la técnica de PCR se amplifican sólo aquellos caminos que comienzan en v_{inicio} y terminan en v_{final}. Para el paso 3, utilizando la técnica de electroforesis en gel, se seleccionan las moléculas de longitud 7×20 pares de nucleótidos. Posteriormente, en el paso 4 se desnaturalizan las dobles hebras de ADN del paso anterior, obteniendo cadenas simples; se sintetizan cadenas $\bar{s_i}$ y se mezclan con las anteriores utilizando la técnica de hibridación y se conservan en el tubo de ensayo únicamente las doble hebras resultantes. Finalmente, en el paso 5 se verifica si existe alguna solución resultante en el tubo de ensayo, cada una de las cuales será una solución correcta del caso particular considerado. Se amplifica el contenido del tubo de ensayo y se aplica la técnica de electroforesis en gel.

Algoritmo 1 Encontrar caminos Hamiltonianos

Entrada: Un grafo dirigido G con n vértices con v_{inicio} y v_{final} asignados.
Salida: Caminos Hamiltonianos

1: Generar caminos aleatorias en G.
2: Seleccionar sólo los caminos que inician en v_{inicio} y terminal en v_{final}.
3: Seleccionar sólo los caminos que visitan exactamente n vértices.
4: Seleccionar sólo los caminos que visitan todos los vértices del grafo a lo más una vez.
5: Leer los caminos resultantes (si existen).

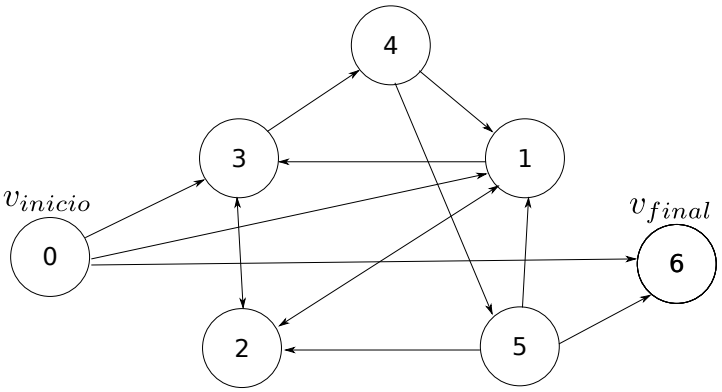


Figura 9. El grafo de Adleman.

Los resultados del experimento de Adleman representan el primer ejemplo práctico de computación molecular y muestran el uso potencial de las moléculas de ADN como estructura de datos física, susceptibles de ser utilizadas como medio de almacenamiento. Además, demuestran la capacidad de las moléculas de ADN para realizar cálculos (Pérez Jiménez y Sancho Caparrini, 2003).

2.2. Auto-ensamblado algorítmico de ADN

El auto-ensamblado es el proceso por el cual una colección relativamente simple de componentes que están en un estado desorganizado, espontáneamente y de manera autónoma se juntan para formar estructuras más complejas. Este proceso es guiado sólo por las interacciones entre los componentes, típicamente siguiendo un conjunto de reglas.

Los sistemas de auto-ensamblado incluso se encuentran en la naturaleza y en una gran variedad de sistemas biológicos, lo que ha motivado a los investigadores a auto-ensamblar diversas estructuras a nanoescala de manera experimental, desde estructuras fractales hasta robots moleculares. Muchos de estos ejemplos demuestran que el auto-ensamblado se puede usar para construir formas geométricas, mecánicas y objetos computacionales a escala nanométrica. Entre las aplicaciones con mayor potencial se incluyen la producción de nuevos materiales con propiedades específicas, tecnologías médicas que sean capaces de diagnosticar e incluso tratar enfermedades *in vivo* a nivel celular (Kari, Kopecki, Meunier, Patitz y Seki, 2015).

Los modelos desarrollados más conocidos para este tipo de sistemas son el *modelo abstracto de ensamblado de mosaicos* (aTAM) y el *modelo cinético de ensamblado de mosaicos* (kTAM) propuestos por Winfree (1998), los cuales se basan en el estudio teórico de los mosaicos de Wang (1961) y en los complejos de ADN sintetizados por Seeman (1982). La versión kTAM comprende aspectos de las propiedades físicas y químicas, que

permiten el estudio de las posibles causas de error y los mecanismos potenciales para detectarlos (Patitz, 2014). El modelo aTAM es una abstracción de alto nivel que ignora la posibilidad de errores y proporciona la teoría para el estudio de este tipo de sistemas.

2.2.1. El modelo aTAM

El aTAM es un modelo formal de auto-ensamblado de moléculas (Rothemund y Winfree, 2000). Los elementos fundamentales del modelo son unidades cuadradas llamadas mosaicos, con etiquetas en cada uno de sus lados como se muestra en la Figura 10. Cada etiqueta a_i representa un tipo de pegado molecular (secuencia de ADN) y tiene asociado una fuerza de pegado g_i (representado por los cuadros en color negro de la Figura 10), el cual debe ser un número mayor o igual a cero. En un ensamble se utilizan mosaicos con fuerzas de pegado igual a uno para representar a los elementos de un patrón y mosaicos con fuerza de pegado igual a cero, uno o dos para formar los límites del patrón, es decir, para controlar su tamaño. Los mosaicos no se pueden rotar ni reflejar. A una "temperatura" τ, los mosaicos se pueden unir si la suma de las fuerzas de pegado g_i de sus lados coincidentes es al menos τ.

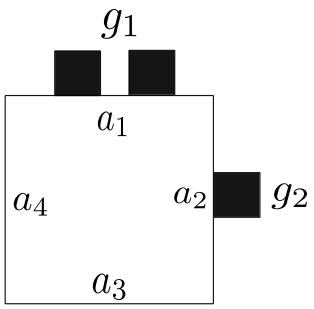


Figura 10. Mosaico del modelo aTAM. Cada etiqueta a_i representa una secuencia de ADN asociada con una fuerza de pegado g_i. Los valores g_1 y g_2, representados por los cuadros negros en los lados, indican las fuerzas de pegado, es decir g_1 tiene una fuerza de pegado de dos y g_2 de uno. Para los lados restantes la fuerza de pegado es cero. Un valor de cero significa que no se permite pegar otro mosaico en ese lado del mosaico.

El color que se le asigna a cada uno de los mosaicos sólo es una representación de los tipos de moléculas DAE y DAO con las que se construyen. En el auto-ensamblado de estas moléculas se percibe un relieve que provoca un cambio de color entre las moléculas y es por ello que se abstraen como dos colores diferentes (negro o blanco) (Rothemund et al., 2004).

Winfree (1998) demostró que este modelo es universalmente computable (todo lo que sea computable con una máquina de Turing también es computable por el modelo aTAM), lo que motivó a implementar compuertas lógicas utilizando mosaicos formados con moléculas de ADN. La Figura 11 muestra un ejemplo de como los mosaicos aTAM computan la función XOR y ensamblan un patrón llamado triángulo de Sierpinski. Los mosaicos con etiquetas numéricas (Figura 11A)) implementan la función XOR: la entrada de la función son

los valores de los lados oeste y sur del mosaico y la salida se coloca en los lados este y norte. Los siguientes tres mosaicos corresponden a los límites del triángulo de Sierpinski. Como se observa en la Figura 11B), el ensamblado comienza con el mosaico de la esquina, llamada semilla, y posteriormente se empiezan a formar los límites. Los mosaicos ejecutan la función XOR con los valores de entrada correspondientes, formando gradualmente el triángulo de Sierpinski.

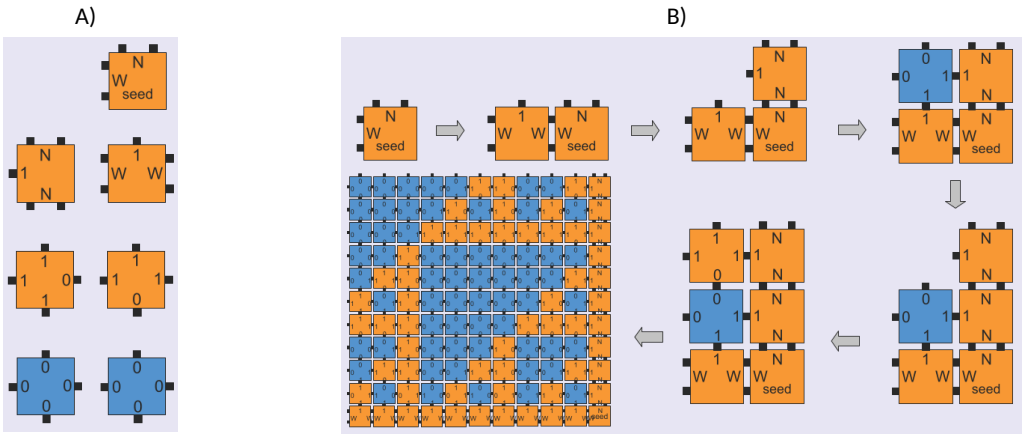


Figura 11. Sistema de ensamblado de mosaicos con el modelo aTAM. A) Conjunto de mosaicos. B) Ensamblado de mosaicos. Tomado de (Doty, 2012).

2.2.2. Ensamblado de patrones

Formalmente, un sistema de auto-ensamblado rectilíneo de mosaicos se define de la siguiente manera (Kari et al., 2015):

Sea $\mathbb{N}$ el conjunto de enteros positivos y $n \in \mathbb{N}$, donde $[n] = \{0, 1, 2, \ldots, n-2, n-1\}$. Para $k \geq 1$, un patrón k-coloreado es una función parcial que va de $\mathbb{N}^2$ al conjunto de índices (colores) $[k]$, y un patrón rectangular k-coloreado (con ancho w y alto h) es un patrón cuyo dominio es $[w] \times [h]$. Considere ahora un alfabeto Σ de pegado. Un tipo de mosaico t es una tupla $t = (g_N, g_E, g_S, g_O, c)$, donde $g_N, g_E, g_S, g_O \in \Sigma$ representa el valor de pegado del norte, este, sur y oeste de t, respectivamente, mientras que $c \in \mathbb{N}$ corresponde al color de t. Para referirse a los valores g_N, g_E, g_S y g_O de la tupla, se utiliza $t(N), t(E), t(S)$ y $t(O)$, respectivamente; se utiliza además $t(c)$ para el color de t. Para un conjunto T de tipos de mosaicos, un ensamble α sobre T es una función parcial de $\mathbb{N}^2$ a T. Su patrón, denotado por $P(\alpha)$, cumple que $dom(P(\alpha)) = dom(\alpha)$ y $P(\alpha)(x, y) = c(\alpha(x, y))$ para todo $(x, y) \in dom(\alpha)$.

Un *sistema de ensamblado rectilíneo de mosaicos* (RTAS, por sus siglas en inglés) es un par $\mathcal{T} = (T, \sigma_L)$, donde T es un conjunto de tipos de mosaicos y σ_L una semilla en forma de L, la cual es un ensamble sobre

otro conjunto de tipos de mosaicos disjuntos de T tal que

$$dom(\sigma)_L = \{(-1,-1) \cup ([w] \times \{-1\}) \cup (\{-1\} \times [h])\}$$

para algún $w, h \in \mathbb{N}$. Su tamaño se define como el número de tipos de mosaicos utilizados, es decir, $|T|$.

Por ejemplo, para un patrón coloreado con dos colores (blanco y negro), con $w = 4$ y $h = 8$; su dominio es $\{0,1,2,3\} \times \{0,1,2,3,4,5,6,7\}$, donde cada par ordenado (x, y) corresponden a las posiciones de los mosaicos blancos y negros en el patrón como se muestra en la Figura 12. Además, el dominio de la semilla σ_L corresponde a los pares ordenados en los mosaicos de color gris. Considere que $\Sigma = \{0, 1\}$ y los tipos de mosaicos del patrón de la Figura 13B), para el primer mosaico de la izquierda, la tupla correspondiente es t = (0, 0, 0, 0, blanco). Así, $t(N) = 0$, $t(E) = 0$, $t(S) = 0$ y $t(O) = 0$, son los tipos de pegado de los lados norte, este, sur y oeste, respectivamente; y $t(c) = blanco$ es el color del mosaico.

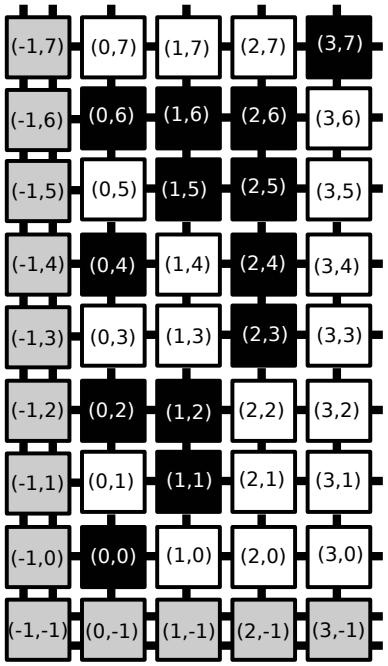


Figura 12. Patrón rectangular coloreado con dos colores (blanco y negro), con dominio $[4] \times [8]$ y con una semilla σ_L con $dom(\sigma)_L = \{(-1,-1) \cup ([4] \times \{-1\}) \cup (\{-1\} \times [8])$.

Como regla general, en todos los RTAS el ensamblado estará delimitado por la semilla, por lo que los primeros mosaicos comenzarán a pegarse a partir de σ_L, cumpliendo con las siguiente reglas:

Regla 2.1. Para un ensamble α, un mosaico $t \in T$ puede pegarse en la posición (x, y) si

1. $\alpha(x, y)$ es indefinido,

2. $\alpha(x-1,y)$ y $\alpha(x,y-1)$ están definidos,

3. $t(O) = \alpha(x-1,y)[E]$ y $t(S) = \alpha(x,y-1)[N]$.

De manera informal, un mosaico puede pegarse a un ensamble α en la posición (x,y) si no se ha asignado ningún mosaico en (x,y) y las posiciones $(x-1,y)$ y $(x,y-1)$ tienen mosaicos asignados y además, si los valores de pegado oeste y sur de t coinciden con los valores de pegado del lado este del mosaico $(x-1,y)$ y del norte de $(x,y-1)$, respectivamente. De esta manera, al inicio del ensamble, la posición $(0,0)$ es la única disponible donde un mosaico puede pegarse. Similar al modelo aTAM de Winfree, en un sistema RTAS la temperatura a la que se realiza el ensamble es dos y todas las fuerzas de pegado de los mosaicos que no forman parte de la semilla σ_L es de uno.

Regla 2.2. Un conjunto de mosaicos T es dirigido si se cumple que para cualquier tipo de mosaicos diferentes $t_1, t_2 \in T$, $t_1(O) \neq t_2(O)$ o $t_1(S) \neq t_2(S)$.

Así, un sistema RTAS $\mathcal{T} = (T, \sigma_L)$ es dirigido si el conjunto de mosaicos T es dirigido. Seki (2014) menciona que un RTAS dirigido admite un ensamble único en cualquier posición. Es decir, en cada posición del ensamble sólo se admite un único tipo de mosaico. Por ejemplo, los tipos de mosaicos de la Figura 13B). La semilla σ_L delimitará el ensamble como se observa en la Figura 13C). La posición $(0,0)$, es la única posición donde un mosaico con oeste = 1 y sur = 0 puede pegarse (Figura 13C)). Así, el mosaico $(1,1,0,0,negro)$ es el que se incorpora en la posición $(0,0)$, lo que permite a las posiciones (0,1) y (1,0) aceptar un nuevo mosaico. El ensamblado ocurre de manera rectilínea de sur-oeste a norte-este hasta que todas las posiciones tengan un mosaico asignado (Figura 13D)).

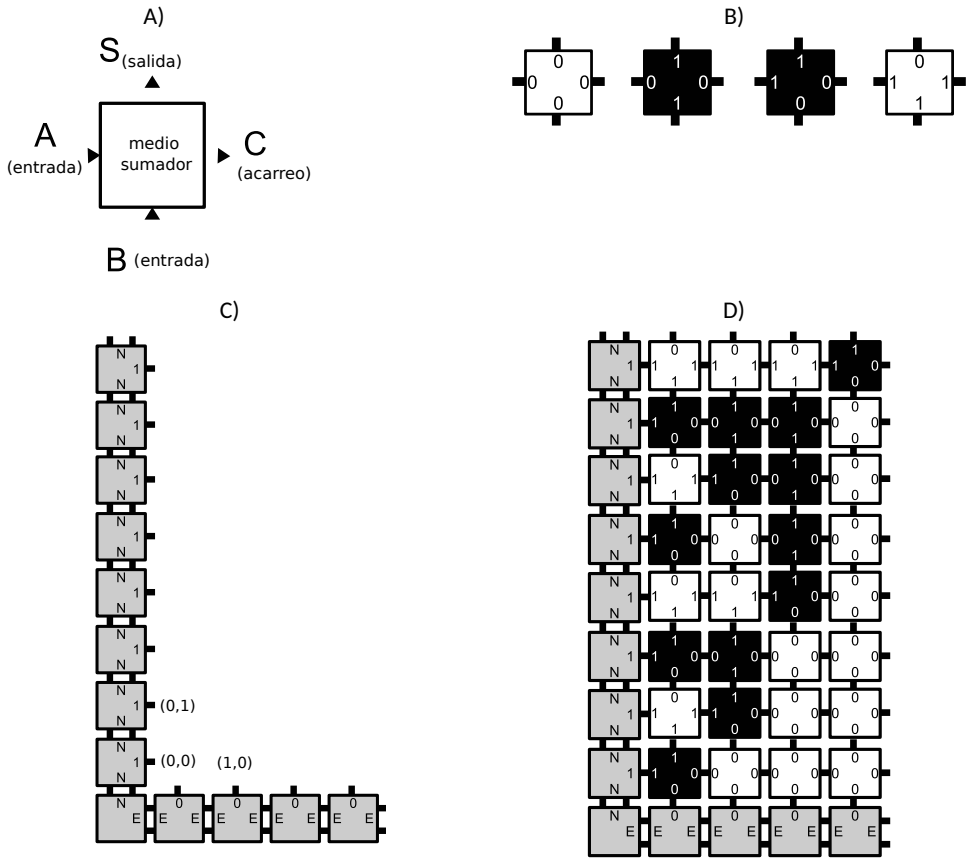


Figura 13. Ensamblado de mosaicos. A) Mosaico modelo para un medio sumador. B) Mosaicos necesarios para implementar el medio sumador. C) Semilla σ_L. D) El resultado del ensamble es un contador binario, en el que cada mosaico representa un medio sumador, las entradas A y B se suman, el resultado se coloca en S y en C, el color negro representa 1 y blanco 0. Adaptado de (Kari et al., 2015).

2.3. Síntesis del conjunto de mosaicos en el auto-ensamblado de patrones

El problema de sintetizar un conjunto de mosaicos para el auto-ensamblado de patrones (PATS, por sus siglas en inglés) fue propuesto por Ma y Lombardi (2008). Consiste en minimizar un sistema RTAS para que auto-ensamble un patrón dado, el cual es medido por la cardinalidad del conjunto de tipos de mosaicos. Como un RTAS mínimo tienen la propiedad de ser dirigido, puede definirse como sigue (Seki, 2014):

Definición 1. Dado un patrón rectangular P, se requiere encontrar un sistema RTAS de cardinalidad mínima que únicamente auto-ensamble P.

La entrada a este problema es un patrón o figura rectangular que contiene k colores y la salida es un conjunto mínimo de mosaicos aTAM. A cada tipo de mosaico se le asigna un color, que corresponde al color que aparece en la misma posición del patrón. El caso general del problema es conocido como k-PATS, donde k

es el número de colores únicos en el patrón de entrada. Muchas variantes del k-PATS se han demostrado que son NP-difíciles, entre las que se encuentran valores para $k = \{60, 29, 11, 3, 2\}$ (Kari et al., 2015). En este trabajo de investigación se consideró el caso particular para $k = 2$.

2.4. Estado del arte

Ma y Lombardi (2008) formulan el problema PATS y proponen dos algoritmos voraces para solucionarlo. Para esto, utilizan un modelo de grafos para la representación del problema. Comienzan con una solución inicial a la que llaman *solución trivial*, la cual consiste en suponer que todos los mosaicos que forman el patrón de tamaño $m \times n$ son diferentes. Los algoritmos voraces propuestos se llaman PATS_Bond y PATS_Tile. El primero reduce los tipos de pegado y el segundo los tipos de mosaicos.

Göös y Orponen (2011) proponen un algoritmo exacto para el problema PATS. Su método extiende el modelo propuesto por Ma y Lombardi (2008) para obtener un algoritmo de ramificación y poda exhaustivo al que llaman PS-BB. La búsqueda comienza a partir de un conjunto formado por $m \times n$ mosaicos diferentes. Realizan una búsqueda exhaustiva sobre el conjunto de todas las particiones de un patrón de tamaño $m \times n$. Cada una de las particiones exploradas es almacenada y se construye un grafo acíclico dirigido y definen reglas para podarlo. Los resultados reportados incluyen soluciones mínimas para patrones de tamaño 6×6 y soluciones aproximadas para patrones más grandes.

Lempiäinen et al. (2011) diseñan un algoritmo de búsqueda llamado PS-H que encuentra soluciones pequeñas pero no necesariamente mínimas para patrones de tamaño grande. Para ello, realizan modificaciones al algoritmo PS-BB agregando guías heurísticas de búsqueda, con el fin de reducir el número de particiones a explorar. Mientras que la estrategia de poda del algoritmo PS-BB reduce el espacio de búsqueda de una manera balanceada, el algoritmo PS-H trata de optimizar vorazmente el orden en que las particiones son exploradas. La idea básica de la heurística es tratar de minimizar el efecto que tiene una operación de unión sobre otras particiones que ya han sido combinadas. La ejecución de su algoritmo se realiza en paralelo. En los resultados reportados se menciona que el algoritmo es capaz de encontrar las soluciones óptimas para los patrones de prueba utilizados.

Lozano (2014) propone un algoritmo genético que lleva por nombre AG-PATS, donde la representación del problema y sus operadores genéticos suponen una estructura matricial. Su representación surge del principio de visualizar al patrón de entrada como una imagen formada de píxeles, donde cada píxel representa un mosaico y a cada mosaico le corresponde una posición en la matriz para formar al patrón. El algoritmo AG-PATS construye sistemas de ensamblado de mosaicos no deterministas, es decir, las soluciones obtenidas

para todos los patrones de prueba no generan un patrón único.

El trabajo de investigación más reciente que aborda el problema PATS es la tesis doctoral de Lempiäinen (2015), en la que presenta los resultados del algoritmo publicado en (Göös, Lempiäinen, Czeizler y Orponen, 2013).

La Tabla 1 contiene los resultados obtenidos por los algoritmos propuestos para resolver el problema PATS, utilizando el conjunto de patrones de prueba de referencia en la literatura. La solución inicial de todos los algoritmos contienen $m \times n$ tipos de mosaicos (solución trivial). Para cada uno de los patrones se muestra el número de tipos de mosaicos obtenidos. En la Tabla 2 se enuncian las diferentes representaciones del problema utilizadas por cada uno de los algoritmos propuestos.

Tabla 1. Resultados de las propuestas desarrolladas en el estado del arte para el problema PATS. El símbolo "–" indica que el autor no incluyó el patrón correspondiente en sus experimentos. [1] (Ma y Lombardi, 2008), [2] (Göös y Orponen, 2011), [3] (Lozano, 2014), [4] Lempiäinen (2015). Los conjuntos de mosaicos obtenidos con el algoritmo propuesto por Lozano (2014) no son dirigidos (ver Regla 2.2).

Patrón	Tamaño	Voraz [1]	PS-BB [2]	AG-PATS [3]	PS-H [4]
Contador binario	36 (6×6)	19	–	4	–
	255 (15×15)	121	–	5	–
	1024 (32×32)	–	307	19	20
Esquina de árbol	529 (23×23)	–	192	11	25
Triángulo de Sierpinski	36 (6×6)	21	–	7	–
	64 (8×8)	37	–	9	–
	144 (12×12)	80	20	18	–
	289 (17×17)	172	23	20	–
	1024 (32×32)	595	42	23	4
	4096 (64×64)	–	95	25	4
Tablero de ajedrez	36 (6×6)	19	–	2	–
	255 (15×15)	119	–	11	–
Aleatorio	144 (12×12)	–	–	10	–
	256 (16×16)	–	–	17	–

Tabla 2. Representación del problema PATS en el estado del arte.

Autor(es)	Algoritmo	Representación
Ma y Lombardi (2008)	PATS_Bond y PATS_Tile	Grafos
Göös y Orponen (2011)	PS-BB	Grafos
Lozano (2014)	AG-PATS	Matricial
Lempiäinen (2015)	PS-H	Basada en PS-BB

2.5. Optimización

La optimización se encuentra en gran parte de las actividades que realizamos y es aplicable a muchas áreas de nuestra vida cotidiana. Por ejemplo, la podemos aplicar en el campo de la ingeniería para encontrar la mejor ruta de un robot que realiza una tarea, de tal manera que finalice en el menor tiempo posible o utilice la menor cantidad de energía.

Un problema de optimización se describe como un problema de minimización o maximización. Es decir, se requiere minimizar o maximizar una función $f(x)$. La función $f(x)$ es llamada función objetivo y el vector x es un conjunto de variables independientes o parámetros. Al número de elementos en x se le llama la dimensión del problema. Cuando se requiere minimizar una función, se nombra a esta función como función de costo (Ecuación 1). En el caso de maximización, a la función se le llama función de aptitud (Ecuación 2) (Ansari y Hou, 1997).

$$\min_x f(x) \implies f(x) \text{ es la función de costo u objetivo} \tag{1}$$

$$\max_x f(x) \implies f(x) \text{ es la función de aptitud u objetivo} \tag{2}$$

2.5.1. Optimización combinatoria

Existen problemas de optimización en los que las variables independientes solamente toman valores de un conjunto discreto. A este tipo de problemas se les conoce como problemas de optimización combinatoria. Por ejemplo, supongamos que un vendedor quiere visitar cuatro ciudades, iniciando y finalizando en una cierta ciudad. El vendedor se encuentra en la ciudad A, y quiere ir a las ciudades B, C y D, terminando en la ciudad E. El vendedor quiere visitar todas las ciudades de tal manera que la distancia de viaje sea la mínima. Hay seis posibles soluciones S_i para el problema:

$$S_1 : A \to B \to C \to D \to E$$
$$S_2 : A \to B \to D \to C \to E$$
$$S_3 : A \to C \to B \to D \to E$$
$$S_4 : A \to C \to D \to B \to E$$
$$S_5 : A \to D \to B \to C \to E$$
$$S_6 : A \to D \to C \to B \to E$$

Es fácil para el viajero calcular la distancia de cada una de las rutas enumerando todas las posibles soluciones en este ejemplo. A este problema se le conoce como el problema del agente viajero (TSP, por sus siglas en inglés). La búsqueda en todas las soluciones posibles en un problema combinatorio es nombrada búsqueda de fuerza bruta o búsqueda exhaustiva. Para instancias con un espacio de búsqueda pequeño es conveniente implementar este tipo de estrategias, ya que no se requiere mucho tiempo para obtener la solución. Sin embargo, cuando se incrementa el número de ciudades a n, también aumenta el espacio de búsqueda en $(n-2)!$ soluciones posibles (la ciudad inicial y final no se permutan). Es decir, si el agente viajero quiere visitar 50 ciudades, entonces el número de soluciones que se tiene que revisar es $48!$. Evidentemente es imposible hacer una búsqueda de fuerza bruta para encontrar la solución al problema.

Además del TSP, muchos problemas de optimización encontrados en la ingeniería y otros campos, han sido catalogados como problemas difíciles de resolver, en los que un algoritmo exhaustivo no es aplicable. Debido a esto se han propuesto nuevas técnicas de optimización, entre las que se encuentran los algoritmos heurísticos. Una heurística es un métodos que utiliza reglas generales o de sentido común para resolver un problema. Usualmente no se espera que encuentre la mejor solución. Sin embargo, puede encontrar soluciones que estén lo suficientemente cerca de la mejor. El término metaheurística se utiliza para describir a una familia de algoritmos heurísticos. Entre ellos, se encuentran los algoritmos genéticos, el recocido simulado, la optimización por cúmulo de partículas, etc.

2.6. Elementos de un algoritmo de búsqueda

En los algoritmos de búsqueda hay tres conceptos básicos comunes: la representación, el objetivo y la función de evaluación. La representación codifica soluciones candidatas, el objetivo describe el propósito a cumplir, y la función de evaluación regresa un valor específico que indica la calidad de una solución en particular. A continuación se describen estos conceptos básicos necesarios para implementar un algoritmo de búsqueda (Michalewicz y Fogel, 2004).

2.6.1. Representación

Para el TSP comentado en la sección anterior, una posible representación del problema que consta de n ciudades consiste en una permutación de números naturales $1, \ldots, n$ donde cada número corresponde a la ciudad que se visita. Con esta representación, el espacio de búsqueda S contiene todas las posibles permutaciones ($n!$).

Para cada problema, la representación de una solución potencial y su correspondiente interpretación está

relacionada con el espacio de búsqueda y su tamaño. El espacio de búsqueda está definido por el multiconjunto de soluciones a las que se pueden llegar con la representación y los operadores propuestos. La selección de la representación es de gran importancia ya que si no resulta adecuado, el inicio de la búsqueda puede agregar numerosas soluciones no válidas o duplicadas a la lista de posibilidades.

2.6.2. El objetivo

Una vez que se ha definido la representación se tiene que decidir qué es lo que se está buscando y cuál es el objetivo del problema. Por lo general, se define con una expresión matemática que representa la tarea a optimizar. En el caso del problema TSP, el objetivo es típicamente minimizar la distancia total recorrida por el agente viajero al visitar todas las ciudades solamente una vez y llegar a la ciudad destino. Por ejemplo, para una solución $S_1 : A \to B \to C \to D \to E$ del problema mostrado en la sección anterior, la distancia recorrida con S_1 está dada por la Ecuación 3. Se calculan las distancias para todas las soluciones posibles S_i y se obtiene el valor mínimo (Ecuación 4). Así, el objetivo expresado en términos matemáticos está dado por la Ecuación 5, donde x es la ciudad de salida y y es la ciudad de llegada.

$$t_{S_1} = dist(A, B) + dist(B, C) + dist(C, D) + dist(D, E), \tag{3}$$

$$\text{mín}\{t_{S_1}, t_{S_2}, t_{S_3}, t_{S_4}, t_{S_5}, t_{S_6}\}, \tag{4}$$

$$\text{mín} \sum dist(x, y). \tag{5}$$

2.6.3. La función de evaluación

La función de evaluación es típicamente una asociación del espacio de posibles soluciones candidatas con un conjunto de números (por ejemplo, los reales), donde a cada elemento del espacio de soluciones se le asigna un valor numérico que indica su calidad. Esto nos permite comparar entre soluciones alternativas y decidir no solamente cuál es mejor que otra, sino también cuantificar la diferencia (Michalewicz y Fogel, 2004).

En el diseño de la función de evaluación se debe considerar que para muchos problemas, las soluciones de interés estarán en un subconjunto del espacio de soluciones S. Por este motivo, solamente estaremos

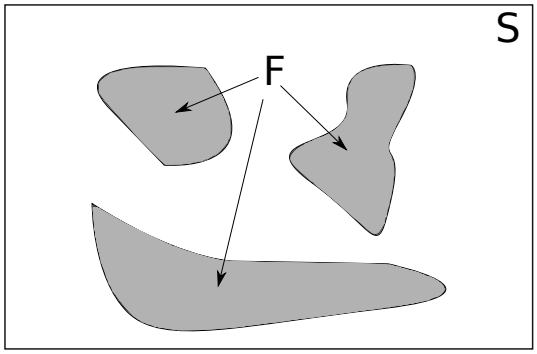


Figura 14. Espacio de búsqueda S y sus soluciones alcanzables F. Tomado de (Michalewicz y Fogel, 2004).

interesados en soluciones *alcanzables* F (Figura 14), es decir, soluciones que satisfacen las restricciones especificas del problema (Michalewicz y Fogel, 2004).

2.6.4. Definición del problema de búsqueda

Dado un espacio de búsqueda S junto con sus partes alcanzables $F \subseteq S$, encontrar $x \in F$ de tal manera que

$$eval(x) \leq eval(y) \tag{6}$$

para toda $y \in F$ (Michalewicz y Fogel, 2004). Es decir, se utiliza una función de evaluación para obtener la solución con el menor valor de evaluación (problema de minimización) o para obtener los valores más grandes (problema de maximización).

El punto x que satisface la Ecuación 6 se llama solución *global*. Encontrar una solución global para un problema puede ser muy difícil. Sin embargo, encontrar la mejor solución es más fácil cuando el espacio de búsqueda es un subconjunto pequeño del total (Michalewicz y Fogel, 2004).

2.6.5. Vecindarios y óptimos locales

Si solamente nos concentramos sobre una región del espacio de búsqueda que esté “cerca” de un punto en particular, entonces se está buscando sobre el *vecindario* de ese punto (Michalewicz y Fogel, 2004). En la Figura 15 se muestra un espacio de búsqueda abstracto S y un punto $x \in S$. El vecindario de x es $N(x)$, denotado por el interior del circulo alrededor de x y representa un conjunto de todos los puntos del espacio de búsqueda S que están cerca en alguna medida, del punto x. Típicamente, la función de distancia se

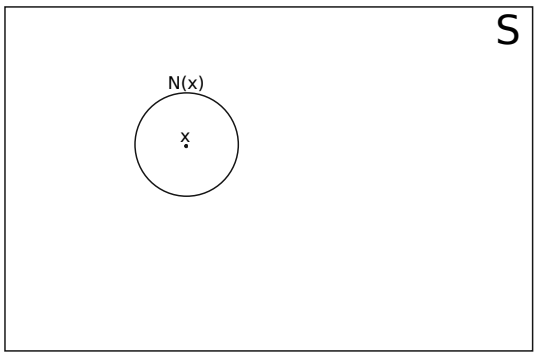


Figura 15. Una solución x y su vecindario $N(x)$. Tomado de (Michalewicz y Fogel, 2004).

define sobre el espacio de búsqueda de la siguiente manera (Michalewicz y Fogel, 2004):

$$dist : S \times S \to \mathbb{R} \tag{7}$$

y el vecindario $N(x)$ es

$$N(x) = \{y \in S : dist(x, y) \leq \epsilon\} \tag{8}$$

para algún $\epsilon \geq 0$. Si y satisface la condición para $N(x)$, entonces x está en el $\epsilon-$vecindario (Michalewicz y Fogel, 2004). Para espacios de búsqueda que están definidos sobre variables continuas, se selecciona la distancia euclidiana (Michalewicz y Fogel, 2004) definida como:

$$dist(x, y) = \sqrt{\sum_{i=1}^{n}(x_i, y_i)^2}. \tag{9}$$

Ya que se ha hablado del concepto de vecindario ahora se puede hablar del concepto de un óptimo *local*. Una solución potencial $x \in F$ es un óptimo *local* con respecto al vecindario de N, si y solamente sí $eval(x) \leq eval(y)$ para todo $y \in N(x)$ (Michalewicz y Fogel, 2004). De igual manera, para este ejemplo se supone que se toma el criterio de minimización. Es decir, se obtiene la solución con el valor de evaluación mínimo de todas las soluciones que hay en el vecindario N.

2.7. Recocido simulado

El algoritmo de recocido simulado es una heurística inspirada en el comportamiento de los fluidos cuando se exponen a un enfriamiento controlado en la producción de cristales. Recocido simulado puede mejorar

varios aspectos de las técnicas de búsqueda local (Rozenberg et al., 2012). En estas heurísticas, una solución inicial es gradualmente mejorada haciendo una pequeña perturbación o cambio. Al conjunto de soluciones que se obtienen haciendo estas perturbaciones se les llama vecindario de la solución y sus elementos son llamados vecinos. Una solución es seleccionada del conjunto de vecinos. Si dicha solución es mejor que la actual, entonces esta última es reemplazada por la solución vecina seleccionada y el proceso continua hasta que ya no haya mejoras. Una de las principales desventajas de estos enfoques es que la búsqueda termina en un óptimo local, en vez de uno global (Rozenberg et al., 2012). El algoritmo de recocido simulado permite seleccionar soluciones peores para que sean aceptadas. Sin embargo, para garantizar que la trayectoria de la búsqueda se encuentra en la dirección de mejoramiento, ésta debe ser realizada de manera controlada (Rozenberg et al., 2012). En el caso de recocido simulado, el mecanismo de control está inspirado por la forma en que la energía termodinámica disminuye cuando una sustancia es sometida a un proceso de enfriamiento.

2.7.1. Algoritmo de recocido simulado

El algoritmo de recocido simulado fue propuesto por Kirkpatrick, Gelatt y Vecchi (1983). Se llama así por su analogía con el proceso físico de recocido de materiales sólidos, que consiste en incrementar la temperatura de un sólido cristalino hasta su punto de fundición, y después disminuir la temperatura hasta alcanzar un estado base de energía mínima. El algoritmo se inspira en este proceso termodinámico para hacer una búsqueda de un mínimo global. El núcleo principal lo constituye el algoritmo propuesto por Metropolis, Rosenbluth, Rosenbluth, Teller y Teller (1953), que simula el proceso de recocido que consiste en enfriar un material que se encuentra a una temperatura elevada.

Cuando un material sólido es calentado sobrepasando su punto de fundición y enfriado otra vez, las propiedades estructurales del material resultante están relacionadas con la tasa de enfriamiento (Rozenberg et al., 2012). Un ejemplo de este proceso es la formación de los cristales, los cuales representan un estado de baja energía y son el resultado de un enfriamiento lento. En contraste, un enfriamiento rápido resulta en cristales con alta energía e imperfecciones. Las leyes de la termodinámica enuncian que para una sustancia en un estado de equilibrio a una temperatura ambiente T, la probabilidad de un incremento en la energía de magnitud δE, está dada por la Ecuación 10, donde k es la constante de Boltzmann (Rozenberg et al., 2012).

$$P(\delta E) = \exp\left(\frac{-\delta E}{kT}\right). \tag{10}$$

En la simulación propuesta por Metropolis, el material es considerado como un sistema de átomos. Para

cada una de las posibles configuraciones de los átomos, la energía derivada puede ser calculada. En cada una de las iteraciones, un átomo es expuesto a una pequeña perturbación, se calcula el incremento de su energía δE, y si $\delta E \leq 0$ una nueva configuración es aceptada. De lo contrario, puede ser aceptada con cierta probabilidad $P(\delta E)$. Esto se realiza por medio de la generación de un número aleatorio r dentro del intervalo $[0, 1)$, y la nueva configuración se acepta si $r < p(\delta E)$. Este algoritmo puede simular el proceso de recocido si en cada repetición se reduce el valor de T una vez que el sistema ha alcanzado su punto de equilibrio con la temperatura actual, hasta llegar a su punto límite de enfriamiento (Rozenberg et al., 2012).

Kirkpatrick formalizó y realizó una analogía entre los elementos del algoritmo de Metropolis y los de un problema de optimización combinatoria de la siguiente manera (Rozenberg et al., 2012):

i) El estado de un sistema es análogo a la solución del problema de optimización combinatoria.

ii) La energía del sistema corresponde al costo de la función objetivo a optimizar.

iii) Las perturbaciones que se hacen al sistema son los movimientos dentro del vecindario de soluciones.

iv) El mecanismo de enfriamiento es similar a los mecanismos del algoritmo de búsqueda.

v) El estado de enfriamiento del sistema corresponde a la solución final.

Dichas analogías dieron como resultado los elementos principales del algoritmo de recocido simulado (Tabla 3). La constante de Boltzmann, dado que corresponde a una medida física, es eliminada de esta analogía, por lo que el término kT se reduce a una constante t que representa a la temperatura del sistema (Rozenberg et al., 2012).

Tabla 3. Analogía Metropolis - Optimización Combinatoria.

Simulación termodinámica	Optimización combinatoria
Estados del sistema	Soluciones alcanzables
Energía	Costo
Cambio de estado	Movimiento en el vecindario
Temperatura	Parámetro de control
Estado de enfriamiento	Solución heurística

El Algoritmo 2 muestra de manera general el funcionamiento del recocido simulado. Los parámetros de entrada son una solución inicial s_0 (línea 1), un valor mayor a cero para la temperatura inicial t_0 (línea 2) y un mecanismo de enfriamiento CS que decrementa el valor de t_0 (línea 3). El algoritmo comienza seleccionando una solución vecina s para s_0 (línea 6). Si s es mejor que s_0, entonces s se convierte en la solución actual s_0 (líneas 7-9); en caso contrario, s es aceptada de acuerdo al criterio probabilístico de las líneas 11 y 12. El algoritmo termina si el número máximo de iteraciones $ITER$ se ha alcanzado (línea 16), en otro caso se decrementa el valor de la temperatura actual t (línea 17) y se vuelve a comenzar en la línea 6 hasta que

el valor de t llega a cero (la condición de paro se cumple en la línea 18). Cuando el algoritmo termina, s_0 contiene la solución final. Sin embargo, puede no ser la de menor costo que se visitó durante la búsqueda. Es por ello que, es una práctica común regresar la mejor solución encontrada en vez de s_0.

De acuerdo a la Ecuación 10, el algoritmo de recocido simulado acepta movimientos que no mejoran la solución (Kirkpatrick et al., 1983). Es decir, la probabilidad de aceptar tal movimiento depende de la magnitud del incremento y la temperatura. Pequeños incrementos se aceptarán más a menudo en vez de los grandes, y la probabilidad de aceptar tales movimientos se irá reduciendo de manera gradual hasta llegar a cero, al mismo tiempo que la temperatura llegué a cero (Rozenberg et al., 2012).

Algoritmo 2 Recocido Simulado (Coello et al., 2007)

1: Seleccionar una solución inicial (s_0)
2: Seleccionar una temperatura inicial $t_0 > 0$
3: Seleccionar un mecanismo de enfriamiento CS
4: **repetir**
5: **repetir**
6: Aleatoriamente seleccionar $s \in N(s_0)$ ($N =$ vecindario)
7: $\delta = f(s) - f(s_0)$ (f es la función objetivo)
8: **si** $\delta < 0$ **entonces**
9: $s_0 \leftarrow s$
10: **si no**
11: Generar aleatoriamente x (con una distribución uniforme en el intervalo (0,1))
12: **si** $x < \exp\left(\frac{-\delta}{t}\right)$ **entonces**
13: $s_0 \leftarrow s$
14: **fin si**
15: **fin si**
16: **hasta que** el máximo número de iteraciones ITER sea alcanzado
17: $t \leftarrow CS(t)$
18: **hasta que** la condición de paro se cumple

Para diseñar un algoritmo de recocido simulado se deben tomar decisiones que podemos clasificarlas en dos categorías: las relacionadas al proceso de enfriamiento (decisiones genéricas) y las involucradas en el modelado del problema (decisiones específicas). Las genéricas incluyen la temperatura inicial, el perfil de enfriamiento y la condición de paro. Las específicas son la definición del espacio de soluciones, la función de evaluación y la generación del vecindario.

Rozenberg et al. (2012) mencionan que algunos resultados empíricos muestran que dichas decisiones se necesitan elegir con cuidado, ya que pueden influir en la velocidad del algoritmo y en la calidad de las soluciones obtenidas, mientras que los resultados teóricos muestran que bajo las condiciones correctas se garantiza la convergencia asintótica a una solución óptima. Por esta razón, no existe un conjunto de decisiones genéricas, sino que la teoría provee algunos factores importantes que deberían considerarse en los dos tipos de decisiones.

2.7.2. Temperatura inicial

Si la solución final es independiente de la inicial, la temperatura inicial debe ser lo suficientemente alta para permitir movimientos libres a través del espacio de búsqueda. En algunos casos, ésta puede ser estimada con los datos del problema, de manera que, si se calcula la mediana o el máximo cambio de la función de evaluación sobre todos los movimientos en el vecindario, este valor puede ser utilizado para estimar la temperatura en la que dichos movimientos se aceptarán con una probabilidad razonable. Para ello, se utiliza la Ecuación 11, donde U es la mediana de los incrementos de la función de evaluación de todos los movimientos en el vecindario y R_{acc} es la proporción de aceptación (Rozenberg et al., 2012).

$$t_0 = \left(\frac{-U}{\ln R_{acc}} \right). \tag{11}$$

El algoritmo de recocido simulado resulta altamente afectado por la proporción a la que se reduce la temperatura, la cual está dada por una función α y por el número de iteraciones (Rozenberg et al., 2012). A continuación se describen los esquemas de enfriamiento más populares en el estado del arte para reducir el valor de la temperatura t.

2.7.3. Enfriamiento lineal

El enfriamiento lineal es uno de los esquemas más simples y está definido por la Ecuación 12 (Simon, 2013), donde t_0 es la temperatura inicial, k es el número de iteración del algoritmo de recocido simulado y η es una constante. Se debe cumplir que $t > 0$ para todo valor de k, por lo que debemos seleccionar el valor de η para que la temperatura en el número máximo de iteraciones sea positiva. La Figura 16 muestra un enfriamiento lineal para un valor de temperatura de 100 y 200 iteraciones.

$$\alpha(t_0) = t_0 - \eta k. \tag{12}$$

2.7.4. Enfriamiento exponencial

Este perfil de enfriamiento se define por la Ecuación 13 (Simon, 2013), donde t_0 es la temperatura inicial y típicamente $a \in$ (0.8,1). Un valor grande de a resultará en un perfil de enfriamiento lento. La Figura 17 muestra el comportamiento de un enfriamiento exponencial para una temperatura inicial de 100, para tres valores de a. Como se puede observar, entre más cercano a uno sea el valor de a, el enfriamiento será más

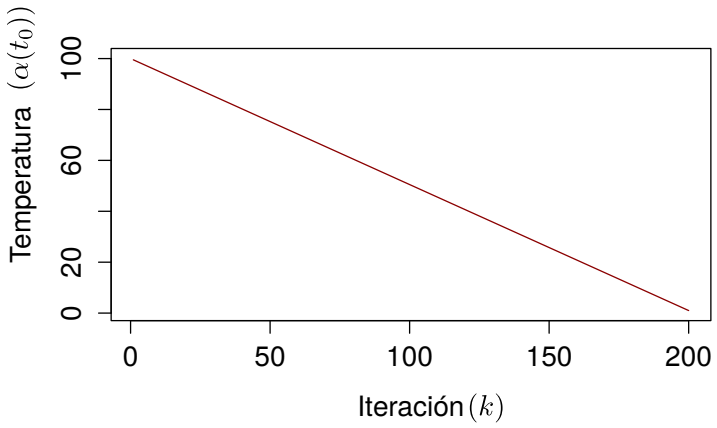


Figura 16. Enfriamiento lineal para un valor de temperatura inicial de 100 y 200 iteraciones.

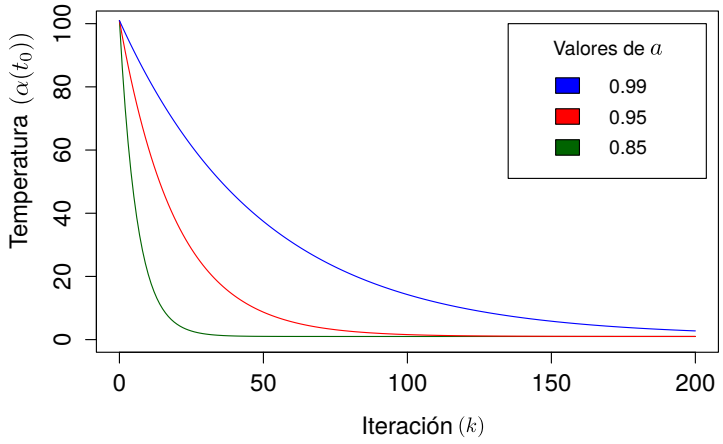


Figura 17. Enfriamiento exponencial para un valor de temperatura inicial de 100 y 200 iteraciones para diferentes valores de a.

lento.

$$\alpha(t_0) = at_0. \tag{13}$$

2.7.5. Enfriamiento hiperbólico

Este perfil de enfriamiento se basa en el comportamiento descrito por la Ecuación 14 (Lukasiewycz, Glaß, Reimann y Teich, 2011), donde t_0 es la temperatura inicial, k es el número de iteración del algoritmo, t_f es

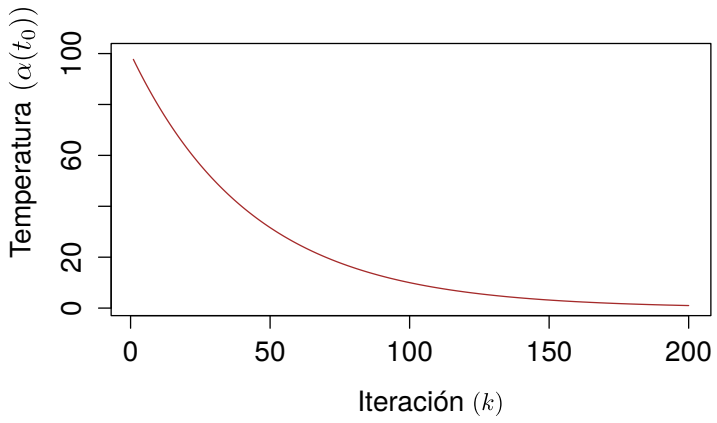


Figura 18. Enfriamiento hiperbólico para una temperatura inicial de 100 y 200 iteraciones.

la temperatura final y n el número máximo de iteraciones (Figura 18).

$$\alpha(t_0) = t_0 \left(\frac{t_f}{t_0}\right)^{\frac{k}{n}}. \tag{14}$$

2.7.6. Punto de enfriamiento o condición de paro

En teoría, la temperatura debe ser reducida a cero cuando no haya más movimientos de mejora posibles. En la práctica, un punto de enfriamiento es alcanzado antes de llegar a este valor. Si el objetivo es encontrar una solución a menos de ε de un óptimo local con probabilidad θ, entonces la condición de paro debe satisfacer la Ecuación 15 (Rozenberg et al., 2012), donde t_f es la temperatura final y $|S|$ es el tamaño del espacio de búsqueda. En otro caso, la condición de paro es alcanzada después de un número predefinido de iteraciones.

$$t_f \leq \frac{\varepsilon}{\ln \frac{|S|-1}{\theta}}. \tag{15}$$

2.7.7. Generación del vecindario de soluciones

Uno de los métodos más sencillos para generar soluciones vecinas consiste en seleccionar de manera aleatoria un punto del espacio de búsqueda. Sin embargo, una vez que el algoritmo de recocido simulado comienza a converger, es deseable que la solución actual s_0 sea la mejor de todo el espacio de búsqueda. Por lo que, seleccionar aleatoriamente una solución vecina probablemente no sea muy efectivo. De manera general, la generación de una solución vecina debe tender hacia la solución actual Simon (2013).

2.8. Herramienta ISU TAS

El software ISU TAS (*Iowa State University Tile Assembly Simulator*) es un simulador y editor de sistemas de ensamblado de mosaicos aTAM (Patitz, 2011). Permite a los usuarios diseñar conjuntos de mosaicos y semillas para simular los ensambles. La interfaz gráfica del simulador permite visualizar el crecimiento de un ensamble partiendo de una semilla. Se puede avanzar, retroceder, realizar acercamientos, desplazamientos, etc., al ensamble que se está simulando.

El editor gráfico de tipos de mosaicos permite especificar los valores y fuerzas de pegado, y asignarle un color. Los conjuntos de mosaicos diseñados se guardan en un archivo de texto y se pueden volver a cargar para realizar una simulación. ISU TAS es liberado bajo una licencia pública general (GPL) para utilizar, compartir y modificar el software. El formato que utiliza ISU TAS para especificar el diseño de un mosaico aTAM consta de las siguientes instrucciones seguidas del valor de cada una de las propiedades indicadas:

TILENAME <nombre del mosaico>
LABEL <etiqueta del mosaico>
NORTHBIND <fuerza de pegado del lado norte>
EASTBIND <fuerza de pegado del lado este>
SOUTHBIND <fuerza de pegado del lado sur>
WESTBIND <fuerza de pegado del lado este>
NORTHLABEL <tipo de pegado del lado norte>
EASTLABEL <tipo de pegado del lado este>
SOUTHLABEL <tipo de pegado del lado sur>
WESTLABEL <tipo de pegado del lado oeste>
TILECOLOR <color del mosaico>
TEXTCOLOR <color del texto que aparece en el mosaico>
CREATE

Para construir un RTAS $\mathcal{T} = (T, \sigma_L)$, se especifican todos los mosaicos del conjunto T con fuerzas de pegado igual a uno. Los mosaicos que forman σ_L son los únicos que pueden tener fuerzas de pegado igual a cero, uno y dos. Por ejemplo, para crear un mosaico de color negro que pertenece al conjunto T, con tipos de pegado como aparece en la Figura 19 donde norte = 48, este = 14, sur = 14 y oeste = 48, así como fuerza de pegado igual a uno para todos sus lados. Las instrucciones en ISU TAS son las siguientes:

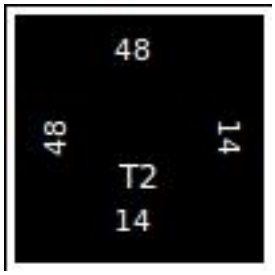


Figura 19. Mosaico en ISU TAS.

TILENAME T2

LABEL

NORTHBIND 1

EASTBIND 1

SOUTHBIND 1

WESTBIND 1

NORTHLABEL 48

EASTLABEL 14

SOUTHLABEL 14

WESTLABEL 48

TILECOLOR black

TEXTCOLOR white

CREATE

Capítulo 3. Diseño del algoritmo de recocido simulado

En este capítulo se describe las representaciones utilizadas para resolver el problema PATS, la construcción de la solución inicial y las operaciones diseñadas para la generación del vecindario de soluciones.

3.1. Representación basada en grafos de mosaicos

En este modelo, para representar un ensamble de mosaicos se construyen dos grafos llamados grafo de mosaicos (G_T) y grafo de pegado (G_B). A continuación se describe cómo se realiza esta construcción. Se supone que cada mosaico del patrón es blanco o negro, y que el auto-ensamblado sucede de este-sur a norte-oeste.

El grafo $G_T = (V_T, E_T)$ está formado por un conjunto de vértices V_T y un conjunto de aristas E_T. Un vértice en V_T puede ser etiquetado como (B_e, B_s) o (B_o, B_n), donde B_e, B_s, B_o y B_n son los tipos de pegado de los lados este, sur, oeste y norte, respectivamente. Cada tipo de mosaico es representado por una arista dirigida en E_T, la cual conecta dos vértices en V_T, con etiquetas (B_e, B_s) y (B_o, B_n).

El grafo $G_B = (X_B, Y_B, E_B)$ es un grafo bipartito no dirigido, en el que cada tipo de pegado es representado por un vértice. Por cada vértice (x, y) en el grafo G_T, existe una arista en E_B que conecta al vértice que representa el tipo de pegado horizontal $x \in X_B$ con el vértice que representa el pegado vertical $y \in Y_B$. E_B es un conjunto de aristas coloreadas. Para una arista (x', y') en el grafo de pegado, si existe un mosaico con el lado este de tipo x' y el lado sur de tipo y', entonces el color del mosaico (negro o blanco) es asignado a esta arista.

Por ejemplo, en la Figura 20A) se puede observar un tipo de mosaico con sus respectivos valores de pegado. El mosaico se representa con dos vértices en el grafo G_T (Figura 20B)), el primero etiquetado con (e, s) y el segundo con (o, n). La arista se dirige del vértice (e, s) a (o, n). En el grafo de pegado (Figura 20C)), existen cuatro vértices etiquetados con los valores e, s, n y o (los tipos de pegado) y dos aristas. La arista que va de e a s está etiquetada con el color del mosaico (negro). La arista que va de o a n permanece sin etiqueta. La Figura 21 muestra un patrón de tamaño 2×2, cuyo correspondiente grafo de mosaicos consta de cuatro vértices dirigidos. El grafo de pegado se encuentra formado por 12 vértices y las aristas etiquetadas corresponden a los mosaicos con el lado este-sur en el grafo G_T.

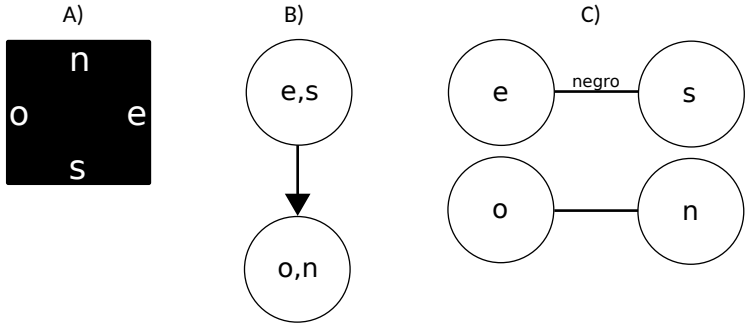


Figura 20. Representación con grafos. A) Mosaico aTAM coloreado. B) Grafo G_T y C) grafo G_B.

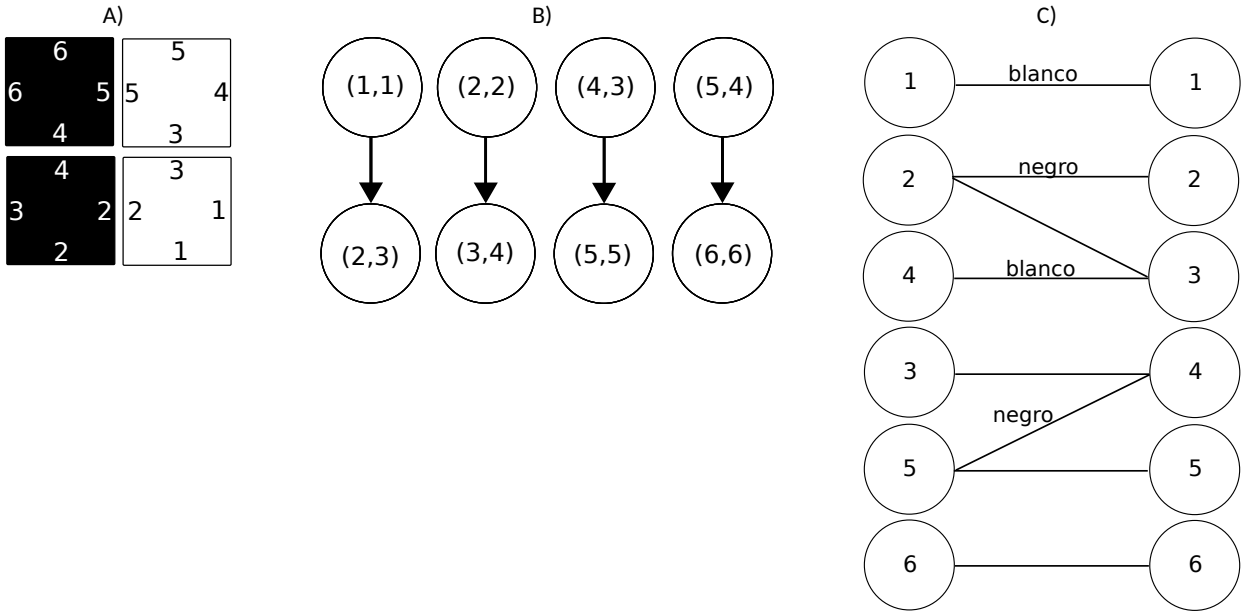


Figura 21. Grafos G_T y G_B. A) Patrón de 2×2. B) Grafo de mosaicos G_T y C) Grafo de pegado G_B.

3.1.1. Generación de la solución inicial

La solución inicial se genera con el Algoritmo 3, que recibe como entrada el patrón P (línea 1). Para cada mosaico t_i se crean dos vértices, el primero etiquetado con los valores de los lados este y sur, y el segundo etiquetado con los lados norte y oeste (línea 3). Además, se crean los vértices correspondientes en el grafo de pegado para cada vértice agregado a G_T (línea 4).

Algoritmo 3 Crear solución inicial con grafos

Entrada: Patrón P de tamaño $m \times n$
Salida: Grafos G_T y G_B

```
1: procedimiento CrearSoluciónInicialConGrafos(P)
2:     para todo mosaico t_i ∈ P hacer
3:         G_T.CrearVértice(t_i)
4:         G_B.CrearVértice(t_i)
5:     fin para
6:     regresar G_T y G_B
7: fin procedimiento
```

3.1.2. Verificando la validez de un grafo $\mathbf{G_T}$

Por definición los grafos G_T y G_B cumplen con el siguiente teorema (Ma y Lombardi, 2008):

Teorema 3.1. Un conjunto de mosaicos es válido sí, y sólo si, las siguientes condiciones se cumplen:

1. El grado de cualquier vértice en el grafo de mosaicos no es mayor que uno.
2. El color de cada arista (x, y) en el grafo de pegado está correctamente definido como blanco, negro, o sin color, de acuerdo al patrón P que será generado.

El primer punto garantiza un ensamblaje determinista. Por ejemplo, si un vértice tiene dos aristas de salida (Figura 22A), dicho grafo representaría dos mosaicos que comparten los valores de pegado este-sur (Figura 22B), lo que implica que ambos tipos de mosaicos tienen la misma posibilidad de asignarse en el mismo lugar durante el ensamble del patrón. En consecuencia, el ensamble correcto del mismo no estaría garantizado (Figura 22C y 22D).

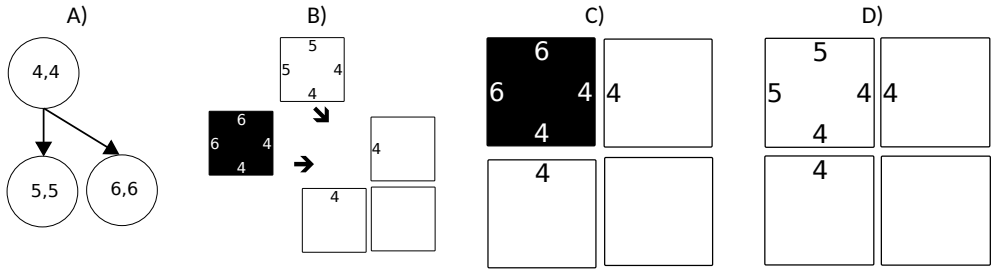


Figura 22. Representación de grafo inválido. A) Vértice con grado mayor a uno en G_T . B) Ensamble no determinista. C) Ensamble correcto. D) Ensamble incorrecto.

El Algoritmo 4 describe los pasos para verificar que el grafo G_T sea válido. Recibe como entrada el grafo G_T (línea 1). Si existe algún v_i que tenga grado de salida mayor a uno, entonces el resultado es falso (líneas 4-7). En otro caso, se devuelve verdadero (línea 9).

Algoritmo 4 Verificar que el grafo G_T es válido

Entrada: Grafo G_T
Salida: verdadero o falso

```
1:  procedimiento esGrafoVálido(G_T)
2:      válido ← verdadero
3:      para todo vértice v_i ∈ G_T hacer
4:          si gradoSalida(v_i)>1 entonces
5:              válido ← falso
6:              regresar válido
7:          fin si
8:      fin para
9:      regresar válido
10: fin procedimiento
```

3.1.3. Generación del vecindario de soluciones

Para esta representación, se diseñó una operación de reducción que permitiera combinar dos vértices pertenecientes a G_T, con el objetivo de generar vecinos con un menor número de tipos de mosaicos. El Algoritmo 5 describe la operación de reducción de vértices. El algoritmo recibe como entrada el grafo de mosaicos G_T y el grafo de pegado G_B del patrón a ensamblar (línea 1). En las líneas 2-4 se realiza una selección aleatoria de dos vértices v_1 y v_2 con grado de salida mayor a cero y del mismo color. Se asignan a v_1 todas las aristas de v_2 (líneas 5-13). Se elimina el vértice v_2 del grafo G_T y los respectivos vértices en G_B (líneas 14-16). Si el número de aristas de v_1 es igual a dos, entonces se obtienen los correspondientes vértices (v_3 y v_4) para combinarlos de la siguiente manera: se asignan todas las aristas de v_4 a v_3 y se elimina el vértice v_4, junto con sus correspondientes vértices en G_B (líneas 17-21). Finalmente, sí el grafo G_T no es válido se eliminan los cambios realizados a G_T y G_B (línea 22); en otro caso, se devuelven los grafos G_T y G_B modificados.

La Figura 23 muestra un ejemplo de reducción de vértices en los grafos G_T y G_B para un patrón de 2×2. Los correspondientes grafos de mosaicos G_T y de pegado G_B son los que se muestran en la Figura 23B). Los mosaicos seleccionados aleatoriamente son $v_1 = (1, 1)$ y $v_2 = (4, 3)$. El vértice v_2 sólo contiene una arista de salida e_1, cuyo vértice destino es $(5, 5)$, entonces se agrega una arista dirigida de v_1 a $(5, 5)$. El siguiente paso es eliminar v_2 de G_T y obtener el valor de su etiqueta, es decir $(4, 3)$. En el grafo de pegado G_B, la etiqueta (x, y) corresponde a un vértice x del lado izquierdo y a un vértice y del lado derecho. Primero, se elimina la arista entre el vértice 3 y el 4, la cual corresponde al vértice eliminado en G_T. Enseguida, se asignan las aristas del vértice 4 y 3 a los correspondientes de v_1 en G_B, es decir, al vértice 1 del lado izquierdo y al vértice 1 del lado derecho. Finalmente, se eliminan los vértices 4 y 3, dando como resultado el grafo G_T y el grafo G_B de la Figura 23D). Dado que se agregaron aristas a v_1 en G_T, su grado de salida es mayor a uno como se observa en la Figura 23E), por lo que se combinan los vértices $v_3 = (2, 1)$ y $v_4 = (5, 5)$. Lo

Algoritmo 5 Reducción de vértices en G_T y G_B

Entrada: Grafo G_T y G_B
Salida: Grafo G_T y G_B modificado

```
1:  procedimiento CombinarVértices(G_T, G_B)
2:      k ← SeleccionarDosVérticesMismoColor(G_T)
3:      v_1 ← k[1]
4:      v_2 ← k[2]
5:      para todo arista e_i de v_2 hacer
6:          si e_i es arista de entrada entonces
7:              fuente ←NodoFuente(e_i)
8:              G_T.AgregarArista(fuente, v_1)
9:          si no
10:             destino ←NodoDestino(e_i)
11:             G_T.AgregarArista(v_1, destino)
12:         fin si
13:     fin para
14:     (x, y) ← G_T.EliminarVertice(v_2)
15:     G_B.EliminarVértice(x)
16:     G_B.EliminarVértice(y)
17:     si GradoDeSalida(v_1) = 2 entonces
18:         v_3 ← ObtenerVerticeEnConflicto(v_1)
19:         v_4 ← ObtenerVerticeEnConflicto(v_1)
20:         CombinarVérticesEnConflicto(v_3, v_4)
21:     fin si
22:     si esGrafoVálido(G_T) = falso entonces
23:         EliminarCambios(G_T, G_B)
24:     fin si
25:     regresar G_T y G_B
26: fin procedimiento
```

anterior se hace para que v_1 tenga grado de salida igual a uno. Para esto, se realiza la asignación de todas las aristas de v_4 al vértice v_3. Por último, se elimina el vértice $(5, 5)$ de G_T, el vértice 5 del lado izquierdo y el vértice 5 del lado derecho de G_B (Figura 23E)). Así, ninguno de los vértices de G_T tiene grado mayor a uno, por lo que el conjunto de vértices y aristas de G_T es válido (Figura 23F)).

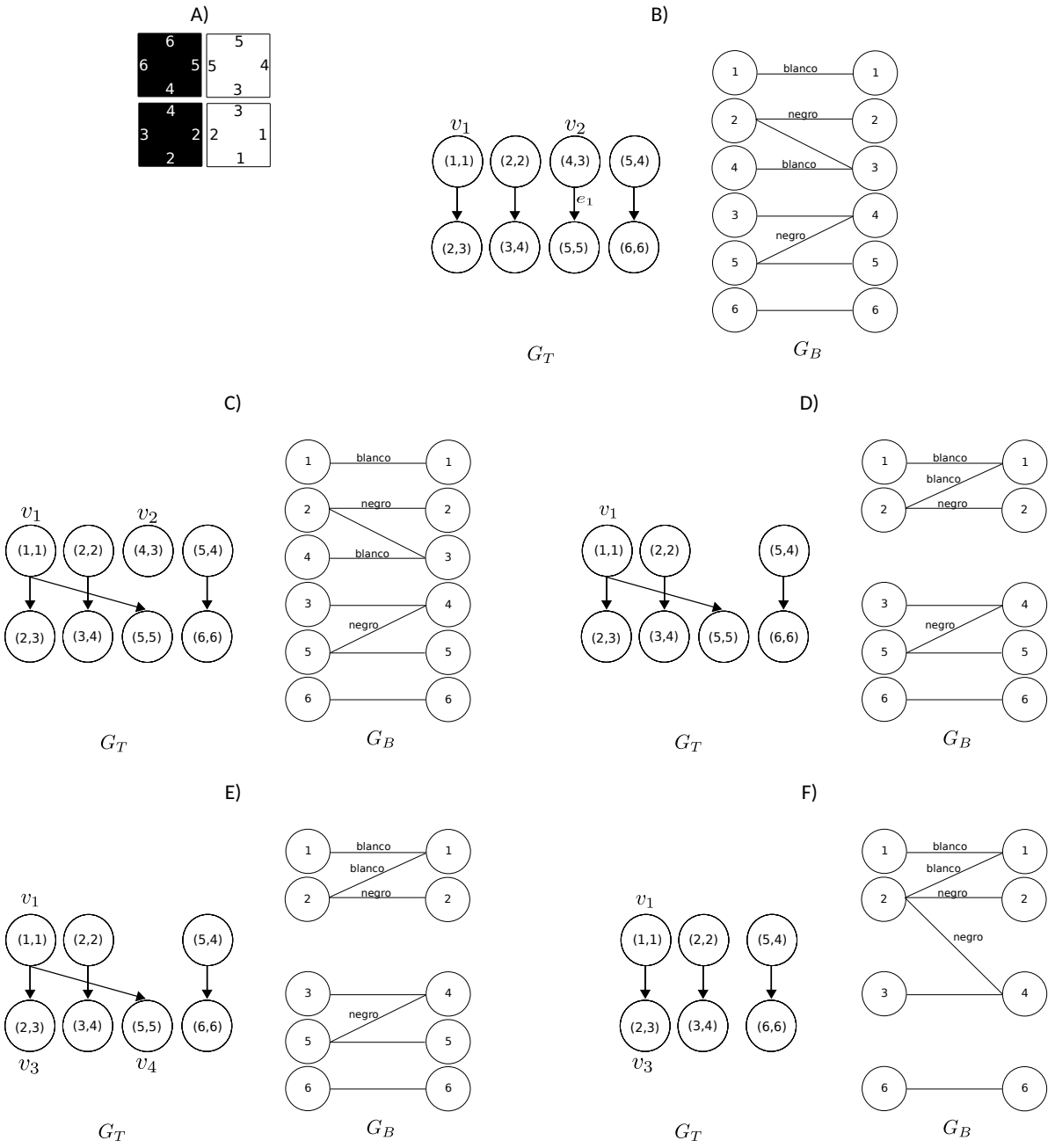


Figura 23. Ejemplo de reducción de dos vértices. A) Patrón de 2×2. B) Grafo de mosaicos G_T y de pegado G_B correspondientes. C) Asignación de arista de v_2 a v_1. D) Eliminación de v_2 en G_T y los respectivos vértices en G_B. E) Selección de vértices en conflicto v_3 y v_4. F) Resultado de la combinación de v_3 y v_4. La reducción de vértices se basa en la propuesta de Ma y Lombardi (2008), utilizada con su algoritmo voraz.

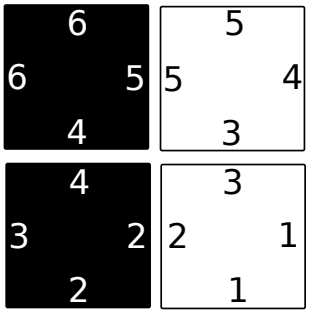


Figura 24. Patrón con mosaicos aTAM.

3.2. Representación mediante matriz de mosaicos

Además de la representación de grafos, en este trabajo se implementó la representación matricial (Lozano, 2014), en donde un patrón de dos dimensiones se modela como una matriz de tamaño $m \times n$ en la que cada elemento corresponde a un mosaico aTAM t, donde t es la tupla $t = (n, e, s, o, c)$. En esta representación se utiliza la notación $N(t_{i,j}), E(t_{i,j}), S(t_{i,j})$ y $O(t_{i,j})$, para indicar los valores norte, este, sur y oeste, respectivamente; se utiliza además $C(t_{i,j})$ para indicar el color del mosaico, el cual puede ser blanco (B) o negro (N). Formalmente, un patrón bidimensional P es una matriz de mosaicos aTAM tal que

$$P = \begin{pmatrix} t_{1,1} & \cdots & t_{1,n} \\ \vdots & \ddots & \vdots \\ t_{m,1} & \cdots & t_{m,n} \end{pmatrix}, \tag{16}$$

y cada

$$t_{i,j} = (n, e, s, o, c) \tag{17}$$

donde $n, e, s, o \in \mathbb{N}$ y $c \in \{N, B\}$. Además, cada $t_{i,j} \in P$ cumple que

$$O(t_{i,j}) = E(t_{i,j-1}) \text{ y } S(t_{i,j}) = N(t_{i+1,j}), \tag{18}$$

es decir, para cualquier mosaico $t_{i,j} \in P$, sus valores de pegado coinciden con los valores de pegado de los mosaicos adyacentes. Por ejemplo, la matriz correspondiente a los mosaicos del patrón de la Figura 24 es

$$P = \begin{pmatrix} (6,5,4,6,N) & (5,4,3,5,B) \\ (4,2,2,3,N) & (3,1,1,2,B) \end{pmatrix}. \tag{19}$$

3.2.1. Generación de la solución inicial

La solución inicial corresponde a un conjunto de mosaicos coloreados de acuerdo al patrón de entrada P y con $m \times n$ tipos de mosaicos diferentes, esto es, ninguno de los mosaicos contiene los mismos valores norte, este, sur y oeste. Para generar dicha solución existen dos alternativas: la estricta, que consiste en etiquetar cada uno de los lados de los mosaicos de tal manera que no tengan errores de coincidencia con los lados de los mosaicos adyacentes (Figura 25A); y la permisiva, que consiste en realizar una asignación de valores relajada que pueda no tener coincidencias con uno o varios de los lados de los mosaicos adyacentes (Figura 25B).

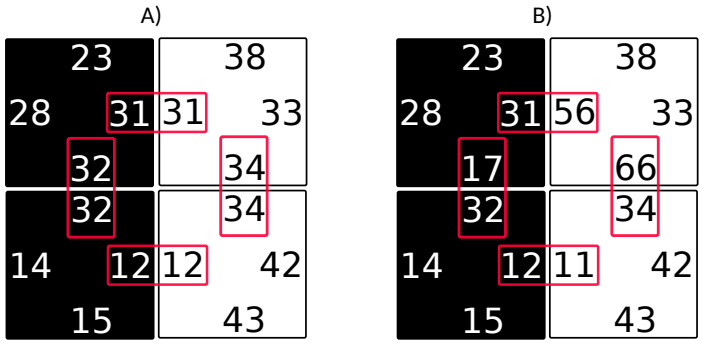


Figura 25. Formas para generar la solución inicial. A) Modo estricto. B) Modo permisivo.

En este trabajo, se utilizó el modo estricto para construir la solución inicial. El Algoritmo 6 muestra los pasos necesarios para generar dicha solución. Dado un patrón de tamaño $m \times n$, se construye una matriz M, con m filas y n columnas (línea 1). En cada una de las posiciones de la matriz se almacena un mosaico aTAM. Para todos los mosaicos de la matriz M (líneas 3 y 4), se asignan valores a cada uno de sus lados de acuerdo a un valor w (línea 5). Enseguida, se asignan al lado norte del mosaico $t_{i,j}$ el valor del lado sur de $t_{i-1,j}$ (línea 6) y al lado oeste el valor del lado este de $t_{i,j-1}$ (línea 7), incrementándose posteriormente el valor de w (línea 8). El resultado del algoritmo es la matriz M que contiene $m \times n$ mosaicos diferentes (línea 11). La Figura 26B) muestra una solución inicial para un patrón de tamaño 4×4.

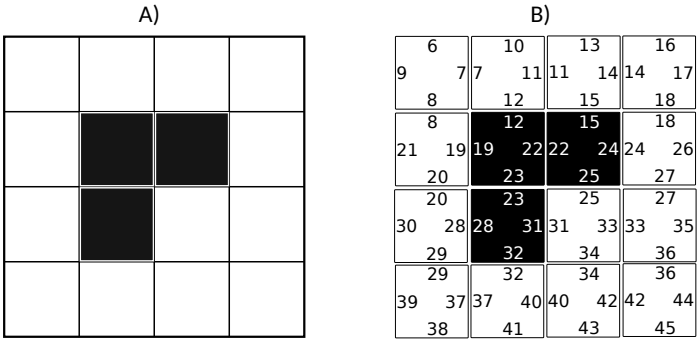


Figura 26. Construcción de una solución inicial. A) Patrón de entrada. B) Solución inicial generada.

Algoritmo 6 Construir solución inicial con matriz

Entrada: Patrón P de tamaño $m \times n$
Salida: Matriz M con $m \times n$ mosaicos diferentes que ensamblan el patrón de entrada

1: **procedimiento** CrearSoluciónInicialEnMatriz(P)
2: $M \leftarrow \begin{pmatrix} t_{1,1} & \cdots & t_{1,n} \\ \vdots & \ddots & \vdots \\ t_{m,1} & \cdots & t_{m,n} \end{pmatrix}$
3: $w \leftarrow 6$
4: **para** $i = 1$ **hasta** m **hacer**
5: **para** $j = 1$ **hasta** n **hacer**
6: $t_{i,j} = (w, w+1, w+2, w+3)$
7: $N(t_{i,j}) = S(t_{i-1,j})$
8: $O(t_{i,j}) = E(t_{i,j-1})$
9: $w = w + 4$
10: **fin para**
11: **fin para**
12: **regresar** M
13: **fin procedimiento**

3.2.2. Generación del vecindario de soluciones

Cada una de las operaciones diseñadas para generar el vecindario de soluciones toman como entrada la solución trivial obtenida con el Algoritmo 6. La exploración del espacio de búsqueda a partir de esta solución se realiza principalmente seleccionando de manera aleatoria un número de mosaicos del mismo color para después cambiar sus valores de pegado norte, este, sur y oeste.

El Algoritmo 7 se utiliza para verificar que dichos cambios en los valores de pegado de los mosaicos no genere conjuntos inválidos en cada una de las operaciones. El algoritmo recibe como entrada una matriz M (línea 1). Primero se revisan los lados oeste y sur de todos los mosaicos. Si existe algún otro mosaico de diferente tipo con los mismos lados, el resultado de la búsqueda es verdadera (líneas 2 a 6); en otro caso, la búsqueda devuelve falso (línea 7).

Algoritmo 7 Buscar lados oeste y sur repetidos

Entrada: Matriz M
Salida: verdadero o falso

```
1: procedimiento BuscarOesteSurRepetidos(M)
2:     para todo t_w, t_x ∈ M hacer
3:         si O(t_w) = O(t_x) y E(t_w) = E(t_x) y t_w ≠ t_x entonces
4:             regresar verdadero
5:         fin si
6:     fin para
7:     regresar falso
8: fin procedimiento
```

3.2.2.1. Asignación de los mismos tipos de pegado a dos mosaicos

Esta operación asigna los mismos tipos de pegado a los lados, norte, este, sur y oeste; respectivamente, de dos mosaicos. El Algoritmo 8 describe el procedimiento utilizado. Se recibe como entrada una matriz M (línea 1). Para iniciar, se seleccionan dos mosaicos t_1 y t_2 del mismo color y se almacenan en una lista (línea 2). Se obtienen los valores de pegado de t_1 y se le asignan a los mismos lados de t_2 (línea 5). Enseguida, se actualizan los valores de pegado de los mosaicos adyacentes a t_2 en la matriz M para que coincidan con los nuevos valores (línea 6). Si las modificaciones realizadas generan al menos un par de mosaicos diferentes con el mismo tipo de pegado para el oeste y sur (línea 7), entonces se deshacen los cambios hechos a la matriz (líneas 8-10). En caso contrario, se regresa la matriz modificada (línea 11). La Figura 27 muestra el resultado de la operación con dos mosaicos del mismo color.

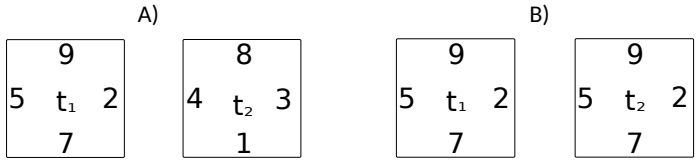


Figura 27. Operación AsignarTiposDePegadoIguales. A) Mosaicos t_1 y t_2 seleccionados. B) Asignación de los mismos valores de t_1 a t_2.

Algoritmo 8 Asignación de los mismos tipos de pegado a dos mosaicos

Entrada: Matriz M
Salida: Matriz M modificada

```
1:  procedimiento AsignarTiposDePegadoIguales(M)
2:      k ← SeleccionarDosMosaicos(M)
3:      t1 ← k[1]
4:      t2 ← k[2]
5:      t2 ← (N(t1), E(t1), S(t1), O(t1))
6:      M ←AgregarMosaicoAMatriz(t2)
7:      repetidos ← BuscarOesteSurRepetidos(M)
8:      si repetidos = verdadero entonces
9:          DeshacerCambiosAMatriz( )
10:     fin si
11:     regresar M
12: fin procedimiento
```

3.2.2.2. Asignación de tipos de pegado mínimos a dos mosaicos

Esta operación selecciona aleatoriamente de la matriz M dos mosaicos t_1 y t_2 del mismo color. Se obtiene el tipo de pegado mínimo entre dos lados con la misma orientación (norte, este, sur y oeste) y se asigna tal valor a los lados correspondientes de los mosaicos t_1 y t_2. El Algoritmo 9 describe el procedimiento utilizado para esta operación. El parámetro de entrada es una matriz M (línea 1). Primero, se seleccionan dos mosaicos (t_1 y t_2) aleatoriamente del mismo color y se almacenan en una lista (líneas 2-4). En las líneas 5-8 se obtienen los valores mínimos entre los lados norte, este, sur y oeste de ambos mosaicos, respectivamente. Dichos valores, se asignan a los lados correspondientes de t_1 y t_2 (líneas 9 y 10). Las líneas 11 y 12 agregan los mosaicos t_1 y t_2 a la matriz actualizando los tipos de pegado adyacentes de cada uno para que sean iguales a los de t_1 y t_2. Por último, se verifica que los cambios realizados en la matriz no genere lados oeste y sur repetidos; si es así, se descartan las modificaciones a t_1 y t_2 (líneas 14-17) y se regresa la matriz original. En otro caso, se regresa la matriz modificada. Por ejemplo, supongamos que $t_1 = (32, 44, 3, 6, N)$ y $t_2 = (10, 2, 25, 78, N)$ (Figura 28A)). Observe que el $\min(N(t_1), N(t_2))$, $\min(E(t_1), E(t_2))$, $\min(S(t_1), S(t_2))$ y $\min(O(t_1), O(t_2))$ es 10, 2, 3 y 6, respectivamente, asignándose dichos valores a cada uno de los lados correspondientes de t_1 y t_2. El resultado es $t_1 = (10, 2, 3, 6, N)$ y $t_2 = (10, 2, 3, 6, N)$ (Figura 28A)). Así, los lados de t_1 y t_2 con la misma orientación tienen un valor mínimo.

Algoritmo 9 Asignación de tipos de pegado mínimos a dos mosaicos

Entrada: Matriz M
Salida: Matriz M modificada

```
1:  procedimiento AsignarTiposDePegadoMínimos(M)
2:      k ← SeleccionarDosMosaicos(M)
3:      t1 ← k[1]
4:      t2 ← k[2]
5:      n ← mín (N(t1), N(t2))
6:      e ← mín (E(t1), E(t2))
7:      s ← mín (S(t1), S(t2))
8:      o ← mín (O(t1), O(t2))
9:      t1 ← (n, e, s, o)
10:     t2 ← (n, e, s, o)
11:     M ←AgregarMosaicoAMatriz(t1)
12:     M ←AgregarMosaicoAMatriz(t2)
13:     repetidos ← BuscarOesteSurRepetidos(M)
14:     si repetidos = verdadero entonces
15:         DeshacerCambiosAMatriz( )
16:     fin si
17:     regresar M
18: fin procedimiento
```

Figura 28. Operación AsignarTiposDePegadoMínimos. A) Mosaicos t_1 y t_2 seleccionados aleatoriamente. B) Los tipos de pegado de los lados con la misma orientación tienen el valor mínimo entre ellos.

3.2.2.3. Cambiar tipos de pegado de un mosaico y sus vecinos

La operación CambiarMosaicosVecinos selecciona un mosaico t_i de la matriz M y se obtienen sus mosaicos vecinos. Por cada mosaico vecino t_j, $i \neq j$, se cambian los tipos de pegado de t_i y t_j con cuatro valores seleccionados aleatoriamente entre los tipos de pegado de ambos mosaicos. En el Algoritmo 10 se muestran los pasos correspondientes. Se recibe como entrada la matriz M (línea 1). Se selecciona un mosaico t_i aleatoriamente (línea 2). Posteriormente, en la línea 3 se obtienen los vecinos del mismo color (ver Algoritmo 11) y se agregan a una lista V. Para cada uno de los mosaicos vecinos en V, se agrega a una lista Z los valores de los lados norte, este, sur y oeste del mosaico t_i y su vecino t_j (líneas 5 y 6). Se seleccionan aleatoriamente cuatro valores de Z (líneas 7-10) y se le asignan a los lados correspondientes de t_i y los del vecino t_j (líneas 11 y 12). Enseguida, se revisa que la matriz no tenga mosaicos diferentes con los mismos lados oeste y sur; si esto ocurre, entonces se descartan los cambios realizados a t_i y t_j (líneas 13-17). Se limpia la lista Z (línea

18) y se continua con los vecinos restantes en V. El resultado del algoritmo es la matriz M modificada (línea 20). La Figura 29 muestra un patrón de tamaño 4×4 al que se le aplica esta operación.

Algoritmo 10 Cambiar tipos de pegado de un mosaico y sus vecinos

Entrada: Matriz M
Salida: Matriz M modificada

```
1:  procedimiento CambiarMosaicosVecinos(M)
2:      t_i ← SeleccionarMosaico(M)
3:      V ← SeleccionarVecinosMismoColor(t_i)
4:      para todo t_j ∈ V hacer
5:          Z.AgregarValores(N(t_j), E(t_j), S(t_j), O(t_j))
6:          Z.AgregarValores(N(t_i), E(t_i), S(t_i), O(t_i))
7:          n ← Z.ObtenerElementoAleatoriamente( )
8:          e ← Z.ObtenerElementoAleatoriamente( )
9:          s ← Z.ObtenerElementoAleatoriamente( )
10:         o ← Z.ObtenerElementoAleatoriamente( )
11:         t_i.ActualizarTiposDePegado(n, e, s, o)
12:         t_j.ActualizarTiposDePegado(n, e, s, o)
13:         repetidos ← BuscarOesteSurRepetidos(M)
14:         si repetidos = verdadero entonces
15:             RestaurarValoresDePegado(t_i)
16:             RestaurarValoresDePegado(t_j)
17:         fin si
18:         Z.Limpiar( )
19:     fin para
20:     regresar M
21: fin procedimiento
```

Algoritmo 11 Seleccionar mosaicos vecinos del mismo color

Entrada: Mosaico t
Salida: Lista de mosaicos vecinos del mismo color V

```
1:  procedimiento SeleccionarVecinosMismoColor(t)
2:      (i, j) ← M.ObtenerPosicionMosaico(t)
3:      L.Agregar(M[i − 1][j])
4:      L.Agregar(M[i − 1][j − 1])
5:      L.Agregar(M[i][j − 1])
6:      L.Agregar(M[i + 1][j − 1])
7:      L.Agregar(M[i + 1][j])
8:      L.Agregar(M[i + 1][j + 1])
9:      L.Agregar(M[i][j + 1])
10:     L.Agregar(M[i − 1][j + 1])
11:     V ← L.ObtenerMosaicos(t.color)
12:     elementos ← NúmeroDeElementos(vecinos)
13:     si elementos = 0 entonces
14:         nuevo ← SeleccionarMosaicoMismoColor(t)
15:         V.Agregar(nuevo)
16:     fin si
17:     regresar V
18: fin procedimiento
```

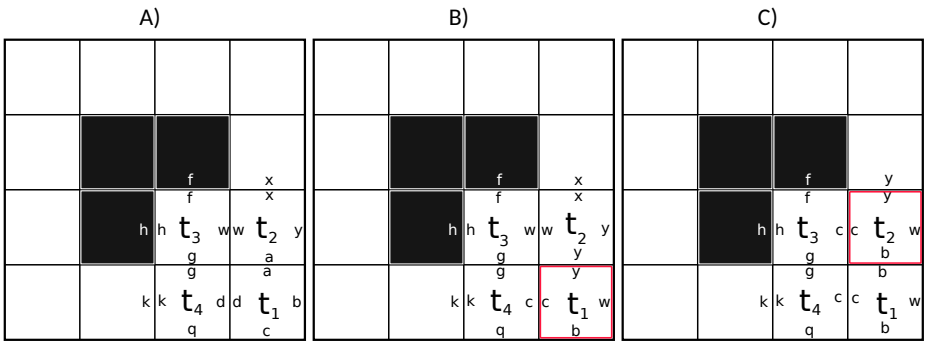


Figura 29. Una iteración de la operación CambiarMosaicosVecinos. El mosaico seleccionado es t_1 y sus vecinos son t_2, t_3, t_4 (Figura A), entonces $V = [t_2, t_3, t_4]$. Se agregan a Z los tipos de pegado del primer vecino t_2 y el mosaico t_1, así $Z = [x, y, a, w, a, b, c, d]$. Los cuatro valores seleccionados de Z son y, w, b, c. En ese orden, se asignan a los lados respectivos de t_1 (Figura B). Enseguida, se asignan los valores y, w, b, c a los lados respectivos de t_2 y se ajustan en t_1 el tipo de pegado adyacente con t_2 para que coincidan (Figura C).

3.2.2.4. Permutación de los tipos de pegado de dos mosaicos

La operación PermutarTiposdePegadoDeMosaicos selecciona aleatoriamente dos mosaicos del mismo color de la matriz M, almacenando sus tipos de pegado en una lista Z para tomar de aquí cuatro elementos que serán asignados a los lados correspondientes de t_1 y t_2. El Algoritmo 12 describe el procedimiento de dicha operación. Este procedimiento recibe como entrada la matriz M (línea 1). Primero, se seleccionan dos mosaicos t_1 y t_2 aleatoriamente de la matriz M (líneas 2-4). Después, se almacenan los valores de pegado

de t_1 y t_2 en una lista Z (líneas 5 y 6). Se realiza una permutación de los elementos de Z y se obtienen aleatoriamente cuatro de sus elementos (líneas 7-11). En las líneas 12 y 13 se actualizan los valores de pegado de t_1 y t_2 con los valores obtenidos de Z. Si t_1 y t_2 son adyacentes, la actualización de t_2 cambia el tipo de pegado del lado adyacente a t_1 haciendo que $t_1 \neq t_2$, por lo que se modifican los valores de ambos mosaicos para que tengan los mismos tipos de pegado al final de ambas actualizaciones; es decir, para que $t_1 = t_2$ (líneas 14-16). Si la actualización de t_1 y t_2 genera mosaicos diferentes con los mismos valores oeste y sur en la matriz M, se asignan los valores originales a t_1 y t_2 (líneas 17-21). En otro caso, se regresa la matriz con los nuevos valores de t_1 y t_2 (línea 22). La Figura 30 muestra la operación PermutarTiposdePegadoDeMosaicos para un patrón de tamaño 4×4, donde t_1 y t_2 corresponden a los mosaicos seleccionados aleatoriamente.

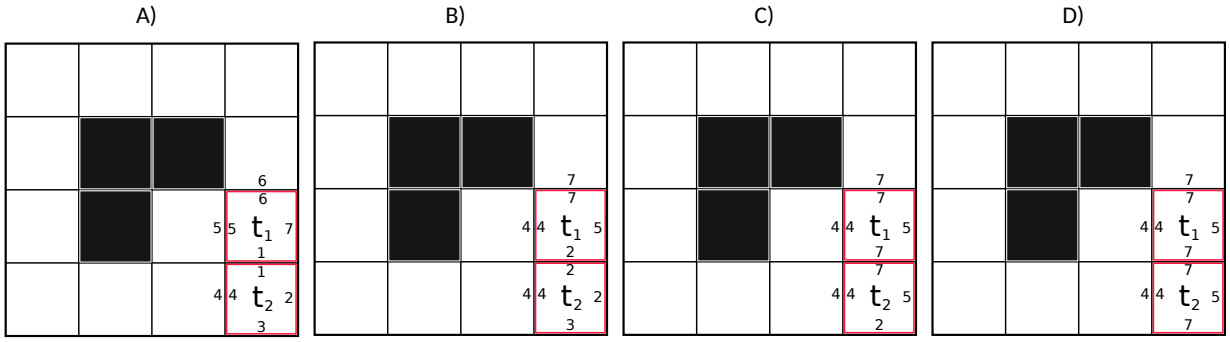


Figura 30. Operación PermutarTiposdePegadoDeMosaicos. A) Selección de dos mosaicos $t_1 = (6, 7, 1, 5, B)$, $t_2 = (1, 2, 3, 4, B)$ con los que se obtienen la lista $Z = [6, 7, 1, 5, 1, 2, 3, 4]$. Se realiza la permutación de los elementos de Z. Suponga que, los valores n, e, s y o seleccionados aleatoriamente de Z son $7, 5, 2$ y 4, respectivamente. Así, los nuevos valores de los dos mosaicos son $t_1 = (7, 5, 2, 4, B)$ y $t_2 = (7, 5, 2, 4, B)$. B) Primero se actualiza los tipos de pegado de t_1. C) Enseguida se actualizan los tipos de pegado de t_2. D) Como t_1 y t_2 son adyacentes, se cambian los valores de pegado de ambos mosaicos para que $t_1 = t_2$, entonces $t_1 = (7, 5, 7, 4, B)$ y $t_2 = (7, 5, 7, 4, B)$.

Algoritmo 12 Permutación de los tipos de pegado de dos mosaicos

Entrada: Matriz M
Salida: Matriz M modificada

```
1:  procedimiento PermutarTiposdePegadoDeMosaicos(M)
2:      k ← SeleccionarDosMosaicos(M)
3:      t1 ← k[1]
4:      t2 ← k[2]
5:      Z.AgregarValores(N(t1), E(t1), S(t1), O(t1))
6:      Z.AgregarValores(N(t2), E(t2), S(t2), O(t2))
7:      Z.PermutarElementos( )
8:      n ← Z.ObtenerElementoAleatoriamente( )
9:      e ← Z.ObtenerElementoAleatoriamente( )
10:     s ← Z.ObtenerElementoAleatoriamente( )
11:     o ← Z.ObtenerElementoAleatoriamente( )
12:     t1.ActualizarTiposDePegado(n, e, s, o)
13:     t2.ActualizarTiposDePegado(n, e, s, o)
14:     si t1 ≠ t2 entonces
15:         IgualarMosaicos(t1, t2)
16:     fin si
17:     repetidos ← BuscarOesteSurRepetidos(M)
18:     si repetidos = verdadero entonces
19:         RestaurarValoresDePegado(t1)
20:         RestaurarValoresDePegado(t2)
21:     fin si
22:     regresar M
23: fin procedimiento
```

3.2.2.5. Cambiar tipos de pegado de los mosaicos vecinos (excluyente)

La operación CambiarMosaicosVecinosExcluyente, a diferencia de la 3, solamente actualiza los tipos de pegado de cada mosaico vecino v_i con los valores almacenados en Z de los mosaicos del mosaico t_1 y v_i. Por otro lado, en la operación PermutarTiposdePegadoDeMosaicos no se obtienen los mosaicos vecinos, solamente se seleccionan dos mosaicos de manera aleatoria. La operación CambiarMosaicosVecinosExcluyente selecciona aleatoriamente un mosaico t_1 de la matriz M. Por cada vecino v_i adyacente del mismo color que t_1, se almacenan los tipos de pegado de ambos mosaicos en una lista, enseguida se seleccionan cuatro elementos y se le asignan a los tipos de pegado del vecino v_1. El Algoritmo 13 describe esta operación. Se recibe como entrada una matriz M (línea 1) y se selecciona un mosaico t_1 (línea 2). Después, se obtienen los mosaicos vecinos del mismo color de t_1 y se almacenan en una lista V (línea 3). Para cada uno de los vecinos v_i se obtienen sus tipos de pegado y se agregan a una lista Z (línea 5). Además, se agregan los valores de los lados de t_1 a la lista Z (línea 6). Posteriormente, se seleccionan cuatro valores de manera aleatoria de Z y se cambian los valores de pegado de v_i de acuerdo a las líneas 7-11. En caso de que se generen mosaicos distintos con lados oeste y sur iguales se eliminan las modificaciones realizadas a v_i (líneas 12-16). Se limpia

el contenido de la lista Z (línea 16) para realizar el mismo procedimiento para todos los vecinos en la lista V y se regresa la matriz M modificada (línea 18). La Figura 31 muestra una iteración de la operación CambiarMosaicosVecinosExcluyente con un patrón de tamaño 4×4. Para cada vecino seleccionado se realizan los mismos pasos del ejemplo.

Algoritmo 13 Cambiar tipos de pegado de los mosaicos vecinos (excluyente)

Entrada: Matriz M
Salida: Matriz M modificada

1: **procedimiento** CambiarMosaicosVecinosExcluyente(M)
2: $t_1 \leftarrow$ SeleccionarMosaico(M)
3: V $\leftarrow$ SeleccionarVecinosMismoColor(t_1)
4: **para todo** $v_i \in V$ **hacer**
5: Z.Agregar($N(v_i), E(v_i), S(v_i), O(v_i)$)
6: Z.Agregar($N(t_1), E(t_1), S(t_1), O(t_1)$)
7: $n \leftarrow$ Z.ObtenerElementoAleatoriamente( )
8: $e \leftarrow$ Z.ObtenerElementoAleatoriamente( )
9: $s \leftarrow$ Z.ObtenerElementoAleatoriamente( )
10: $o \leftarrow$ Z.ObtenerElementoAleatoriamente( )
11: v_i.ActualizarTiposDePegado(n, e, s, o)
12: repetidos $\leftarrow$ BuscarOesteSurRepetidos(M)
13: **si** repetidos = verdadero **entonces**
14: RestaurarValoresDePegado(v_i)
15: **fin si**
16: Z.Limpiar( )
17: **fin para**
18: **regresar** M
19: **fin procedimiento**

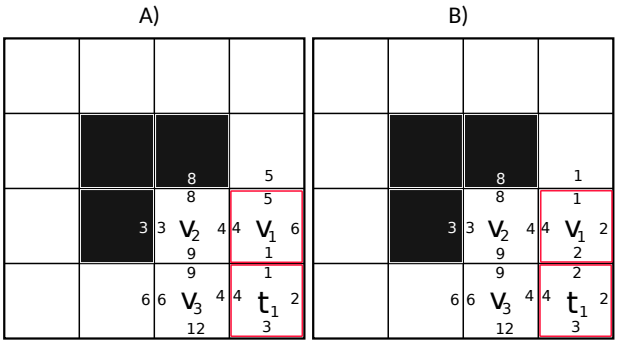


Figura 31. Una iteración de la operación CambiarMosaicosVecinosExcluyente. A) Selección de vecinos de t_1. Así, $V = [v_1, v_2, v_3]$. Se agregan los tipos de pegado de v_1 y de t_1 a Z, entonces $Z = [5, 6, 1, 4, 1, 2, 3, 4]$. Suponga que los valores n, e, s, o obtenidos aleatoriamente de Z son $1, 2, 2$ y 4, respectivamente. B) Después, se actualizan los tipos de pegado de v_1 para que $v_1 = (1, 2, 2, 4, B)$.

3.3. Implementación del algoritmo

3.3.1. Plataforma de desarrollo OPT4J

El sistema OPT4J es una plataforma de desarrollo para resolver problemas de optimización con algoritmos meta-heurísticos (Lukasiewycz et al., 2011). Fue creado por el Departamento de Ciencias de la Computación de la Universidad de Erlanger-Nurember, Alemania. Contiene algoritmos de optimización, tales como algoritmos evolutivos multi-objetivo (SPEA y NSGA2), evolución diferencial, optimización por cúmulo de partículas (PSO) y recocido simulado. La principal ventaja de esta plataforma es su programación modular, lo cual permite que el problema pueda simplificarse en diferentes módulos para después utilizar cualquiera de los algoritmos mencionados.

En OPT4J se utiliza una representación genética para especificar el problema que se quiere resolver, es decir, por medio de un individuo. Cada individuo se compone de: un genotipo que representa la codificación del problema, un fenotipo que es una solución al problema, un decodificador para traducir un genotipo a un fenotipo, y un evaluador que recibe el fenotipo del individuo y lo evalúa con las funciones objetivo (Figura 32). Para utilizar el algoritmo de recocido simulado en OPT4J, se implementó el genotipo, el fenotipo, la función de evaluación y las operaciones para generar las soluciones vecinas con la representación con grafos y matricial.

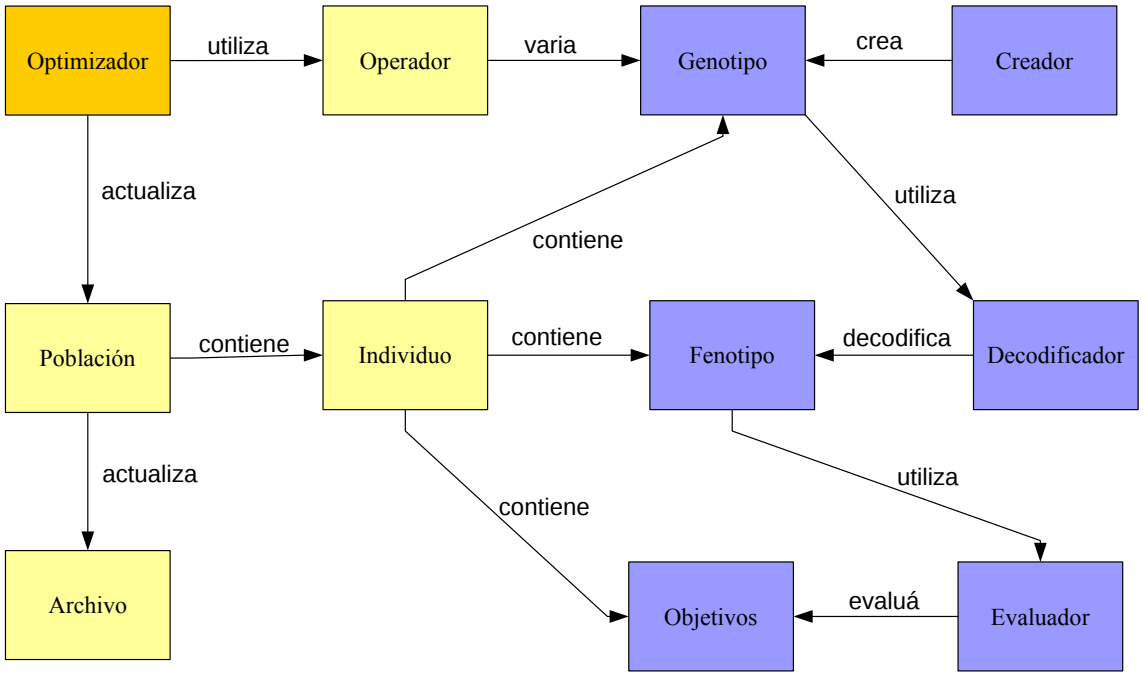


Figura 32. Arquitectura básica de OPT4J. Una tarea de optimización se define implementando las interfaces `Creador`, `Decodificador` y `Evaluador`. Un algoritmo de optimización se implementa con la interfaz `Optimizador`. Adaptado de (Lukasiewycz et al., 2011).

3.3.1.1. Algoritmo para la representación con grafos

El Algoritmo 14 muestra la implementación realizada para la representación con grafos. Recibe como parámetros de entrada un patrón P, el número de iteraciones i, la temperatura inicial t_i, la temperatura final t_f y el perfil de enfriamiento CS (línea 1). El procedimiento de la línea 2 crea la solución inicial y devuelve los grafos G_T y G_B. Posteriormente, se inicializa la temperatura actual t con el valor de la temperatura final t_f (línea 3). La variable n indica la iteración actual del algoritmo (línea 4). Se aplica la operación vecindario a los grafos G_T y G_B (línea 8) y se almacena en s. Si la nueva solución s es mejor que la actual s_0, entonces se toma a s como solución actual (líneas 9-11). En caso contrario, se selecciona a s con un criterio probabilístico (líneas 12-16). El valor de la iteración actual se incrementa en una unidad (línea 17) y se decrementa el valor de la temperatura actual t de acuerdo al perfil de enfriamiento seleccionado (línea 19). El algoritmo se repite hasta que el número de iteraciones n sea mayor o igual a i (línea 21) y el resultado es un conjunto de mosaicos Z (línea 21).

Algoritmo 14 Recocido simulado con la representación de grafos

Entrada: Patrón P, iteraciones i, temperatura inicial t_0, temperatura final t_f, perfil de enfriamiento CS.
Salida: Conjunto de mosaicos Z.

```
1:  procedimiento RecocidoSimulado(P, i, t_0, t_f, CS)
2:      s_0 ← CrearSoluciónInicialConGrafos(P)
3:      t ← t_f
4:      n ← 0
5:      repetir
6:          G_T ← s_0[1]
7:          G_B ← s_0[2]
8:          s ← CombinarVértices(G_T, G_B)                ▷ operación vecindario
9:          δ ← f(s) − f(s_0)
10:         si δ < 0 entonces
11:             s_0 ← s
12:         si no
13:             x ←GenerarNúmeroAleatorio(0,1)
14:             si x < exp(−δ/t) entonces
15:                 s_0 ← s
16:             fin si
17:         fin si
18:         n ← n + 1
19:         t ← CS(t_i, t_f, t, n, i)
20:         regresar Z
21:     hasta que n < i
22: fin procedimiento
```

3.3.1.2. Algoritmo para la representación matricial

El Algoritmo 15 describe la implementación del recocido simulado con la representación matricial. Dicho procedimiento es similar al Algoritmo 14, excepto por la línea 2 en donde se construye la solución inicial del patrón P y se devuelve una matriz de mosaicos aTAM como solución actual s_0. Además, en la línea 6 la operación vecindario corresponde a alguna de las diseñadas para la representación matricial, en las que el parámetro de entrada es una matriz de mosaicos s_0 y se devuelve una matriz de mosaicos modificada s_0. El resto de las instrucciones son iguales a las líneas 9-22 del Algoritmo 14.

Algoritmo 15 Recocido simulado con la representación matricial

Entrada: Patrón P, iteraciones i, temperatura inicial t_0, temperatura final t_f, perfil de enfriamiento CS.
Salida: Conjunto de mosaicos Z.

```
1:  procedimiento RecocidoSimulado(P, i, t_0, t_f, CS)
2:      s_0 ← CrearSoluciónInicialEnMatriz(P)
3:      t ← t_f
4:      n ← 0
5:      repetir
6:          s ← CambiarMosaicosVecinosExcluyente(s_0)        ▷ operación vecindario
7:          δ ← f(s) − f(s_0)
8:          si δ < 0 entonces
9:              s_0 ← s
10:         si no
11:             x ←GenerarNúmeroAleatorio(0,1)
12:             si x < exp(−δ/t) entonces
13:                 s_0 ← s
14:             fin si
15:         fin si
16:         n ← n + 1
17:         t ← CS(t_i, t_f, t, n, i)
18:     hasta que n < i
19:     regresar Z
20: fin procedimiento
```

Capítulo 4. Experimentos y resultados

En este capítulo se presentan las especificaciones computacionales donde se realizaron las ejecuciones del algoritmo de recocido simulado, así como las características de cada uno de los casos de prueba. Se describen los resultados obtenidos en los experimentos realizados con las diferentes configuraciones de parámetros del algoritmo. Finalmente, se muestran las dos configuraciones de parámetros con las que se obtuvieron los mejores resultados para todos los patrones de prueba y se realiza la comparación con los resultados del estado del arte.

4.1. Especificaciones computacionales

El equipo de cómputo utilizado para la ejecución del algoritmo fue una iMac con un procesador Intel Core i5, con 2.7 GHz., memoria RAM DDR3 de 4 Gigabytes, con capacidad de almacenamiento de 1 Terabyte. De igual manera se utilizó un servidor Mac Pro con procesador Intel Xeon Dual Core, 2.6 GHz., memoria RAM de 64 Gigabytes y disco duro de 1 Terabyte de almacenamiento.

4.2. Casos de prueba

El conjunto de patrones de prueba con los que se realizaron los experimentos fueron tomados del estado del arte del problema PATS (Tabla 4). Todos los patrones son de dos dimensiones y los mosaicos pueden ser de color blanco o negro. El conjunto incluye cinco patrones principales: el contador binario, el tablero de ajedrez, la esquina de árbol, el patrón aleatorio y el triángulo de Sierpinski. Así mismo, cada patrón cuenta con casos de diferentes tamaños. A continuación se describen los 14 casos de prueba utilizados en los experimentos con el algoritmo propuesto.

4.2.1. Patrón contador binario

El patrón de contador binario fue introducido por Ma y Lombardi (2008), y simula un contador en base dos. El mosaico más a la izquierda representa el bit menos significativo. Los mosaicos negros representan el valor 1 y los blancos al valor 0. Por ejemplo, el valor binario 001 se representa con mosaicos B, B, N en el patrón. El tamaño del patrón determina cuál es la cuenta máxima, por lo que para una instancia de 15×15, el contador binario llegaría hasta el valor 15 (Figura 33). Hay tres casos para este patrón de prueba: 6×6,

15×15 y 32×32.

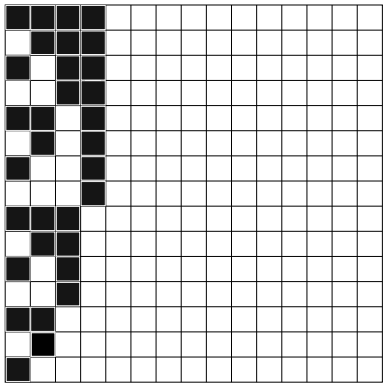


Figura 33. Contador binario de 15×15.

4.2.2. Patrón esquina de árbol

El patrón esquina de árbol (Figura 34) fue propuesto por Lempiäinen et al. (2011). Este patrón solamente incluye un caso de prueba de tamaño 23×23.

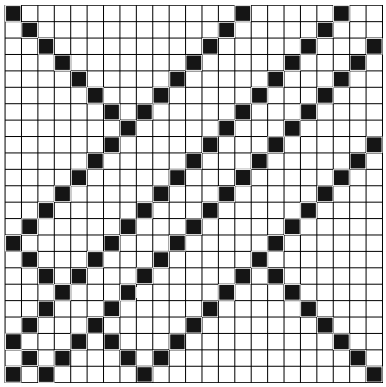


Figura 34. Esquina de árbol de 23×23.

4.2.3. Patrón tablero de ajedrez

El patrón tablero de ajedrez es uno de los casos de prueba más conocidos, el cual también fue propuesto por Ma y Lombardi (2008). Por razones obvias, el tamaño de este patrón siempre es $n \times n$. El conjunto de pruebas incluye casos de 6×6 y 15×15 (Figura 35).

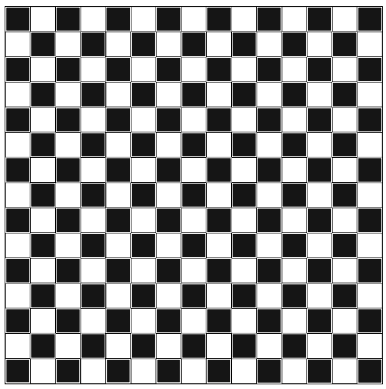


Figura 35. Tablero de ajedrez de 15×15.

4.2.4. Patrón aleatorio

El patrón aleatorio fue introducido por Göös y Orponen (2011). Para generarlo, se toma una matriz conformada solamente con mosaicos de color blanco, seleccionando de manera aleatoria mosaicos para cambiarles el color a negro. La Figura 36, muestra el patrón utilizado por Lozano (2014) como caso de prueba (caso de 16×16). El conjunto de prueba incluye tamaños de 12×12 y 16×16.

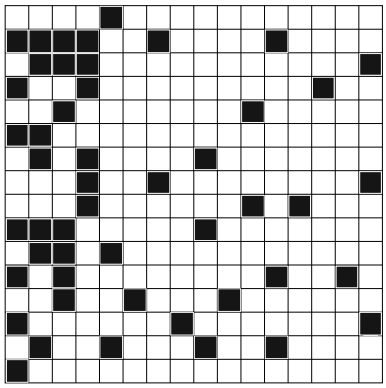


Figura 36. Patrón aleatorio de 16×16.

4.2.5. Patrón triángulo de Sierpinski

El patrón triángulo de Sierpinski es el más utilizado en el estado del arte del auto-ensamblado de ADN. Rothemund et al. (2004) construyeron este patrón a través de un autómata celular y la operación binaria XOR; con los resultados de esta investigación se realizó la construcción experimental en laboratorio. Ma y Lombardi (2008) incluyen casos de 6×6, 8×8, 12×12, 17×17 y 32×32. Lempiäinen et al. (2011) incluye el caso de 64×64 (Figura 37), siendo esta la de mayor tamaño de todos los del conjunto de prueba.

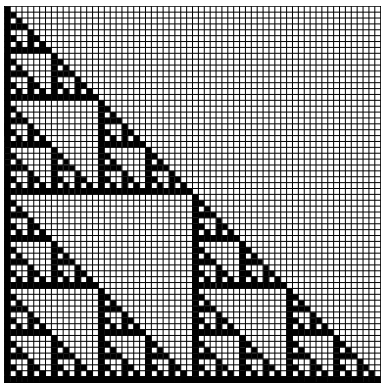


Figura 37. Patrón triángulo de Sierpinski de 64×64.

4.3. Formato de los patrones a ensamblar

La especificación del patrón P de tamaño $n \times m$ se realiza en un archivo de texto en formato CSV, en donde cada mosaico es representado con un color blanco o negro (B o N, respectivamente) y están separados por comas. Por ejemplo, para el patrón de la Figura 38A), el formato del archivo CSV correspondientes es como aparece en la Figura 38B). Así, para todos los patrones de prueba mencionados anteriormente se especifican de esta manera.

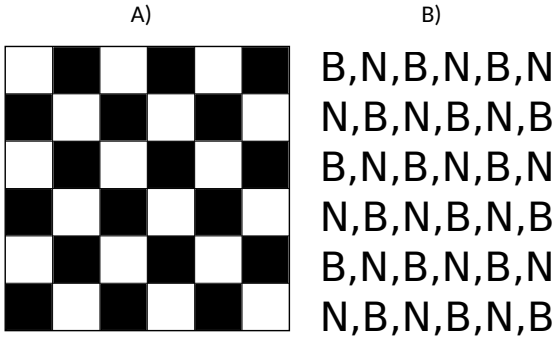


Figura 38. Especificación de un patrón. A) Patrón a ensamblar. B) Especificación del patrón de acuerdo a los colores blanco o negro de los mosaicos (B o N, respectivamente), separados por comas.

4.4. Experimentos y resultados

En el algoritmo de recocido simulado, los parámetros de configuración son: el número de iteraciones, la temperatura inicial y final, el perfil de enfriamiento y la operación que genera las soluciones vecinas. Para cada configuración de parámetros se muestran los resultados generados por el algoritmo y se realiza una comparación con el estado del arte del problema.

Tabla 4. Notación para los patrones de prueba.

Identificador	Patrón
P_1	Contador binario 15×15
P_2	Contador binario 32×32
P_3	Contador binario 6×6
P_4	Esquina de árbol 23×23
P_5	Aleatorio 12×12
P_6	Aleatorio 16×16
P_7	Triángulo de Sierpinski 12×12
P_8	Triángulo de Sierpinski 17×17
P_9	Triángulo de Sierpinski 32×32
P_{10}	Triángulo de Sierpinski 64×64
P_{11}	Triángulo de Sierpinski 6×6
P_{12}	Triángulo de Sierpinski 8×8
P_{13}	Tablero de ajedrez 15×15
P_{14}	Tablero de ajedrez 6×6

4.4.1. Representación basada en grafos de mosaicos

Los experimentos con la representación de grafos consistieron en ejecutar el algoritmo 10 veces con los parámetros de configuración de la Tabla 5 para todos los patrones de prueba (Tabla 4). Los resultados obtenidos se muestran en la Tabla 6. Se puede observar que con los casos del patrón tablero de ajedrez se obtienen los óptimos. Además, para los casos del triángulo de Sierpinski de 32×32, 12×12 y 8×8, solamente se mejora el desempeño obtenido por el algoritmo voraz. Sin embargo, para el patrón triangulo de Sierpinski de 64×64, el resultado de la minimización no fue tan significativa con respecto al conjunto de mosaicos de la solución inicial.

Tabla 5. Parámetros de configuración del algoritmo para la representación con grafos.

Parámetro	Valor
Número de iteraciones	1000000
Temperatura inicial	100000
Temperatura final	1
Perfil de enfriamiento	Exponencial con $\alpha = 0.85$
Operación	Reducción del grafo G_T

Tabla 6. Comparación de los resultados obtenidos con la representación grafos de mosaicos (SA-GRAFOS) con el estado del arte: [1] (Ma y Lombardi, 2008), [2] (Göös y Orponen, 2011), [3] Lempiäinen (2015). F* es el valor del óptimo.

Patrón	Tamaño	Voraz [1]	PS-BB [2]	PS-H [4]	SA-GRAFOS	F*
Contador binario	6×6	19	–	–	17	4
	15×15	121	–	–	121	4
	32×32	–	307	20	680	4
Esquina de árbol	23×23	–	192	25	279	–
Triángulo de Sierpinski	6×6	21	–	–	21	4
	8×8	37	–	–	31	4
	12×12	80	20	–	72	4
	17×17	172	23	–	144	4
	32×32	595	42	4	517	4
	64×64	–	95	4	3029	4
Tablero de ajedrez	6×6	19	–	–	2	2
	15×15	119	–	–	2	2
Aleatorio	12×12	–	–	–	120	–
	16×16	–	–	–	200	–

Durante los experimentos con esta representación, se observaron algunas desventajas relacionadas con la generación de las soluciones vecinas. Recuerde que en el grafo de mosaicos G_T se representan los diferentes tipos de mosaicos que tiene un patrón y que en el grafo de pegado G_B se almacena el color de los mosaicos, indicando además los tipos de pegado que tienen en común cada uno de ellos. Al aplicar la operación de reducción de vértices en G_T, la cardinalidad del conjunto de tipos de mosaicos siempre disminuye en por lo menos una unidad, lo que causaba que la solución vecina generada siempre era mejor o igual que la solución actual y que en el vecindario de soluciones no existieran soluciones peores.

Generar una solución peor que la solución actual en el grafo G_T implica aumentar el número de vértices o aristas. Las etiquetas correspondientes de cada vértice (valores para los tipos de pegado), no deben ser arbitrarios; es decir, los elementos que se agreguen a G_T deben estar relacionados con los distintos mosaicos y fuerzas de pegado que ya existen en el grafo, por lo que se necesita conocer los valores de pegado específicos que se pueden cambiar para aumentar el número de vértices o aristas. Sin embargo, lo anterior es imposible con esta representación ya que con los grafos G_T y G_B solamente se conocen los tipos de mosaicos y los valores de pegado que tienen en común, pero no así su ubicación dentro del patrón. Es por ello que se necesita una estructura adicional de datos para representar al patrón formado por G_T y G_B. Con base en la propuesta de (Lozano, 2014), se decidió cambiar la representación del problema a una estructura de datos matricial y diseñar las operaciones para el vecindario tomando en cuenta las observaciones realizadas.

4.4.2. Representación matricial

Para evaluar el desempeño de esta representación, se realizó una serie de experimentos que consistió en ejecutar el algoritmo con diferentes combinaciones de los parámetros mostrados en la Tabla 7. En total se probaron 30 ($2 \times 3 \times 5$) configuraciones de parámetros (Tabla 8). Además, cada configuración se ejecutó 10 veces con cada uno de los casos de los patrones de prueba, haciendo un total de 4,200 experimentos. El promedio y la desviación estándar de los resultados de cada configuración también fueron calculados. Para realizar una comparación justa entre las diferentes configuraciones, se normalizaron los promedios sobre el intervalo $[0, 1]$, siendo 1 el mayor promedio.

Tabla 7. Parámetros de configuración del algoritmo para la representación matricial.

Parámetro	Valor
Número de iteraciones	1000000
Temperatura inicial	10000 1000000
Temperatura final	1
Tipos de enfriamiento	Lineal Hiperbólico Exponencial con $\alpha = 0.85$
Operaciones	AsignarTiposDePegadoIguales AsignarTiposDePegadoMínimos CambiarMosaicosVecinos PermutarTiposdePegadoDeMosaicos CambiarMosaicosVecinosExcluyente

Los resultados de los promedios normalizados para las 30 configuraciones se muestran en la Figura 39. En términos generales, se puede observar que las configuraciones que utilizan las operaciónes AsignarTiposDePegadoIguales y AsignarTiposDePegadoMínimos ($C_1 - C_{12}$) son las que muestran el peor desempeño para todos los casos de prueba. Sin embargo, la operación AsignarTiposDePegadoIguales tiene un buen desempeño en los patrones contador binario de 6×6, tablero de ajedrez de 6×6 y 15×15, triángulo de Sierpinski de 6×6 y 8×8. Es decir, la operación AsignarTiposDePegadoIguales obtiene mejores resultados que la operación AsignarTiposDePegadoMínimos sólo para los casos con un tamaño menor o igual a 8×8. Así, podemos decir que las configuraciones que utilizan a la operación AsignarTiposDePegadoMínimos ($C_7 - C_{12}$) son las peores para todos los casos de prueba.

La Figura 39 muestra que las configuraciones de la operación CambiarMosaicosVecinos ($C_{13} - C_{18}$), de la operación PermutarTiposdePegadoDeMosaicos ($C_{19} - C_{24}$) y de la operación CambiarMosaicosVecinosExcluyente ($C_{25} - C_{30}$) tienen un buen desempeño para todos los casos de prueba. Se observa que la operación PermutarTiposdePegadoDeMosaicos es la mejor para la mayoría de los patrones, sin embargo para el patrón

Tabla 8. Configuraciones para el algoritmo de recocido simulado.

Configuración	Operación	Perfil de enfriamiento	Temperatura inicial
C_1	AsignarTiposDePegadoIguales	Lineal	10000
C_2	AsignarTiposDePegadoIguales	Lineal	1000000
C_3	AsignarTiposDePegadoIguales	Hiperbólico	10000
C_4	AsignarTiposDePegadoIguales	Hiperbólico	1000000
C_5	AsignarTiposDePegadoIguales	Exponencial	10000
C_6	AsignarTiposDePegadoIguales	Exponencial	1000000
C_7	AsignarTiposDePegadoMínimos	Lineal	10000
C_8	AsignarTiposDePegadoMínimos	Lineal	1000000
C_9	AsignarTiposDePegadoMínimos	Hiperbólico	10000
C_{10}	AsignarTiposDePegadoMínimos	Hiperbólico	1000000
C_{11}	AsignarTiposDePegadoMínimos	Exponencial	10000
C_{12}	AsignarTiposDePegadoMínimos	Exponencial	1000000
C_{13}	CambiarMosaicosVecinos	Lineal	10000
C_{14}	CambiarMosaicosVecinos	Lineal	1000000
C_{15}	CambiarMosaicosVecinos	Hiperbólico	10000
C_{16}	CambiarMosaicosVecinos	Hiperbólico	1000000
C_{17}	CambiarMosaicosVecinos	Exponencial	10000
C_{18}	CambiarMosaicosVecinos	Exponencial	1000000
C_{19}	PermutarTiposdePegadoDeMosaicos	Lineal	10000
C_{20}	PermutarTiposdePegadoDeMosaicos	Lineal	1000000
C_{21}	PermutarTiposdePegadoDeMosaicos	Hiperbólico	10000
C_{22}	PermutarTiposdePegadoDeMosaicos	Hiperbólico	1000000
C_{23}	PermutarTiposdePegadoDeMosaicos	Exponencial	10000
C_{24}	PermutarTiposdePegadoDeMosaicos	Exponencial	1000000
C_{25}	CambiarMosaicosVecinosExcluyente	Lineal	10000
C_{26}	CambiarMosaicosVecinosExcluyente	Lineal	1000000
C_{27}	CambiarMosaicosVecinosExcluyente	Hiperbólico	10000
C_{28}	CambiarMosaicosVecinosExcluyente	Hiperbólico	1000000
C_{29}	CambiarMosaicosVecinosExcluyente	Exponencial	10000
C_{30}	CambiarMosaicosVecinosExcluyente	Exponencial	1000000

tablero de ajedrez de 15×15 los resultados obtenidos son los peores en las configuraciones que ocupan tal operación. La operación CambiarMosaicosVecinosExcluyente con las configuraciones $C_{25} - C_{28}$ obtiene el peor desempeño para tal caso; sin embargo, las configuraciones C_{29} y C_{30} obtienen las mejores soluciones en los experimentos para ese patrón. Así, con la configuración C_{30} se obtienen los mejores resultados en todos los experimentos con todos los casos de prueba. En general, se puede concluir que la operación CambiarMosaicosVecinosExcluyente obtiene los mejores resultados para todos los patrones de prueba.

Con respecto a los perfiles de enfriamiento, se puede observar lo siguiente: El perfil de enfriamiento en la operación AsignarTiposDePegadoIguales con el que se obtuvo un buen desempeño fue el exponencial (C_5 y C_6). En la operación AsignarTiposDePegadoMínimos los tres perfiles de enfriamiento tienen un desempeño similar, mientras que en las operaciones CambiarMosaicosVecinos, PermutarTiposdePegadoDeMosaicos y

CambiarMosaicosVecinosExcluyente, las configuraciones que utilizan el enfriamiento exponencial obtienen mejores resultados en comparación con el lineal e hiperbólico. Además, las configuraciones con una temperatura inicial de 1,000,000 obtienen mejores resultados a diferencia de las que utilizan una temperatura inicial de 10,000.

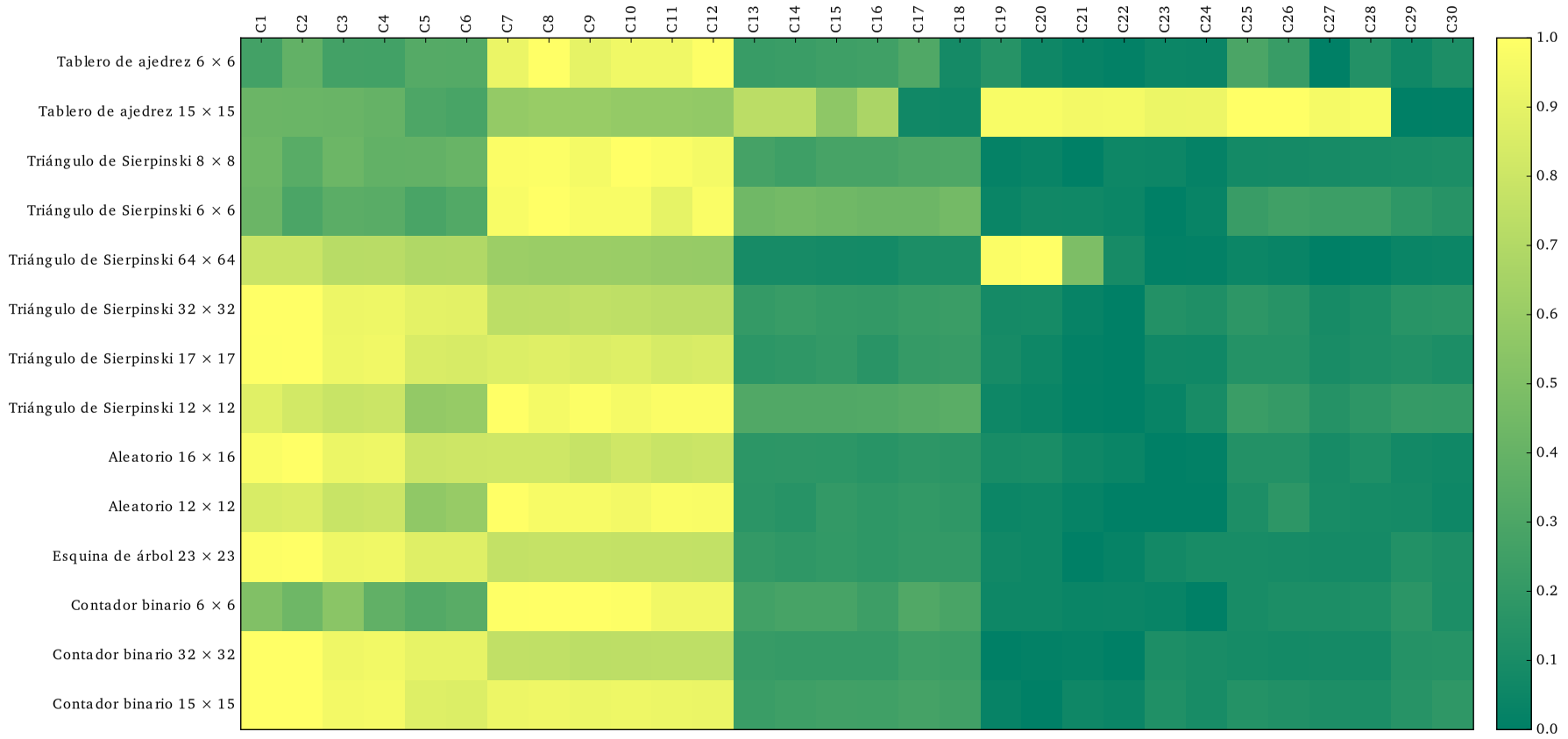


Figura 39. Mapa de calor de los resultados obtenidos con las 30 configuraciones. Un valor de 1 representa el promedio más alto que se obtuvo en los experimentos. Las configuraciones están divididas por operación de la siguiente manera: operación AsignarTiposDePegadoIguales de C_1 a C_6, operación AsignarTiposDePegadoMínimos de C_7 a C_{12}, operación CambiarMosaicosVecinos de C_{13} a C_{18}, operación PermutarTiposdePegadoDeMosaicos de C_{19} a C_{24} y operación CambiarMosaicosVecinosExcluyente de C_{25} a C_{30}.

La Tabla 9 contiene los mejores resultados de las configuraciones C_{22} y C_{30} (x), incluyendo el promedio ($\bar{x}$) y la desviación estándar (σ) de las soluciones obtenidas para cada patrón de prueba. Se puede observar que para el patrón triángulo de Sierpinski de 6×6 con la configuración C_{30}, el tamaño promedio del conjunto de mosaicos obtenido en los experimentos fue de siete y el mejor resultado fue de cuatro. Se observa además que para el patrón tablero de ajedrez de 15×15 con la misma configuración se obtiene un peor resultado comparado con el de C_{22}.

De lo anterior, se puede concluir que, los mejores resultados fueron obtenidos con la operación PermutarTiposdePegadoDeMosaicos y CambiarMosaicosVecinosExcluyente. Las mejores configuraciones en cada una de estas operaciones son la C_{22} y C_{30}, respectivamente. La configuración C_{22} obtiene las mejores soluciones para la mayoría de los casos de prueba, excepto para el patrón tablero de ajedrez de 15×15. Las mejores configuraciones son las que utilizan un perfil de enfriamiento exponencial, una temperatura inicial de 100000 y las operaciones PermutarTiposdePegadoDeMosaicos y CambiarMosaicosVecinosExcluyente. En particular, la configuración C_{22} obtuvo en promedio más soluciones mínimas para todos los patrones de prueba.

Tabla 9. Resultados con las dos mejores configuraciones (C_{22} y C_{30}). Para cada patrón de prueba se muestra el promedio ($\bar{x}$), la desviación estándar (σ) y el tamaño del conjunto de mosaicos de la mejor solución encontrada (x).

		Configuración 22		Configuración 30	
Patrón	Tamaño	$\bar{x}(\pm\sigma)$	x	$\bar{x}(\pm\sigma)$	x
Contador binario	6×6	5.80 ($\pm$ 0.98)	4	7 ($\pm$ 1.89)	4
	15×15	23.2 ($\pm$ 7.98)	12	40.70 ($\pm$ 5.48)	31
	32×32	84.70 ($\pm$ 23.66)	51	189.50 ($\pm$ 15.97)	169
Esquina de árbol	23×23	95.10 ($\pm$ 8.73)	83	122 ($\pm$ 7.06)	110
Triángulo de Sierpinski	6×6	8.80 ($\pm$ 0.75)	8	10.30 ($\pm$ 2.54)	4
	8×8	11.60 ($\pm$ 2.11)	10	13.20 ($\pm$ 2.94)	9
	12×12	29.60 ($\pm$ 4.90)	24	42 ($\pm$ 4.32)	36
	17×17	59.70 ($\pm$ 12.26)	39	74.90 ($\pm$ 4.63)	66
	32×32	183.60 ($\pm$ 25.57)	145	281.70 ($\pm$ 16.55)	259
	64×64	1322 ($\pm$ 19.59)	1291	1207.60 ($\pm$ 28.93)	1161
Tablero de ajedrez	6×6	2.80 ($\pm$ 0.98)	2	4.80 ($\pm$ 2.57)	2
	15×15	204 ($\pm$ 0.89)	203	17.80 ($\pm$ 16.02)	2
Aleatorio	12×12	43.50 ($\pm$ 2.80)	39	46.10 ($\pm$ 2.64)	43
	16×16	81.30 ($\pm$ 3.13)	76	83.30 ($\pm$ 3.37)	77

Las figuras 40, 41, 42, 43, 44 y 45 muestran el comportamiento promedio y desviación estándar de las configuraciones C_{22} y C_{30} para los diferentes patrones de prueba. Por razones de apreciación, las figuras presentan los resultados obtenidos hasta la iteración 100,000. Por ejemplo, para el patrón contador binario 32×32 resulta evidente la enorme desviación estándar de la configuración C_{22} (Figura 40E) durante las primeras 100,000 iteraciones (alrededor de $\sigma = \pm 500$ mosaicos). Después de 1,000,000 de iteraciones,

el algoritmo se estabiliza para obtener resultados de 84.7 ± 23.66 mosaicos (Tabla 9). En este sentido, es deseable que la desviación estándar sea lo más reducida posible para asegurar que los resultados obtenidos sean confiables (repetibles) convergiendo hacia el promedio.

La Figura 41 muestra que la desviación estándar es más reducida con la configuración C_{22} para el patrón esquina de árbol de 23×23. A partir de la iteración 20,000 la desviación estándar es muy grande y el número de mosaicos es mayor que con la configuración C_{30}. Para este patrón la configuración C_{30} tiene el mejor desempeño.

La Figura 42 muestra que la configuración C_{30} es la que obtiene un comportamiento más estable (menor desviación estándar) durante las iteraciones del algoritmo de recocido simulado para los casos de 12×12 y 16×16 del patrón aleatorio. A diferencia de la configuración C_{22}, en la que a partir de la iteración 60,000 comienza a disminuir el valor de la desviación estándar para ambos patrones.

En la Figura 43 se observa la desviación estándar y el promedio para los casos de 6×6, 8×8 y 12×12 del patrón triángulo de Sierpinski. Ambas configuraciones tienen un comportamiento estable para los tres tamaños del patrón. Sin embargo, para los casos de 17×17, 32×32 y 64×64, la configuración C_{22} es la menos estable como se observa en la Figura 44. Con el patrón de 32×32, la desviación estándar comienza a aumentar aproximadamente a partir de la iteración 40,000. En el caso de 64×64 se observa que en un principio, el algoritmo no mejora la solución actual hasta antes de la iteración 100,000; a diferencia de la configuración C_{30} que tiene un comportamiento estable y mejora la solución actual desde que inician las iteraciones.

Finalmente, para los casos del patrón tablero de ajedrez la desviación estándar en ambas configuraciones no es estable, como se observa en la Figura 45. Con el tamaño 15×15 la configuración 22 presenta una menor desviación estándar, sin embargo la mejor solución se obtiene con la configuración C_{30}.

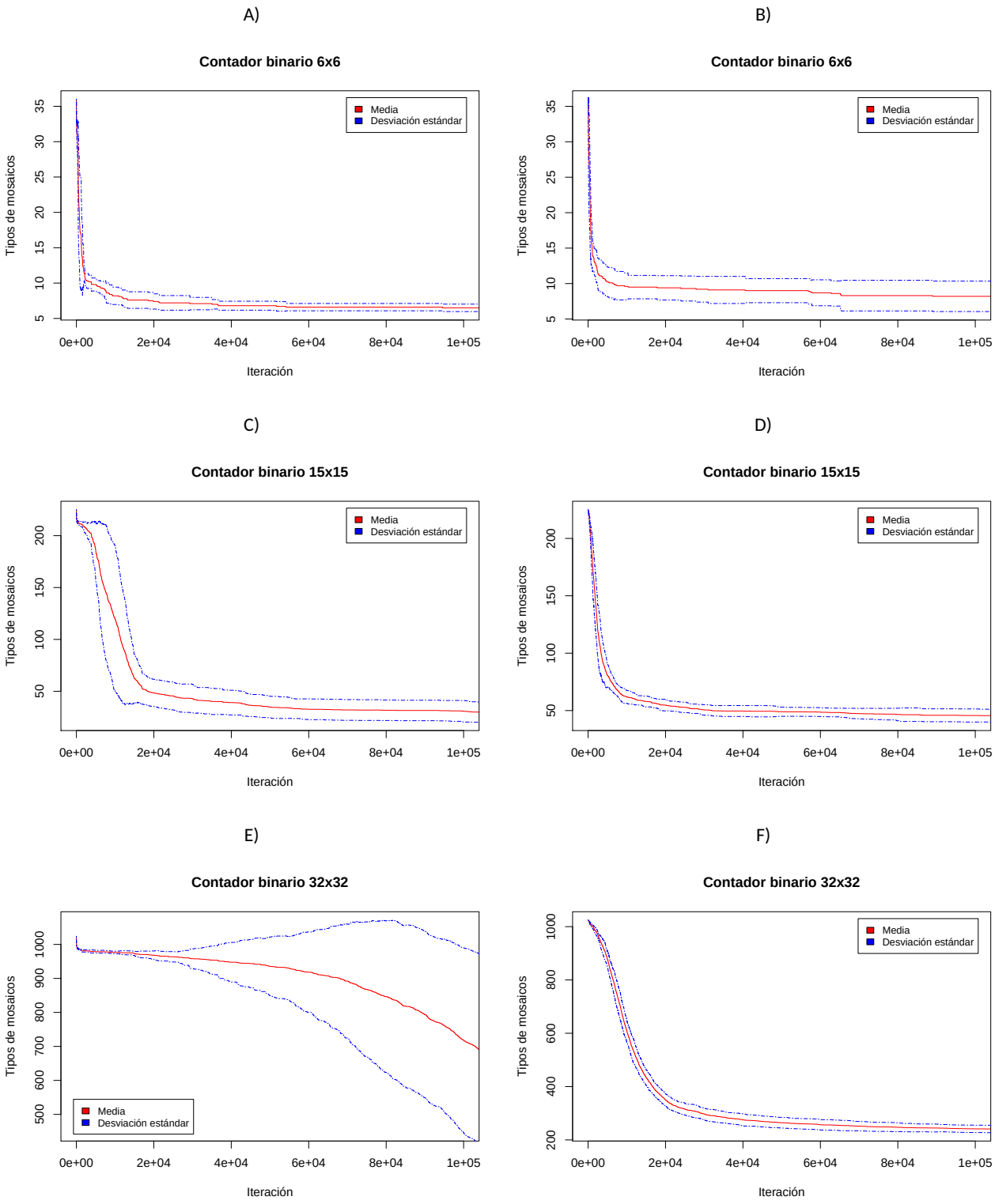


Figura 40. Media y desviación estándar de las soluciones obtenidas con las configuraciones C_{22} (paneles a la izquierda) y C_{30} (paneles a la derecha) para los casos del patrón contador binario. Las figuras presentan los resultados obtenidos hasta la iteración 100,000.

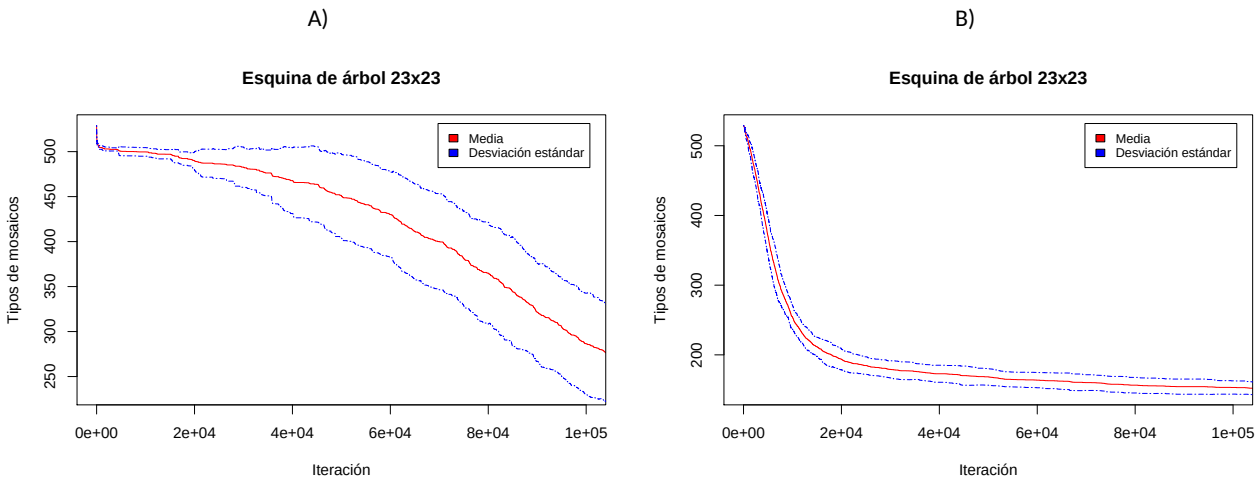


Figura 41. Media y desviación estándar de las soluciones obtenidas con las configuraciones C_{22} (paneles a la izquierda) y C_{30} (paneles a la derecha) para el patrón esquina de árbol. Las figuras presentan los resultados obtenidos hasta la iteración 100,000.

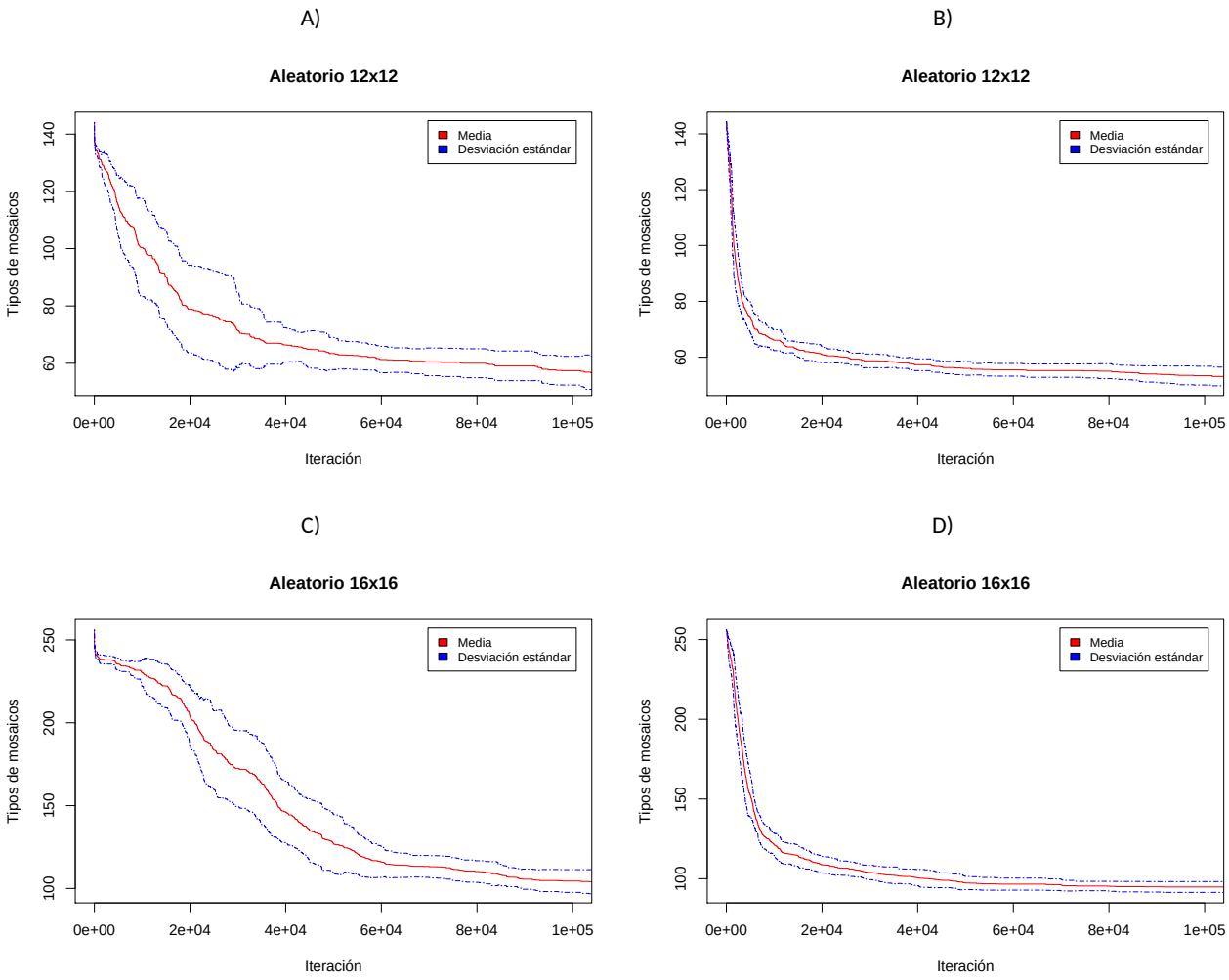


Figura 42. Media y desviación estándar de las soluciones obtenidas con las configuraciones C_{22} (paneles a la izquierda) y C_{30} (paneles a la derecha) para el patrón aleatorio. Las figuras presentan los resultados obtenidos hasta la iteración 100,000.

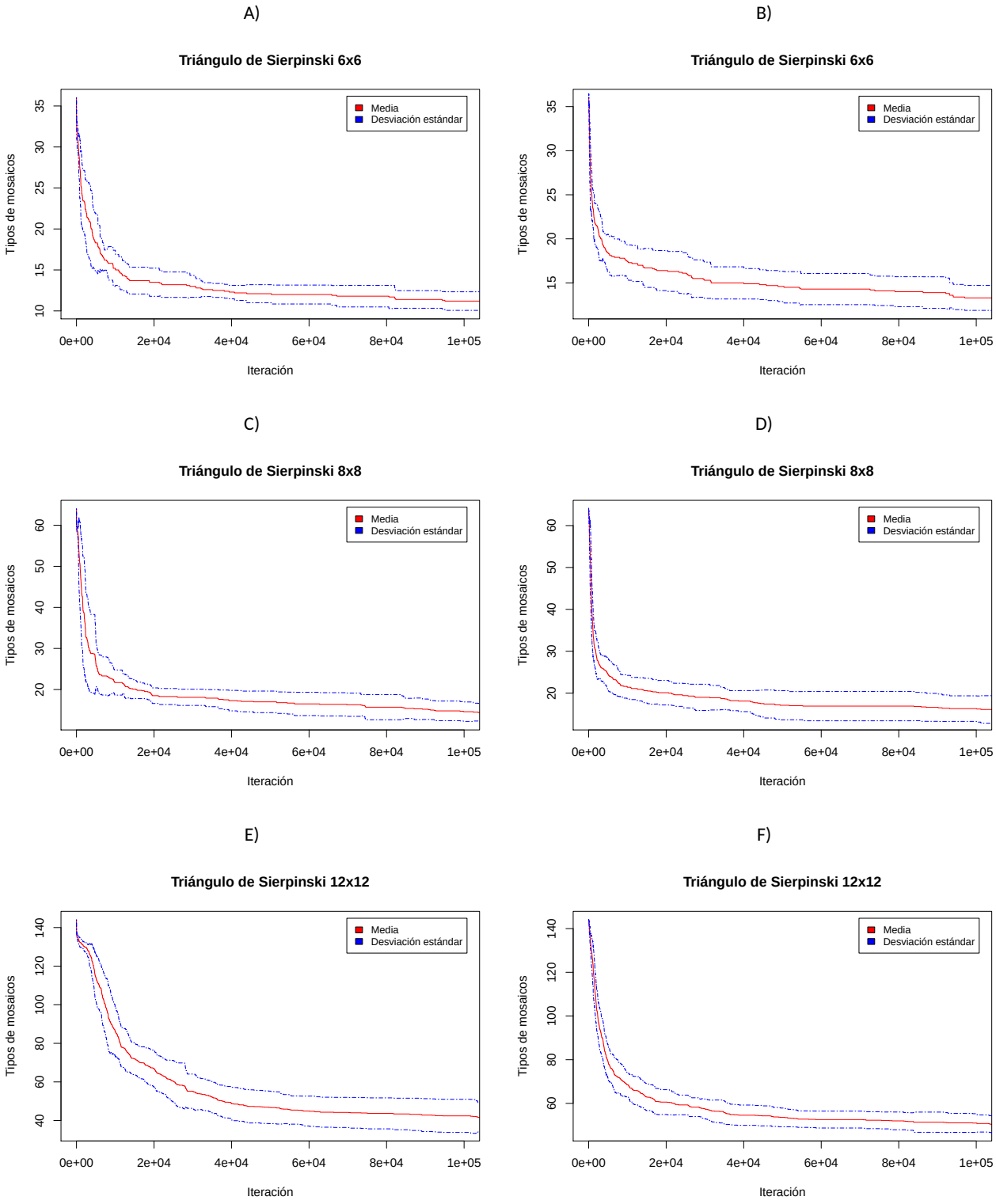


Figura 43. Media y desviación estándar de las soluciones obtenidas con las configuraciones C_{22} (paneles a la izquierda) y C_{30} (paneles a la derecha) para los casos del patrón triángulo de Sierpinski. Las figuras presentan los resultados obtenidos hasta la iteración 100,000.

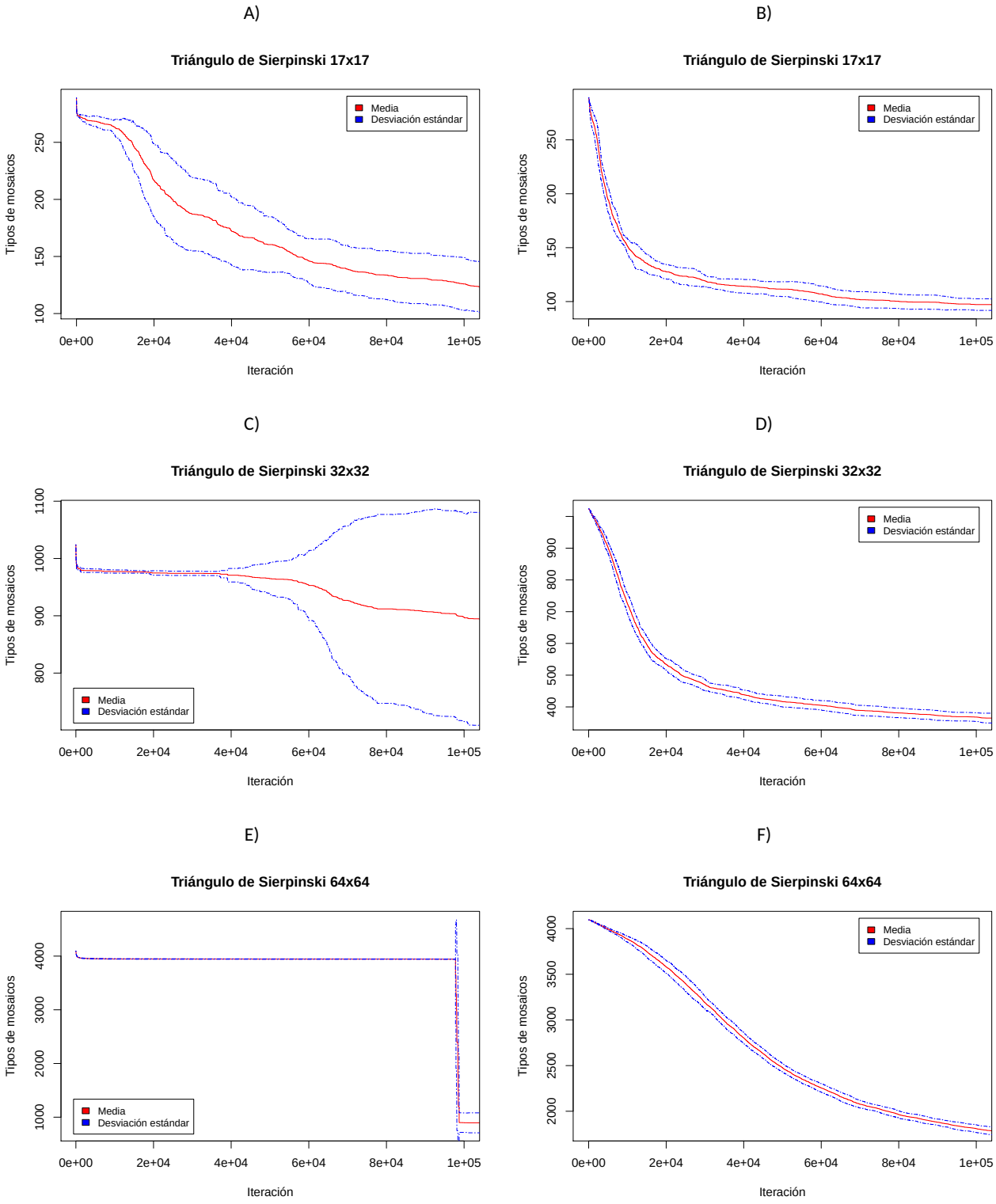


Figura 44. Media y desviación estándar de las soluciones obtenidas con las configuraciones C_{22} (paneles a la izquierda) y C_{30} (paneles a la derecha) para los casos del patrón triángulo de Sierpinski. Las figuras presentan los resultados obtenidos hasta la iteración 100,000.

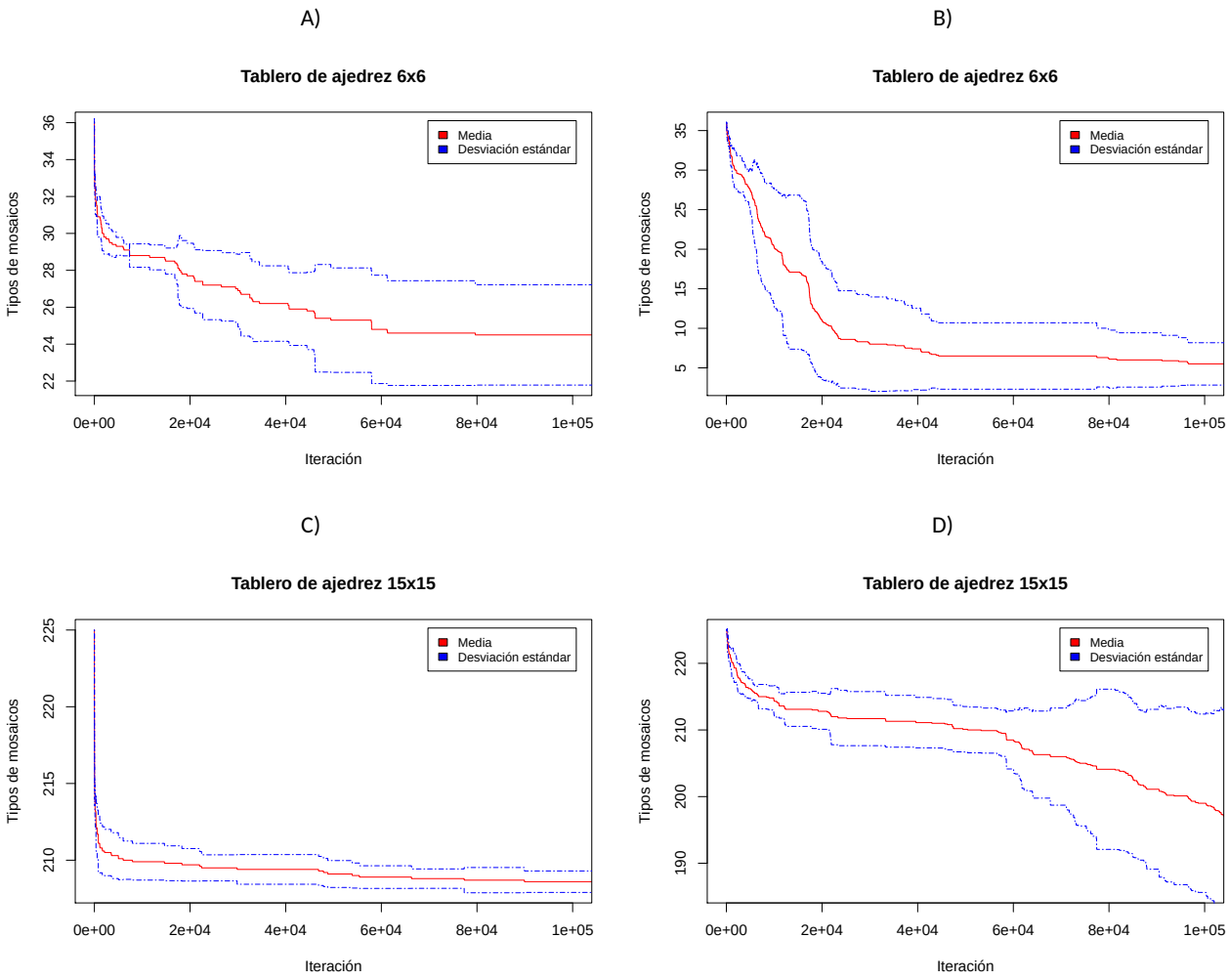


Figura 45. Media y desviación estándar de las soluciones obtenidas con las configuraciones C_{22} (paneles a la izquierda) y C_{30} (paneles a la derecha) para los casos del patrón tablero de ajedrez. Las figuras presentan los resultados obtenidos hasta la iteración 100,000.

De acuerdo al valor del promedio y la desviación estándar discutidos anteriormente, el comportamiento de las configuraciones C_{22} y C_{30} está relacionado con el tipo y tamaño del patrón; es decir, para los casos del tablero de ajedrez ambas configuraciones presentan una desviación estándar muy grande, sin embargo son las que obtienen las mejores soluciones. En relación al tamaño de los casos, se observó que para los patrones con un tamaño mayor a 17×17, la desviación estándar no era estable a lo largo de las iteraciones a diferencia de las de menor tamaño. La configuración C_{22} obtiene mejores soluciones para la mayoría de los patrones de prueba, excepto para el tablero de ajedrez de 15×15 (Tabla 10), para el que la configuración C_{30} obtiene una mejor solución. En general, la configuración C_{22} es más estable para patrones con tamaños menores a 17×17. La configuración C_{30} en promedio tiene un comportamiento estable para todos los tamaños de los casos de prueba. Los mejores resultados obtenidos con las configuraciones C_{22} y C_{30} aparecen en la Tabla 10.

Tabla 10. Resultados de las propuestas desarrolladas por métodos del estado del arte para el problema PATS. El símbolo "–" indica que el autor no incluye el patrón correspondiente en sus experimentos. [1] (Ma y Lombardi, 2008), [2] (Göös y Orponen, 2011), [3] Lempiäinen (2015). $SA_{c_{22}}$ y $SA_{c_{30}}$ corresponden a la solución obtenida por el algoritmo de recocido simulado con la configuración 22 y 30, respectivamente.

Patrón	Tamaño	Voraz [1]	PS-BB [2]	PS-H [3]	$SA_{c_{22}}$	$SA_{c_{30}}$
Contador binario	6×6	19	–	–	4	4
	15×15	121	–	–	12	31
	32×32	–	307	20	51	169
Esquina de árbol	23×23	–	192	25	83	110
Triángulo de Sierpinski	6×6	21	–	–	8	4
	8×8	37	–	–	10	9
	12×12	80	20	–	24	36
	17×17	172	23	–	39	66
	32×32	595	42	4	145	259
	64×64	–	95	4	1291	1161
Tablero de ajedrez	6×6	19	–	–	2	2
	15×15	119	–	–	203	2

4.5. Simulación en ISU TAS

Se realizaron las simulaciones de los resultados con el software ISU TAS (`http://self-assembly.net/wiki/index.php?title=ISU_TAS`) para verificar que los conjuntos de mosaicos obtenidos para cada patrón de prueba, particularmente aquellos generados con la configuración C_{30}, se ensamblan de manera correcta. El software recibe como entrada el conjunto de mosaicos para ensamblar el patrón y los mosaicos que forman la semilla σ_L. El valor de la temperatura para el auto-ensamblado es de $\tau = 2$.

El conjunto de mosaicos T que genera el algoritmo de recocido simulado tienen el siguiente formato: cada tipo de mosaico se representa con {`w` = <número entero> `s` = <número entero> `e` = <número entero> `n` = <número entero> `c` = <carácter>}, donde `w` es el valor del tipo de pegado para el lado oeste, `s` para el lado sur, `e` para el este y `n` para el norte. El color del mosaico `c` puede tomar valores N = negro o B = blanco (ver Apéndice A). Para un patrón de tamaño $m \times n$, se genera un conjunto de mosaicos σ_L con $m+n+1$ mosaicos diferentes. En todas las simulaciones realizadas, dichos mosaicos son de color rojo y son los que comienzan el auto-ensamblado a partir de un mosaico semilla (*"seed"*). Por ejemplo, considere la solución obtenida para el patrón triángulo de Sierpinski de 6×6 con la configuración C_{30}. El conjunto de mosaicos T se compone de los siguientes elementos: $t_1 = \{4, 4, 4, 4, B\}$, $t_2 = \{62, 62, 62, 4, B\}$, $t_3 = \{62, 62, 4, 62, N\}$ y $t_4 = \{4, 4, 62, 62, N\}$. El conjunto σ_L consta de 13 mosaicos, entre los que se encuentra el mosaico *seed*. La Figura 46 muestra el resultado de crear dichos conjuntos de mosaicos en ISU TAS y la ejecución de la simulación del auto-ensamblado.

Las figuras 47, 48, 49, 50 y 51 muestran el resultado de las simulaciones utilizando las soluciones encontradas

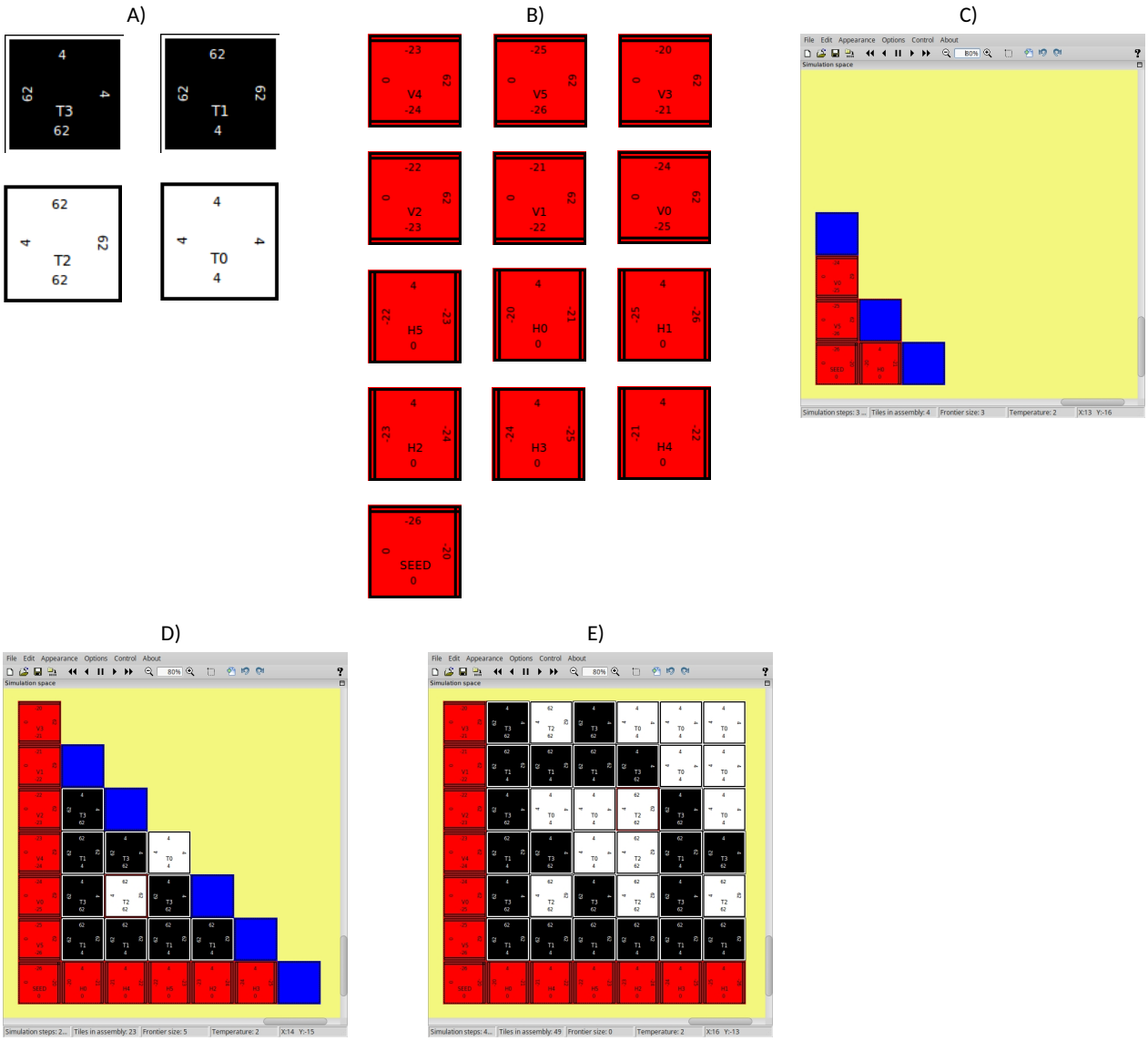
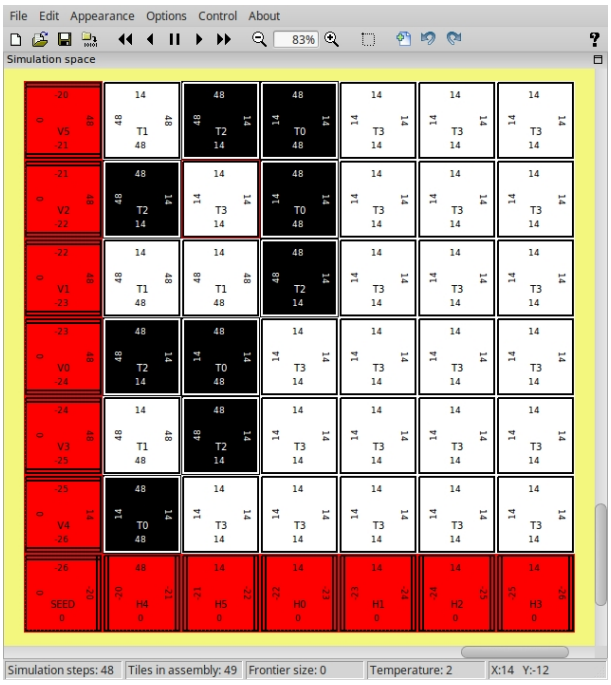


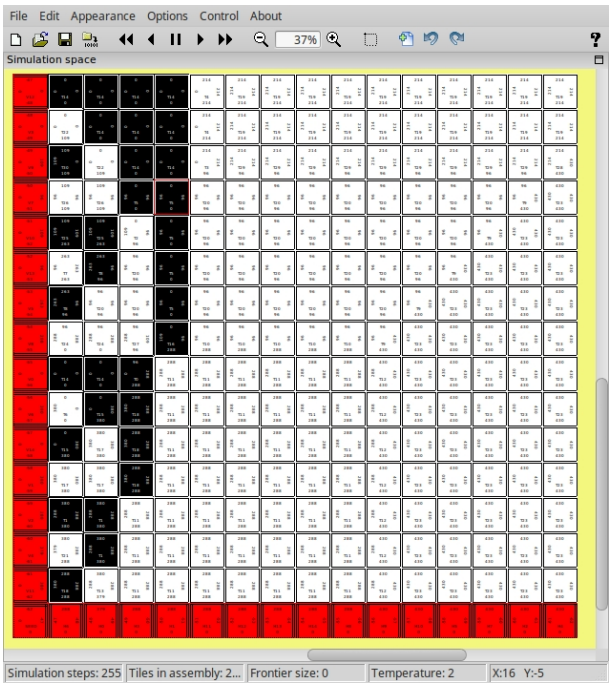
Figura 46. Simulación del patrón triángulo de Sierpinski de 6×6. A) Conjunto T. B) Conjunto σ_L. C) Se ensamblan los mosaicos de σ_L. D) Los mosaicos del conjunto T se ensamblan a partir de σ_L. E) La simulación finaliza cuando se ensamblan 6×6 mosaicos.

por el algoritmo con la configuración C_{30} (Tabla 9) para los casos del patrón contador binario, triángulo de Sierpinski, esquina de árbol, tablero de ajedrez y aleatorio, respectivamente.

A)



B)



C)

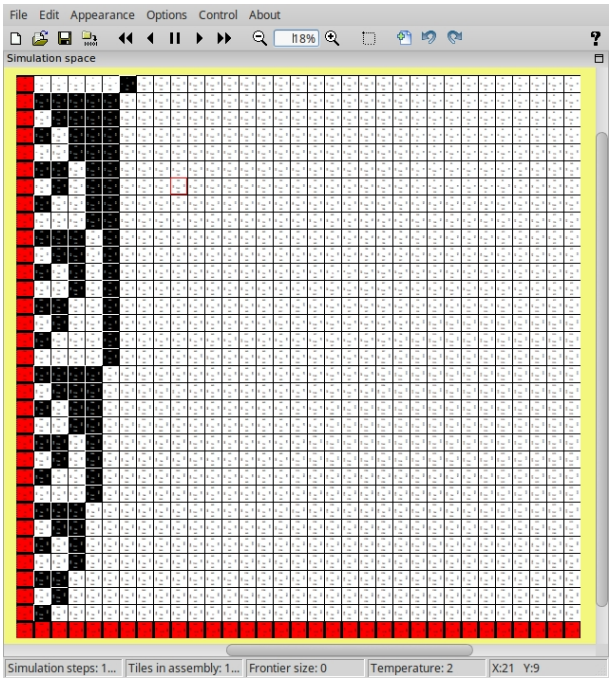


Figura 47. Simulaciones del patrón contador binario. A) 6×6. B) 15×15. C) 32×32.

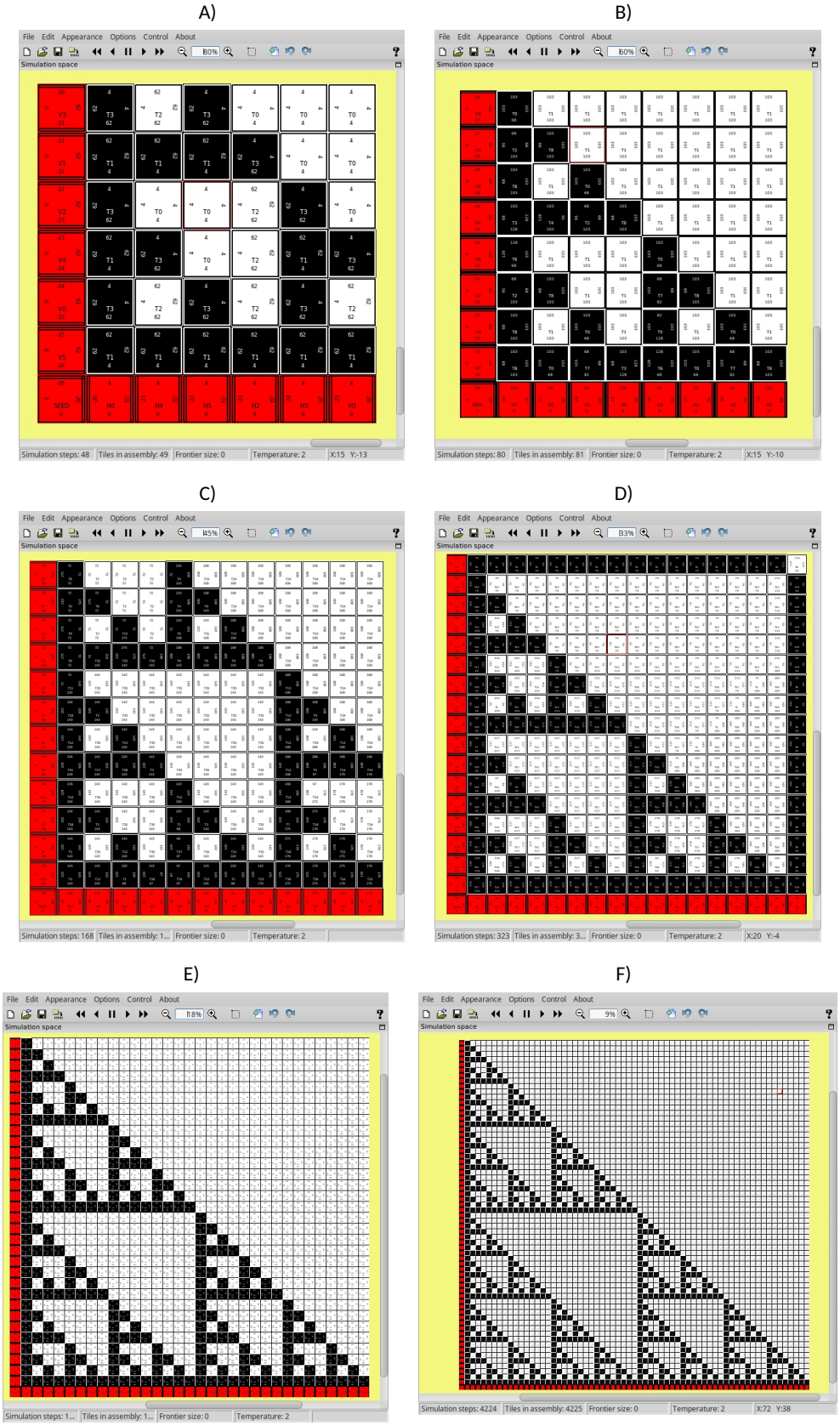


Figura 48. Simulaciones del patrón triángulo de Sierpinski. A) 6×6. B) 8×8. C) 12×12. D) 17×17. E) 32×32. F) 64×64.

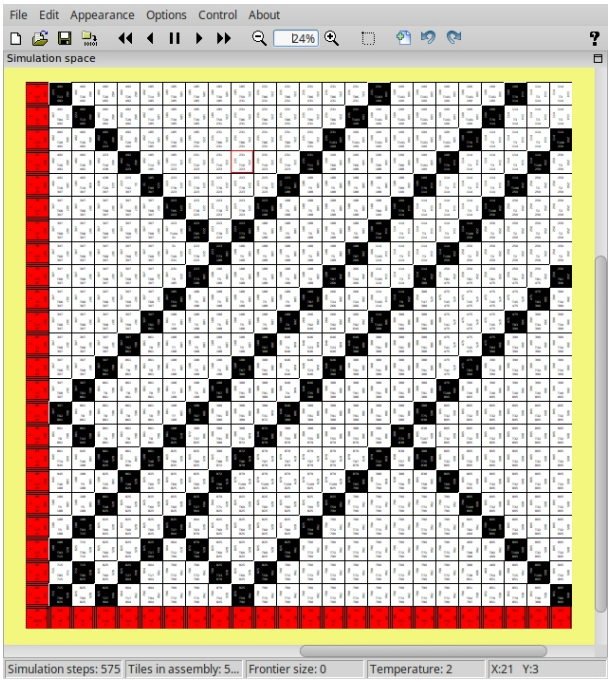
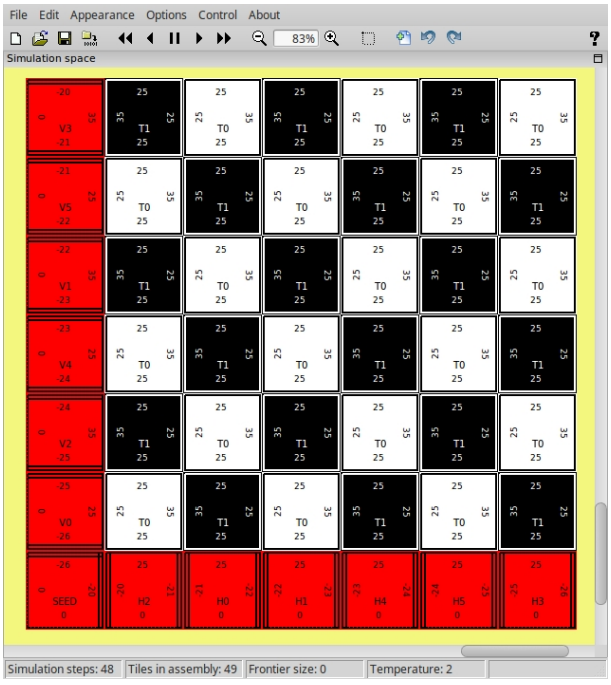


Figura 49. Simulación del patrón esquina de árbol de 23×23.

A)



B)

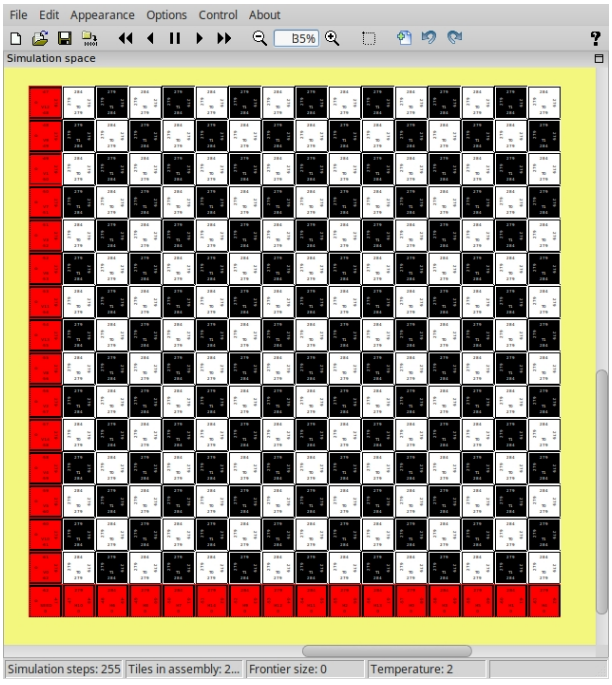
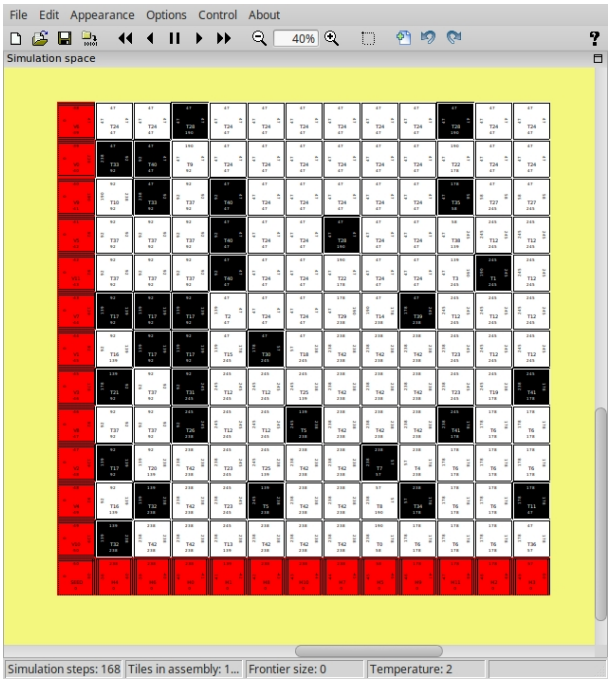


Figura 50. Simulaciones del patrón tablero de ajedrez. A) 6×6. B) 15×15.

A)



B)

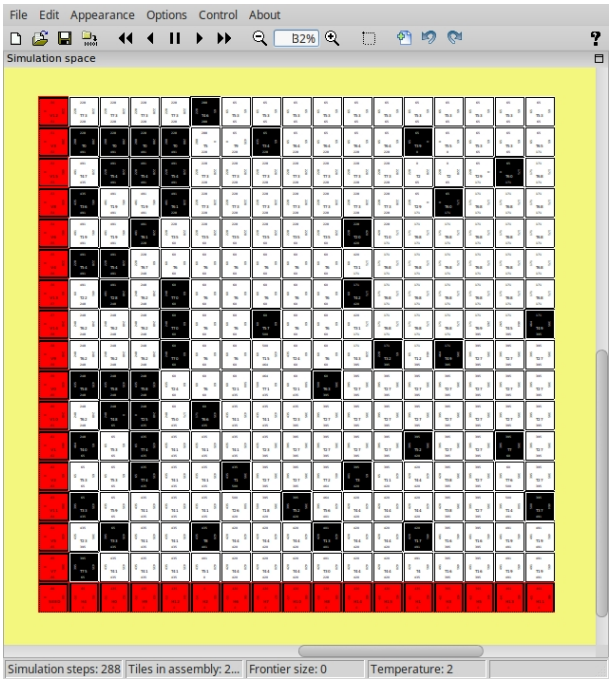


Figura 51. Simulaciones del patrón aleatorio. A) 12×12. B) 16×16.

Capítulo 5. Conclusiones

En este capítulo se presenta un sumario del trabajo de investigación realizado, así como las conclusiones a las que se llegó y algunas ideas de trabajo a futuro.

5.1. Sumario

En este trabajo se diseñó un algoritmo de recocido simulado para resolver el problema PATS. Se utilizaron dos representaciones del problema: con grafos (Ma y Lombardi, 2008) y matricial (Lozano, 2014). Se construyó una solución inicial y se diseñaron operaciones para la generación de soluciones vecinas con ambas representaciones. La implementación del algoritmo se realizó en la plataforma de desarrollo OPT4J. Para evaluar su desempeño, se ejecutaron experimentos con 14 patrones de prueba utilizados por métodos del estado del arte. Para cada uno de los patrones de entrada se construyó una solución inicial, cuyo conjunto de mosaicos fue de cardinalidad máxima. El objetivo del problema consistió en minimizar dicho conjunto.

Con respecto a la representación con grafos, se diseñó solamente una operación de vecindario. Una vez implementada en el algoritmo de recocido simulado, se realizaron una serie de experimentos con 14 casos del problema para estimar su desempeño. Con los resultados obtenidos se observó que la operación no generaba soluciones peores, es decir, no se aprovechaba el criterio probabilístico en la selección de una nueva solución, lo que impedía al algoritmo seleccionar soluciones candidatas que no precisamente mejoraban la solución actual. Debido a las desventajas de los grafos de mosaicos y de pegado para generar una solución vecina peor a la actual, se optó por implementar una representación alternativa del problema.

De esta manera, se seleccionó la representación matricial de Lozano (2014) y se diseñaron cinco operaciones para generar las soluciones vecinas. Cada operación modifica los valores de pegado de dos o más mosaicos seleccionados aleatoriamente de una matriz de dos dimensiones. Es decir, se asignan nuevos valores para los lados norte, este, sur y oeste de los mosaicos para combinar dos o más mosaicos, permitiendo disminuir o aumentar los tipos de mosaicos existentes en la matriz. El algoritmo se ejecutó con cada una de las operaciones diseñadas de acuerdo a los parámetros de la Tabla 7. En total se probaron 30 configuraciones de parámetros y cada una de ellas se ejecutó 10 veces con cada uno de los casos de los patrones de prueba, haciendo un total de 4,200 experimentos.

Posteriormente, se realizó una comparación de los resultados de todas las configuraciones y se seleccionaron las dos con el mejor desempeño en todos los casos de prueba. Enseguida, se realizó la simulación en

ISU TAS de los resultados obtenidos con la mejor configuración (C_{30}) para verificar que los conjuntos de mosaicos ensamblarán el patrón deseado. Finalmente, se realizó una comparación con los resultados existentes en la literatura para el problema PATS, considerando únicamente aquellos enfoques que construyen un RTAS.

5.2. Conclusiones

De acuerdo a los experimentos ejecutados con la representación matricial, la configuración de parámetros C_{30} es la que obtiene los mejores resultados para todos los casos de prueba. Sin embargo, para los patrones más grandes (32×32 y 64×64), esta configuración no encuentra conjuntos de mosaicos menores a los obtenidos en el estado del arte, por lo que claramente se afecta su eficiencia al aumentar el tamaño de los patrones. No obstante, para los casos de tamaño 6×6, se obtienen las soluciones óptimas.

Por otro lado, se pudo observar que la estructura (forma) de los patrones de entrada puede influir en el rendimiento de las operaciones que generan una nueva solución. En particular, el patrón triángulo de Sierpinski de 64×64 contiene regiones extensas de mosaicos que no pueden ser reducidas a mosaicos de un solo tipo al terminar de ejecutarse el algoritmo. En contraste, para los casos del patrón tablero de ajedrez (en donde no hay regiones del mismo color), los mosaicos de un mismo color en el patrón se reducen a un solo tipo.

En relación a los perfiles de enfriamiento utilizados en el algoritmo, el perfil de enfriamiento exponencial fue el que obtuvo el mejor desempeño con todas las operaciones diseñadas. El enfriamiento hiperbólico solo generó buenos resultados en combinación con la operación PermutarTiposdePegadoDeMosaicos y una temperatura inicial de 1,000,000. El enfriamiento lineal fue el de peor rendimiento con todos los patrones de prueba.

En consecuencia, el desempeño del algoritmo de recocido simulado comparado con los métodos del estado del arte, mejora al algoritmo voraz de Ma y Lombardi (2008). No obstante, para los restantes algoritmos (Tabla 10), los cuales utilizan un procesamiento en paralelo, no se mejoran los resultados para el conjunto de prueba.

5.3. Trabajo a futuro

Como trabajo a futuro, se proponen las siguientes investigaciones:

- Aplicar pruebas estadísticas a los resultados de los experimentos realizados.

- Sintonización de los parámetros del algoritmo de recocido simulado.
- Implementar otros perfiles de enfriamiento para las operaciones PermutarTiposdePegadoDeMosaicos y CambiarMosaicosVecinosExcluyente de la representación matricial.
- Combinar el algoritmo de recocido simulado con un algoritmo de búsqueda Tabú.
- Tratar de describir el comportamiento de las operaciones diseñadas en los patrones de tamaño 32×32 y 64×64 en las que se tienen regiones de mosaicos muy grandes del mismo color.
- Diseñar una representación del problema diferente a las propuestas en este trabajo de investigación.

Literatura citada

Adleman, L. M. (1994). Molecular Computation of Solutions to Combinatorial Problems. *Science*, *266*(5187), 1021-1024. Descargado de http://www.jstor.org/stable/2885489

Ansari, N. y Hou, E. (1997). *Computational Intelligence for Optimization*. Springer US.

Carbone, A. y Seeman, N. C. (2004). Molecular Tiling and DNA Self-assembly. En N. Jonoska, G. Păun y G. Rozenberg (Eds.), *Aspects of Molecular Computing: Essays Dedicated to Tom Head, on the Occasion of His 70th Birthday* (pp. 61–83). Berlin, Heidelberg: Springer Berlin Heidelberg. Descargado de http://dx.doi.org/10.1007/978-3-540-24635-0_5 doi: 10.1007/978-3-540-24635-0_5

Coello, C. A. C., Lamont, G. B. y Veldhuizen, D. A. V. (2007). Simulated Annealing. En *Evolutionary Algorithms for Solving Multi-Objective Problems* (pp. 548–557). Springer US.

Czeizler, E. y Popa, A. (2013). Synthesizing minimal tile sets for complex patterns in the framework of patterned DNA self-assembly. *Theoretical Computer Science*, *499*, 23–37. Descargado de http://dl.acm.org/citation.cfm?id=2514169.2514237http://www.sciencedirect.com/science/article/pii/S0304397513003605 doi: 10.1016/j.tcs.2013.05.009

Doty, D. (2012). Theory of Algorithmic Self-assembly. *Communications of the ACM*, *55*(12), 78–88. Descargado de http://doi.acm.org/10.1145/2380656.2380675 doi: 10.1145/2380656.2380675

Fu, T. J. y Seeman, N. C. (1993). DNA double-crossover molecules. *Biochemistry*, *32*(13), 3211–3220. Descargado de http://pubs.acs.org/doi/abs/10.1021/bi00064a003 doi: 10.1021/bi00064a003

Göös, M., Lempiäinen, T., Czeizler, E. y Orponen, P. (2013). Search Methods for Tile Sets in Patterned DNA self-assembly. *Journal of Computer and System Sciences*, *80*(1), 297–319. Descargado de http://dx.doi.org/10.1016/j.jcss.2013.08.003 doi: 10.1016/j.jcss.2013.08.003

Göös, M. y Orponen, P. (2011). Synthesizing Minimal Tile Sets for Patterned DNA Self-assembly. En Y. Sakakibara y Y. Mi (Eds.), *DNA Computing and Molecular Programming: 16th International Conference, DNA 16, Hong Kong, China, June 14-17, 2010, Revised Selected Papers* (pp. 71–82). Berlin, Heidelberg: Springer Berlin Heidelberg. Descargado de http://dx.doi.org/10.1007/978-3-642-18305-8_7 doi: 10.1007/978-3-642-18305-8_7

Haddow, P. C. y Gautam, V. K. (2013). DNA Tiling for Digital Evolvable Hardware. *Proceedings of the 2013 IEEE International Conference on Evolvable Systems, ICES 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, 104–111. doi: 10.1109/ICES.2013.6613289

Kari, L., Kopecki, S., Meunier, P.-É., Patitz, M. J. y Seki, S. (2015). Binary Pattern Tile Set Synthesis Is NP-hard. En M. M. Halldórsson, K. Iwama, N. Kobayashi y B. Speckmann (Eds.), *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings,*

Part I (pp. 1022–1034). Berlin, Heidelberg: Springer Berlin Heidelberg. Descargado de http://dx.doi.org/10.1007/978-3-662-47672-7_83 doi: 10.1007/978-3-662-47672-7_83

Kirkpatrick, S., Gelatt, C. D. y Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), pp. 671–680. Descargado de http://www.jstor.org/stable/1690046

Lempiäinen, T. (2015). *The PATS Problem: Search Methods and Reliability* (Tesis Doctoral no publicada). University of Helsinki.

Lempiäinen, T., Czeizler, E. y Orponen, P. (2011). Synthesizing Small and Reliable Tile Sets for Patterned DNA Self-assembly. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6937 LNCS*, 145–159. doi: 10.1007/978-3-642-23638-9_13

Lin, C., Liu, Y., Rinker, S. y Yan, H. (2006). DNA Tile Based Self-Assembly: Building Complex Nanoarchitectures. *ChemPhysChem*, *7*(8), 1641–1647. Descargado de http://dx.doi.org/10.1002/cphc.200600260 doi: 10.1002/cphc.200600260

Lozano, E. L. (2014). *Aplicación de algoritmos genéticos al problema PATS de auto-ensamblado de ADN* (Tesis de Maestría en Ciencias). Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California.

Lukasiewycz, M., Glaß, M., Reimann, F. y Teich, J. (2011). Opt4J - A Modular Framework for Meta-heuristic Optimization. En *Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011)* (pp. 1723–1730). Dublin, Ireland.

Ma, X. y Lombardi, F. (2008). Synthesis of Tile Sets for DNA Self-Assembly. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *27*(5), 963–967. doi: 10.1109/TCAD.2008.917973

Mao, C. (2004). The Emergence of Complexity: Lessons from DNA. *PLoS Biol*, *2*(12). Descargado de http://dx.doi.org/10.1371%2Fjournal.pbio.0020431 doi: 10.1371/journal.pbio.0020431

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. y Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. Descargado de http://dx.doi.org/10.1063/1.1699114 doi: 10.1063/1.1699114

Michalewicz, Z. y Fogel, D. B. (2004). Basic Concepts. En *How to Solve It: Modern Heuristics* (pp. 35–48). Berlin, Heidelberg: Springer Berlin Heidelberg.

Patitz, M. J. (2011). Simulation of Self-Assembly in the Abstract Tile Assembly Model with ISU TAS. *CoRR*. Descargado de http://arxiv.org/abs/1101.5151

Patitz, M. J. (2014). An Introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, *13*(2), 195–224. doi: 10.1007/s11047-013-9379-4

Pérez Jiménez, M. d. J. y Sancho Caparrini, F. (2003). *Máquinas Moleculares Basadas en ADN*. Sevilla:

Universidad de Sevilla. Secretariado de Publicaciones.

Reif, J. y Labean, T. (2012). Self-Assembled DNA Nanostructures and DNA Devices. En S. Cabrini y S. Kawata (Eds.), *Nanofabrication Handbook* (pp. 299–328). CRC Press. doi: 10.1201/b11626-17

Rothemund, P. W. K., Papadakis, N. y Winfree, E. (2004). Algorithmic Self-Assembly of DNA Sierpinski Triangles. *PLoS Biology*, *2*(12). doi: 10.1371/journal.pbio.0020424

Rothemund, P. W. K. y Winfree, E. (2000). The Program-Size Complexity of Self-Assembled Squares (Extended Abstract). En *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing* (pp. 459–468). ACM. Descargado de http://doi.acm.org/10.1145/335305.335358 doi: 10.1145/335305.335358

Rozenberg, G., Bck, T. y Kok, J. N. (2012). *Handbook of Natural Computing*. Springer Berlin Heidelberg.

Seeman, N. C. (1982). Nucleic Acid Junctions and Lattices. *Journal of Theoretical Biology*, *99*(2), 237–247. Descargado de http://www.sciencedirect.com/science/article/pii/0022519382900029 doi: http://dx.doi.org/10.1016/0022-5193(82)90002-9

Seeman, N. C. (2004). Nanotechnology and the double helix. *Scientific American*, *290*(6), 64–75.

Seki, S. (2014). Patterned Self-Assembly Tile Set Synthesis. En M.-Y. Kao (Ed.), *Encyclopedia of Algorithms* (pp. 1–4). Springer Berlin Heidelberg. Descargado de http://dx.doi.org/10.1007/978-3-642-27848-8_666-1 doi: 10.1007/978-3-642-27848-8_666-1

Serrato, A., Flores, L., Aportela, J. y Sierra, E. (2016). *PCR: Reacción en Cadena de la Polimerasa*. Descargado de http://www2.inecc.gob.mx/publicaciones/libros/710/pcr.pdf

Simon, D. (2013). Simulated Annealing. En *Evolutionary Optimization Algorithms* (pp. 227–234). John Wiley & Sons, Inc.

Sponk. (2012). *Comparación de una cadena sencilla de ARN con una doble hélice de ADN con sus correspondientes bases nitrogenadas*. Descargado de http://commons.wikimedia.org/wiki/File:Difference_DNA_RNA-EN.svg

Wang, H. (1961). Proving Theorems by Pattern Recognition II. *Bell Systems Technical Journal*, 1–42.

Winfree, E. (1998). *Algorithmic Self-Assembly of DNA* (Tesis Doctoral). California Institute of Technology.

Winfree, E., Liu, F., Wenzler, L. A. y Seeman, N. C. (1998). Design and self-assembly of two-dimensional DNA crystals. *Nature*, *394*(6693), 539–44. Descargado de http://dx.doi.org/10.1038/28998 doi: 10.1038/28998

Apéndice A. Conjuntos de mosaicos

En esta sección se presentan los conjuntos de mosaicos obtenidos por el algoritmo de recocido simulado con la configuración de parámetros C_{30} para todos los casos de prueba. Los resultados para cada patrón tienen el siguiente formato: cada tipo de mosaico se representa con {w=<número entero> s=<número entero> e=<número entero> n=<número entero> c=<carácter>}, donde w es el valor del tipo de pegado para el lado oeste, s para el lado sur, e para el este y n para el norte. El color del mosaico es c que toma valores de N=negro o B=blanco.

A.1. Contador binario 6×6

Tipos de mosaicos = 4

{w=14 s=14 e=14 n=14 c=B}

{w=14 s=48 e=14 n=48 c=N}

{w=48 s=14 e=14 n=48 c=N}

{w=48 s=48 e=48 n=14 c=B}

A.2. Contador binario 15×15

Tipos de mosaicos = 31

{w=0 s=0 e=0 n=0 c=N}

{w=0 s=96 e=214 n=214 c=B}

{w=0 s=109 e=0 n=0 c=B}

{w=0 s=214 e=214 n=214 c=B}

{w=0 s=288 e=288 n=96 c=N}

{w=0 s=380 e=380 n=0 c=N}

{w=96 s=0 e=96 n=0 c=N}

{w=96 s=96 e=96 n=96 c=B}

{w=96 s=109 e=96 n=109 c=B}

{w=96 s=263 e=263 n=263 c=B}

{w=96 s=288 e=96 n=96 c=B}

{w=96 s=430 e=430 n=96 c=B}

{w=109 s=96 e=96 n=0 c=B}

{w=109 s=109 e=0 n=109 c=N}

{w=109 s=263 e=109 n=109 c=N}

{w=109 s=288 e=96 n=0 c=N}

{w=214 s=96 e=214 n=214 c=B}

{w=214 s=214 e=214 n=214 c=B}

{w=214 s=430 e=430 n=214 c=B}

{w=263 s=96 e=96 n=263 c=N}

{w=288 s=0 e=288 n=96 c=B}

{w=288 s=96 e=109 n=96 c=B}

{w=288 s=288 e=288 n=288 c=B}

{w=288 s=379 e=288 n=380 c=B}

{w=288 s=380 e=288 n=380 c=N}

{w=288 s=430 e=430 n=430 c=B}

{w=379 s=288 e=288 n=380 c=B}

{w=380 s=0 e=0 n=0 c=B}

{w=380 s=288 e=288 n=288 c=N}

{w=380 s=380 e=380 n=380 c=B}

{w=430 s=430 e=430 n=430 c=B}

A.3. Contador binario 32×32

Tipos de mosaicos = 169

{w=3 s=252 e=252 n=3 c=N}

{w=3 s=289 e=1381 n=418 c=B}

{w=3 s=418 e=3 n=3 c=B}

{w=81 s=81 e=81 n=81 c=B}

{w=81 s=145 e=81 n=81 c=B}

{w=81 s=466 e=145 n=672 c=B}

{w=81 s=521 e=672 n=81 c=B}

{w=81 s=534 e=81 n=81 c=B}

{w=81 s=672 e=145 n=81 c=B}

{w=81 s=827 e=827 n=81 c=B}

{w=121 s=640 e=444 n=252 c=B}

{w=145 s=145 e=145 n=145 c=B}

{w=145 s=672 e=145 n=145 c=B}

{w=193 s=81 e=81 n=81 c=B}

{w=193 s=193 e=193 n=193 c=B}

{w=193 s=534 e=534 n=81 c=B}

{w=193 s=640 e=640 n=193 c=B}

{w=193 s=786 e=193 n=193 c=B}

{w=226 s=418 e=418 n=418 c=B}

{w=252 s=193 e=193 n=193 c=B}

{w=252 s=252 e=252 n=252 c=B}

{w=252 s=487 e=640 n=252 c=B}

{w=252 s=640 e=579 n=252 c=B}

{w=289 s=289 e=289 n=289 c=N}

{w=289 s=418 e=418 n=1381 c=B}

{w=289 s=548 e=418 n=289 c=B}

{w=289 s=1381 e=418 n=418 c=B}

{w=380 s=640 e=193 n=252 c=B}

{w=418 s=252 e=252 n=252 c=B}

{w=418 s=289 e=289 n=289 c=B}

{w=418 s=418 e=418 n=418 c=N}

{w=418 s=565 e=1381 n=1381 c=B}

{w=418 s=640 e=812 n=252 c=B}

{w=418 s=1003 e=1003 n=418 c=B}

{w=418 s=1381 e=418 n=418 c=B}

{w=444 s=640 e=380 n=252 c=B}

{w=487 s=640 e=640 n=252 c=B}

{w=512 s=640 e=121 n=252 c=B}

{w=521 s=672 e=145 n=521 c=B}

{w=534 s=81 e=672 n=521 c=B}

{w=534 s=466 e=521 n=81 c=B}

{w=534 s=521 e=672 n=466 c=B}

{w=534 s=534 e=534 n=534 c=B}

{w=534 s=672 e=672 n=830 c=B}

{w=534 s=827 e=827 n=534 c=B}

{w=534 s=830 e=672 n=81 c=B}

{w=534 s=1115 e=830 n=534 c=B}

{w=548 s=418 e=289 n=548 c=B}

{w=548 s=565 e=565 n=548 c=N}
{w=565 s=565 e=565 n=565 c=B}
{w=565 s=640 e=640 n=640 c=N}
{w=565 s=1003 e=565 n=1003 c=N}
{w=565 s=1515 e=640 n=640 c=B}
{w=565 s=1966 e=565 n=565 c=N}
{w=579 s=640 e=512 n=252 c=B}
{w=640 s=81 e=786 n=837 c=B}
{w=640 s=193 e=193 n=193 c=B}
{w=640 s=252 e=487 n=252 c=B}
{w=640 s=640 e=640 n=640 c=B}
{w=640 s=786 e=786 n=193 c=B}
{w=640 s=827 e=81 n=640 c=B}
{w=640 s=837 e=786 n=786 c=B}
{w=640 s=1010 e=827 n=1010 c=B}
{w=640 s=1017 e=827 n=640 c=B}
{w=640 s=1068 e=1644 n=1966 c=N}
{w=640 s=1115 e=1115 n=1909 c=B}
{w=640 s=1252 e=1644 n=1068 c=N}
{w=640 s=1265 e=1644 n=640 c=N}
{w=640 s=1351 e=1798 n=1739 c=B}
{w=640 s=1390 e=640 n=1017 c=B}
{w=640 s=1515 e=640 n=640 c=B}
{w=640 s=1525 e=1815 n=640 c=B}
{w=640 s=1642 e=1642 n=640 c=B}
{w=640 s=1644 e=1644 n=1252 c=N}
{w=640 s=1739 e=1798 n=1115 c=B}
{w=640 s=1798 e=1798 n=1525 c=B}
{w=640 s=1815 e=1351 n=640 c=B}
{w=640 s=1889 e=640 n=640 c=N}
{w=640 s=1909 e=1798 n=1010 c=B}
{w=640 s=1966 e=1068 n=1068 c=B}
{w=672 s=145 e=145 n=466 c=B}
{w=672 s=672 e=672 n=672 c=B}
{w=672 s=1115 e=672 n=672 c=B}
{w=786 s=534 e=534 n=534 c=B}
{w=786 s=786 e=786 n=786 c=B}
{w=786 s=827 e=786 n=786 c=B}
{w=786 s=1115 e=1115 n=534 c=B}
{w=812 s=640 e=886 n=252 c=B}
{w=827 s=534 e=534 n=534 c=B}
{w=827 s=640 e=1017 n=640 c=B}
{w=827 s=672 e=672 n=534 c=B}
{w=827 s=786 e=786 n=827 c=B}
{w=827 s=827 e=827 n=827 c=B}
{w=827 s=830 e=1115 n=786 c=B}
{w=827 s=1010 e=1017 n=640 c=B}
{w=827 s=1115 e=1296 n=827 c=B}
{w=827 s=1296 e=1115 n=1554 c=B}
{w=827 s=1554 e=1554 n=827 c=B}
{w=827 s=1798 e=640 n=827 c=B}
{w=827 s=1909 e=1909 n=827 c=B}
{w=830 s=1115 e=672 n=827 c=B}
{w=886 s=640 e=487 n=487 c=B}
{w=1003 s=418 e=418 n=418 c=B}
{w=1003 s=548 e=565 n=418 c=N}
{w=1003 s=565 e=565 n=1003 c=N}
{w=1003 s=1003 e=548 n=1003 c=B}
{w=1017 s=640 e=640 n=1017 c=B}
{w=1017 s=827 e=827 n=1017 c=B}
{w=1068 s=640 e=640 n=1966 c=N}
{w=1068 s=1068 e=1966 n=1966 c=N}
{w=1068 s=1966 e=1966 n=1966 c=B}
{w=1115 s=1115 e=1115 n=1115 c=B}
{w=1115 s=1372 e=1115 n=1115 c=B}
{w=1115 s=1554 e=1372 n=1115 c=B}

{w=1115 s=1798 e=1115 n=1115 c=B}
{w=1252 s=1644 e=1644 n=1515 c=B}
{w=1265 s=1265 e=1265 n=1265 c=B}
{w=1265 s=1644 e=1644 n=1265 c=N}
{w=1265 s=1889 e=1644 n=1889 c=B}
{w=1296 s=1115 e=1115 n=1296 c=B}
{w=1351 s=1798 e=1798 n=1351 c=B}
{w=1372 s=1372 e=1372 n=1372 c=B}
{w=1372 s=1554 e=1372 n=1372 c=B}
{w=1372 s=1760 e=1372 n=1372 c=B}
{w=1372 s=2045 e=2045 n=1372 c=B}
{w=1381 s=289 e=418 n=1381 c=B}
{w=1381 s=418 e=548 n=289 c=B}
{w=1381 s=640 e=640 n=418 c=N}
{w=1381 s=1003 e=1003 n=1003 c=B}
{w=1390 s=1644 e=640 n=1390 c=B}
{w=1395 s=1644 e=1252 n=1515 c=B}
{w=1515 s=1390 e=640 n=1390 c=B}
{w=1515 s=1515 e=1515 n=1515 c=B}
{w=1515 s=1644 e=1390 n=1515 c=B}
{w=1554 s=1115 e=1115 n=830 c=B}
{w=1554 s=1372 e=1372 n=1372 c=B}
{w=1554 s=1554 e=1554 n=1554 c=B}
{w=1554 s=1760 e=1372 n=1554 c=B}
{w=1554 s=1798 e=1815 n=1554 c=B}
{w=1554 s=1815 e=1760 n=1798 c=B}
{w=1642 s=1642 e=1642 n=1642 c=B}
{w=1642 s=1798 e=1798 n=1798 c=B}
{w=1644 s=640 e=640 n=640 c=B}
{w=1644 s=1265 e=1644 n=1889 c=N}
{w=1644 s=1642 e=1642 n=640 c=B}
{w=1644 s=1644 e=1644 n=1644 c=B}
{w=1644 s=1889 e=1889 n=1889 c=N}
{w=1760 s=1372 e=1372 n=1760 c=B}
{w=1760 s=1750 e=2045 n=1372 c=B}
{w=1760 s=1760 e=1760 n=1760 c=B}
{w=1798 s=1115 e=1115 n=1798 c=B}
{w=1798 s=1554 e=1554 n=1554 c=B}
{w=1798 s=1760 e=1760 n=1815 c=B}
{w=1798 s=1798 e=1798 n=1798 c=B}
{w=1798 s=1815 e=1798 n=1554 c=B}
{w=1798 s=1990 e=1760 n=1798 c=B}
{w=1815 s=1525 e=1798 n=1815 c=B}
{w=1815 s=1760 e=1760 n=1554 c=B}
{w=1815 s=1798 e=640 n=1525 c=B}
{w=1889 s=1265 e=1644 n=1644 c=N}
{w=1889 s=1644 e=1644 n=1265 c=N}
{w=1889 s=1889 e=1889 n=1889 c=B}
{w=1909 s=1115 e=1115 n=827 c=B}
{w=1966 s=640 e=1966 n=1966 c=B}
{w=1966 s=1068 e=1068 n=1966 c=N}
{w=1966 s=1515 e=1515 n=1515 c=B}
{w=1966 s=1644 e=1395 n=1515 c=B}
{w=1966 s=1966 e=1966 n=1966 c=N}
{w=2045 s=2045 e=2045 n=2045 c=B}

A.4. Tablero de ajedrez 6×6

Tipos de mosaicos = 2

{w=25 s=25 e=35 n=25 c=B}

{w=35 s=25 e=25 n=25 c=N}

A.5. Tablero de ajedrez 15×15

Tipos de mosaicos = 2

{w=279 s=279 e=279 n=284 c=B}

{w=279 s=284 e=279 n=279 c=N}

A.6. Esquina de árbol 23×23

Tipos de mosaicos = 110

{w=31 s=307 e=223 n=185 c=N}

{w=100 s=100 e=100 n=100 c=B}

{w=100 s=114 e=114 n=100 c=N}

{w=100 s=188 e=100 n=100 c=B}

{w=114 s=114 e=114 n=114 c=B}

{w=114 s=250 e=250 n=114 c=N}

{w=114 s=388 e=250 n=114 c=N}

{w=185 s=185 e=185 n=185 c=B}

{w=185 s=223 e=231 n=185 c=B}

{w=185 s=231 e=231 n=185 c=B}

{w=185 s=307 e=438 n=492 c=B}

{w=188 s=100 e=100 n=188 c=B}

{w=188 s=114 e=114 n=100 c=N}

{w=188 s=188 e=188 n=188 c=B}

{w=188 s=266 e=266 n=188 c=N}

{w=188 s=388 e=388 n=188 c=N}

{w=188 s=646 e=266 n=188 c=N}

{w=188 s=715 e=825 n=188 c=N}

{w=188 s=825 e=825 n=861 c=N}

{w=188 s=861 e=188 n=825 c=B}

{w=223 s=188 e=188 n=223 c=N}

{w=223 s=223 e=223 n=223 c=B}

{w=223 s=307 e=31 n=223 c=B}

{w=223 s=492 e=492 n=492 c=N}

{w=231 s=100 e=100 n=231 c=N}

{w=231 s=188 e=100 n=231 c=N}

{w=231 s=223 e=231 n=231 c=B}

{w=231 s=231 e=231 n=231 c=B}

{w=250 s=250 e=250 n=250 c=B}

{w=250 s=388 e=475 n=388 c=B}

{w=250 s=390 e=250 n=250 c=N}

{w=250 s=475 e=250 n=250 c=B}

{w=266 s=114 e=114 n=114 c=B}

{w=266 s=266 e=266 n=266 c=B}

{w=266 s=388 e=388 n=114 c=N}

{w=266 s=646 e=266 n=266 c=B}

{w=307 s=31 e=307 n=223 c=B}

{w=307 s=185 e=223 n=231 c=B}

{w=307 s=188 e=188 n=185 c=N}

{w=307 s=223 e=223 n=223 c=N}

{w=307 s=231 e=223 n=31 c=B}

{w=307 s=307 e=307 n=307 c=B}

{w=307 s=492 e=223 n=492 c=B}

{w=307 s=861 e=861 n=307 c=N}

{w=388 s=388 e=388 n=388 c=B}

{w=388 s=475 e=475 n=475 c=B}

{w=388 s=646 e=646 n=646 c=B}

{w=388 s=790 e=838 n=388 c=B}

{w=388 s=805 e=390 n=475 c=N}

{w=388 s=838 e=805 n=388 c=N}

{w=388 s=851 e=790 n=805 c=B}

{w=388 s=872 e=646 n=388 c=B}

{w=388 s=879 e=388 n=388 c=B}

{w=390 s=390 e=390 n=390 c=B}

{w=390 s=790 e=805 n=805 c=N}

{w=390 s=805 e=390 n=390 c=B}

{w=438 s=307 e=223 n=438 c=B}

{w=469 s=851 e=388 n=469 c=B}

{w=475 s=390 e=390 n=475 c=N}

{w=475 s=475 e=475 n=475 c=B}

{w=492 s=185 e=185 n=185 c=B}

{w=492 s=223 e=492 n=492 c=N}

{w=492 s=307 e=185 n=492 c=B}

{w=492 s=438 e=492 n=223 c=B}

{w=492 s=492 e=492 n=492 c=B}

{w=646 s=388 e=388 n=646 c=N}

{w=646 s=646 e=646 n=646 c=B}

{w=646 s=879 e=388 n=872 c=N}

{w=715 s=715 e=715 n=715 c=B}

{w=715 s=825 e=825 n=715 c=N}

{w=790 s=388 e=790 n=469 c=B}

{w=790 s=469 e=790 n=805 c=B}

{w=790 s=790 e=790 n=790 c=B}

{w=790 s=797 e=851 n=851 c=B}

{w=790 s=805 e=805 n=805 c=N}

{w=790 s=825 e=825 n=879 c=B}

{w=790 s=851 e=469 n=388 c=B}

{w=790 s=879 e=825 n=825 c=N}

{w=790 s=904 e=790 n=904 c=B}

{w=805 s=388 e=805 n=838 c=B}

{w=805 s=790 e=390 n=838 c=B}

{w=805 s=805 e=805 n=805 c=B}

{w=825 s=715 e=825 n=715 c=B}

{w=825 s=790 e=790 n=825 c=N}

{w=825 s=825 e=825 n=825 c=B}

{w=825 s=861 e=188 n=861 c=B}

{w=825 s=872 e=825 n=388 c=B}

{w=825 s=879 e=879 n=872 c=N}

{w=825 s=904 e=904 n=825 c=N}

{w=838 s=388 e=388 n=838 c=B}

{w=838 s=790 e=805 n=388 c=B}

{w=851 s=388 e=851 n=805 c=B}

{w=851 s=805 e=805 n=805 c=N}

{w=851 s=851 e=851 n=851 c=B}

{w=861 s=188 e=188 n=188 c=B}

{w=861 s=825 e=872 n=188 c=N}

{w=861 s=861 e=861 n=861 c=B}

{w=861 s=945 e=861 n=861 c=N}

{w=872 s=790 e=825 n=879 c=N}

{w=872 s=825 e=388 n=388 c=B}

{w=879 s=388 e=838 n=388 c=N}

{w=879 s=790 e=388 n=879 c=N}

{w=879 s=825 e=879 n=879 c=B}

{w=879 s=879 e=879 n=879 c=B}

{w=904 s=790 e=872 n=904 c=B}

{w=904 s=825 e=825 n=879 c=B}

{w=904 s=879 e=825 n=904 c=B}

{w=904 s=904 e=790 n=904 c=B}

{w=945 s=188 e=188 n=945 c=B}

{w=945 s=945 e=945 n=861 c=B}

A.7. Aleatorio 12×12

Tipos de mosaicos = 43

{w=47 s=47 e=47 n=47 c=B}

{w=47 s=58 e=58 n=178 c=N}

{w=47 s=92 e=47 n=190 c=B}

{w=47 s=139 e=245 n=58 c=B}

{w=47 s=178 e=47 n=190 c=B}

{w=47 s=190 e=47 n=47 c=N}

{w=47 s=238 e=190 n=178 c=B}

{w=47 s=245 e=190 n=139 c=B}

{w=57 s=178 e=178 n=238 c=N}

{w=57 s=238 e=178 n=238 c=B}

{w=57 s=245 e=238 n=47 c=B}

{w=58 s=245 e=58 n=47 c=B}

{w=92 s=47 e=47 n=47 c=N}

{w=92 s=92 e=92 n=92 c=B}

{w=92 s=139 e=139 n=92 c=B}

{w=92 s=238 e=245 n=245 c=N}

{w=92 s=245 e=245 n=92 c=N}

{w=139 s=47 e=47 n=47 c=B}

{w=139 s=92 e=139 n=92 c=N}

{w=139 s=139 e=238 n=92 c=B}

{w=139 s=238 e=238 n=139 c=N}

{w=139 s=245 e=178 n=47 c=B}

{w=178 s=47 e=178 n=178 c=N}

{w=178 s=57 e=178 n=47 c=B}

{w=178 s=92 e=92 n=139 c=N}

{w=178 s=178 e=178 n=178 c=B}

{w=178 s=238 e=245 n=47 c=N}

{w=178 s=245 e=57 n=47 c=N}

{w=190 s=92 e=238 n=92 c=B}

{w=190 s=238 e=178 n=47 c=B}

{w=190 s=245 e=245 n=245 c=N}

{w=238 s=57 e=57 n=238 c=N}

{w=238 s=58 e=178 n=190 c=B}

{w=238 s=92 e=92 n=47 c=N}

{w=238 s=139 e=238 n=245 c=B}

{w=238 s=178 e=178 n=245 c=N}

{w=238 s=190 e=57 n=57 c=B}

{w=238 s=238 e=238 n=238 c=B}

{w=238 s=245 e=245 n=245 c=B}

{w=245 s=139 e=238 n=245 c=B}

{w=245 s=178 e=238 n=245 c=B}

{w=245 s=238 e=238 n=139 c=N}

{w=245 s=245 e=245 n=245 c=B}

A.8. Aleatorio 16×16

Tipos de mosaicos = 77

{w=0 s=0 e=65 n=65 c=B}

{w=0 s=171 e=171 n=65 c=N}

{w=0 s=228 e=435 n=65 c=B}

{w=0 s=435 e=248 n=248 c=N}

{w=60 s=60 e=60 n=60 c=B}

{w=60 s=171 e=171 n=420 c=B}

{w=60 s=248 e=491 n=491 c=B}

{w=60 s=395 e=228 n=171 c=B}

{w=60 s=420 e=171 n=171 c=N}

{w=60 s=435 e=500 n=60 c=B}

{w=60 s=464 e=435 n=500 c=B}

{w=60 s=500 e=60 n=60 c=N}

{w=65 s=0 e=0 n=65 c=N}

{w=65 s=65 e=65 n=65 c=B}

{w=65 s=171 e=65 n=65 c=B}

{w=65 s=228 e=65 n=65 c=B}

{w=65 s=395 e=464 n=171 c=B}

{w=65 s=420 e=420 n=395 c=B}

{w=65 s=435 e=435 n=435 c=N}

{w=171 s=171 e=171 n=171 c=B}

{w=171 s=395 e=265 n=171 c=B}

{w=171 s=435 e=435 n=60 c=N}

{w=228 s=60 e=228 n=228 c=B}

{w=228 s=65 e=228 n=0 c=B}

{w=228 s=171 e=0 n=65 c=B}

{w=228 s=228 e=228 n=228 c=B}

{w=228 s=248 e=60 n=228 c=B}

{w=228 s=288 e=65 n=288 c=N}

{w=228 s=395 e=65 n=171 c=N}

{w=228 s=420 e=420 n=228 c=N}

{w=228 s=491 e=228 n=491 c=N}

{w=248 s=60 e=60 n=60 c=N}

{w=248 s=65 e=0 n=248 c=N}

{w=248 s=248 e=248 n=248 c=B}

{w=248 s=435 e=171 n=60 c=B}

{w=265 s=395 e=464 n=171 c=B}

{w=288 s=228 e=0 n=288 c=B}

{w=288 s=491 e=288 n=228 c=N}

{w=395 s=60 e=395 n=395 c=N}

{w=395 s=65 e=435 n=395 c=N}

{w=395 s=395 e=395 n=395 c=B}

{w=395 s=401 e=420 n=464 c=B}

{w=395 s=420 e=395 n=395 c=N}

{w=395 s=435 e=435 n=65 c=N}

{w=395 s=464 e=401 n=395 c=B}

{w=395 s=491 e=500 n=500 c=B}

{w=395 s=500 e=395 n=60 c=B}

{w=401 s=420 e=65 n=395 c=N}

{w=420 s=171 e=171 n=228 c=B}

{w=420 s=228 e=420 n=401 c=B}

{w=420 s=395 e=395 n=395 c=B}

{w=420 s=401 e=420 n=401 c=N}

{w=420 s=420 e=420 n=420 c=B}

{w=420 s=435 e=491 n=491 c=B}

{w=420 s=491 e=491 n=420 c=N}

{w=435 s=0 e=420 n=401 c=B}

{w=435 s=60 e=60 n=60 c=B}

{w=435 s=65 e=435 n=65 c=B}

{w=435 s=228 e=65 n=65 c=N}

{w=435 s=248 e=435 n=248 c=N}

{w=435 s=395 e=395 n=435 c=B}

{w=435 s=401 e=420 n=435 c=N}

{w=435 s=420 e=500 n=500 c=B}

{w=435 s=435 e=435 n=435 c=B}

{w=435 s=491 e=491 n=435 c=N}

{w=435 s=500 e=395 n=435 c=N}

{w=464 s=65 e=65 n=248 c=N}

{w=464 s=395 e=395 n=171 c=N}

{w=491 s=228 e=228 n=491 c=N}

{w=491 s=248 e=248 n=491 c=N}

{w=491 s=395 e=491 n=395 c=B}

{w=491 s=435 e=228 n=491 c=B}

{w=491 s=491 e=491 n=491 c=B}

{w=500 s=395 e=395 n=60 c=N}

{w=500 s=420 e=395 n=395 c=B}

{w=500 s=435 e=60 n=464 c=B}

{w=500 s=491 e=464 n=395 c=N}

A.9. Triángulo de Sierpinski 6×6

Tipos de mosaicos = 4

{w=4 s=4 e=4 n=4 c=B}

{w=4 s=62 e=62 n=62 c=B}

{w=62 s=4 e=62 n=62 c=N}

{w=62 s=62 e=4 n=4 c=N}

A.10. Triángulo de Sierpinski 8×8

Tipos de mosaicos = 9

{w=68 s=103 e=103 n=103 c=N}

{w=68 s=128 e=128 n=103 c=N}

{w=95 s=103 e=68 n=68 c=N}

{w=103 s=68 e=103 n=103 c=N}

{w=103 s=82 e=68 n=68 c=N}

{w=103 s=103 e=103 n=103 c=B}

{w=103 s=128 e=103 n=82 c=N}

{w=128 s=68 e=103 n=128 c=N}

{w=128 s=103 e=95 n=103 c=N}

A.11. Triángulo de Sierpinski 12×12

Tipos de mosaicos = 36

{w=67 s=67 e=143 n=67 c=N}

{w=72 s=72 e=72 n=72 c=B}

{w=72 s=108 e=210 n=210 c=N}

{w=72 s=143 e=210 n=72 c=N}

{w=72 s=210 e=108 n=210 c=N}

{w=72 s=220 e=143 n=72 c=N}

{w=72 s=275 e=72 n=72 c=N}

m{w=108 s=67 e=108 n=286 c=N}

{w=108 s=108 e=108 n=108 c=B}

{w=108 s=143 e=108 n=108 c=N}

{w=108 s=179 e=108 n=108 c=N}

{w=108 s=275 e=179 n=67 c=B}

{w=108 s=286 e=143 n=143 c=B}

{w=143 s=67 e=143 n=68 c=N}

{w=143 s=68 e=220 n=220 c=N}

{w=143 s=108 e=108 n=108 c=N}

{w=143 s=143 e=143 n=143 c=B}

{w=143 s=179 e=179 n=275 c=N}

{w=143 s=220 e=143 n=143 c=N}

{w=143 s=275 e=143 n=108 c=N}

{w=179 s=179 e=179 n=179 c=B}

{w=179 s=275 e=179 n=179 c=N}

{w=203 s=143 e=220 n=220 c=N}

{w=210 s=72 e=275 n=72 c=N}

{w=210 s=108 e=143 n=143 c=B}

{w=210 s=143 e=275 n=275 c=N}

{w=220 s=67 e=67 n=143 c=N}

{w=220 s=143 e=143 n=143 c=N}

{w=220 s=179 e=275 n=143 c=N}

{w=229 s=143 e=203 n=143 c=N}

{w=229 s=220 e=143 n=229 c=N}

{w=229 s=229 e=229 n=220 c=N}

{w=275 s=72 e=72 n=72 c=N}

{w=275 s=143 e=108 n=72 c=N}

{w=275 s=179 e=179 n=275 c=N}

{w=275 s=275 e=275 n=179 c=N}

A.12. Triángulo de Sierpinski 17×17

Tipos de mosaicos = 66

{w=50 s=50 e=50 n=50 c=N}

{w=50 s=55 e=517 n=517 c=N}

{w=50 s=79 e=50 n=79 c=N}

{w=50 s=115 e=115 n=50 c=N}

{w=50 s=339 e=50 n=55 c=B}

{w=50 s=359 e=50 n=452 c=B}

{w=50 s=378 e=522 n=378 c=N}

{w=50 s=452 e=452 n=452 c=N}

{w=50 s=517 e=359 n=359 c=N}

{w=50 s=566 e=566 n=339 c=N}

{w=55 s=50 e=50 n=517 c=N}

{w=55 s=55 e=55 n=55 c=N}

{w=55 s=339 e=517 n=566 c=B}

{w=55 s=517 e=517 n=517 c=N}

{w=79 s=50 e=79 n=359 c=N}

{w=79 s=79 e=79 n=79 c=B}

{w=79 s=115 e=50 n=115 c=B}

{w=79 s=213 e=79 n=79 c=B}

{w=79 s=359 e=359 n=79 c=N}

{w=79 s=452 e=79 n=79 c=N}

{w=79 s=485 e=50 n=115 c=B}

{w=79 s=517 e=213 n=79 c=N}

{w=115 s=50 e=115 n=115 c=B}

{w=115 s=485 e=485 n=79 c=B}

{w=213 s=79 e=115 n=213 c=B}

{w=213 s=213 e=213 n=213 c=B}

{w=213 s=359 e=378 n=517 c=B}

{w=213 s=378 e=485 n=359 c=N}

{w=213 s=485 e=485 n=79 c=B}

{w=213 s=517 e=213 n=213 c=N}

{w=339 s=55 e=55 n=517 c=N}

{w=339 s=339 e=339 n=339 c=N}

{w=339 s=452 e=339 n=566 c=N}

{w=339 s=517 e=517 n=517 c=B}

{w=339 s=566 e=517 n=339 c=B}

{w=359 s=79 e=452 n=452 c=B}

{w=359 s=213 e=517 n=517 c=B}

{w=359 s=517 e=452 n=452 c=N}

{w=378 s=50 e=517 n=50 c=B}

{w=378 s=378 e=378 n=378 c=B}

{w=378 s=485 e=485 n=485 c=N}

{w=378 s=522 e=522 n=485 c=N}

{w=452 s=50 e=452 n=50 c=N}

{w=452 s=79 e=79 n=79 c=N}

{w=452 s=213 e=359 n=79 c=N}

{w=452 s=339 e=339 n=517 c=B}

{w=452 s=452 e=452 n=452 c=B}

{w=452 s=517 e=213 n=452 c=N}

{w=485 s=50 e=50 n=50 c=N}

{w=485 s=378 e=485 n=485 c=N}

{w=485 s=485 e=485 n=485 c=B}

{w=485 s=522 e=485 n=485 c=N}

{w=517 s=50 e=378 n=378 c=N}

{w=517 s=55 e=517 n=517 c=N}

{w=517 s=213 e=213 n=213 c=N}

{w=517 s=339 e=55 n=55 c=N}

{w=517 s=378 e=517 n=485 c=N}

{w=517 s=485 e=378 n=213 c=N}

{w=517 s=517 e=517 n=517 c=B}

{w=517 s=566 e=452 n=517 c=N}

{w=522 s=50 e=522 n=522 c=N}

{w=522 s=485 e=50 n=485 c=N}

{w=522 s=522 e=485 n=378 c=B}

{w=566 s=50 e=517 n=50 c=N}

{w=566 s=339 e=55 n=50 c=N}

{w=566 s=517 e=55 n=55 c=N}

A.13. Triángulo de Sierpinski 32×32

Tipos de mosaicos = 259

{w=7 s=692 e=692 n=169 c=B}

{w=51 s=215 e=51 n=215 c=B}

{w=51 s=343 e=343 n=215 c=B}

{w=51 s=516 e=51 n=51 c=B}

{w=80 s=215 e=979 n=215 c=B}

{w=80 s=343 e=80 n=215 c=B}

{w=169 s=7 e=7 n=169 c=B}

{w=169 s=169 e=169 n=169 c=B}

{w=169 s=215 e=215 n=169 c=B}

{w=169 s=239 e=692 n=944 c=N}

{w=169 s=692 e=692 n=215 c=B}

{w=169 s=748 e=748 n=994 c=N}

{w=169 s=867 e=944 n=1396 c=N}

{w=169 s=944 e=1396 n=169 c=N}

{w=169 s=994 e=169 n=169 c=N}

{w=215 s=51 e=51 n=215 c=B}

{w=215 s=80 e=80 n=215 c=B}

{w=215 s=215 e=215 n=215 c=B}

{w=215 s=262 e=215 n=215 c=B}

{w=215 s=343 e=343 n=80 c=B}

{w=215 s=516 e=516 n=51 c=B}

{w=215 s=692 e=215 n=215 c=B}

{w=215 s=979 e=979 n=979 c=B}

{w=215 s=1054 e=1054 n=774 c=B}

{w=239 s=239 e=239 n=239 c=B}

{w=239 s=262 e=262 n=996 c=B}

{w=239 s=692 e=994 n=748 c=N}

{w=239 s=748 e=239 n=239 c=B}

{w=239 s=867 e=239 n=867 c=N}

{w=239 s=944 e=867 n=867 c=N}

{w=239 s=996 e=996 n=692 c=N}

{w=239 s=1254 e=1254 n=239 c=N}

{w=239 s=1847 e=996 n=996 c=N}

{w=262 s=215 e=262 n=262 c=B}

{w=262 s=262 e=262 n=262 c=B}

{w=262 s=365 e=365 n=262 c=N}

{w=262 s=516 e=516 n=215 c=B}

{w=262 s=565 e=516 n=1253 c=B}

{w=262 s=890 e=902 n=565 c=B}

{w=262 s=1253 e=516 n=516 c=B}

{w=262 s=1391 e=262 n=262 c=N}

{w=343 s=215 e=979 n=215 c=B}

{w=343 s=343 e=343 n=343 c=B}

{w=343 s=774 e=215 n=215 c=B}

{w=343 s=890 e=343 n=343 c=B}

{w=343 s=979 e=774 n=343 c=B}

{w=343 s=1054 e=1054 n=979 c=B}

{w=343 s=1497 e=1054 n=343 c=B}

{w=365 s=890 e=890 n=262 c=B}

{w=516 s=343 e=343 n=343 c=B}

{w=516 s=516 e=516 n=516 c=B}

{w=516 s=890 e=565 n=516 c=B}

{w=565 s=890 e=343 n=516 c=B}

{w=692 s=215 e=215 n=215 c=B}

{w=692 s=239 e=692 n=692 c=N}

{w=692 s=262 e=215 n=215 c=B}

{w=692 s=692 e=692 n=692 c=B}

{w=692 s=996 e=262 n=692 c=B}

{w=748 s=7 e=692 n=7 c=B}

{w=748 s=239 e=748 n=239 c=N}

{w=748 s=692 e=692 n=7 c=N}

{w=748 s=748 e=748 n=748 c=B}

{w=748 s=994 e=748 n=169 c=N}

{w=748 s=1193 e=1445 n=1445 c=N}

{w=748 s=1253 e=1253 n=748 c=N}

{w=748 s=1254 e=1193 n=1254 c=N}

{w=748 s=1391 e=1847 n=1847 c=N}

{w=748 s=1445 e=748 n=748 c=N}

{w=748 s=1847 e=1847 n=1847 c=N}

{w=774 s=1054 e=215 n=343 c=B}

{w=867 s=169 e=944 n=169 c=N}

{w=867 s=867 e=239 n=239 c=N}

{w=867 s=944 e=867 n=169 c=N}

{w=867 s=1254 e=944 n=944 c=N}

{w=890 s=890 e=890 n=890 c=B}

{w=890 s=1165 e=1165 n=1483 c=B}

{w=890 s=1354 e=1664 n=1664 c=N}

{w=890 s=1483 e=1483 n=890 c=B}

{w=890 s=1497 e=1497 n=890 c=B}

{w=890 s=1664 e=1165 n=1847 c=B}

{w=890 s=1847 e=890 n=890 c=N}

{w=902 s=890 e=1253 n=516 c=B}

{w=944 s=169 e=994 n=994 c=N}

{w=944 s=239 e=169 n=169 c=N}

{w=944 s=944 e=994 n=944 c=N}

{w=944 s=1254 e=748 n=867 c=B}

{w=979 s=774 e=979 n=979 c=B}

{w=979 s=979 e=979 n=979 c=B}

{w=979 s=1054 e=1054 n=979 c=B}

{w=994 s=169 e=169 n=169 c=N}

{w=994 s=692 e=748 n=994 c=B}

{w=994 s=994 e=169 n=994 c=N}

{w=994 s=1396 e=169 n=944 c=N}

{w=996 s=262 e=239 n=692 c=B}

{w=996 s=692 e=996 n=692 c=B}

{w=996 s=1847 e=262 n=692 c=N}

{w=1054 s=1054 e=1054 n=1054 c=B}

{w=1054 s=1177 e=1177 n=1054 c=B}

{w=1054 s=1497 e=1054 n=343 c=B}

{w=1100 s=1823 e=1928 n=1177 c=N}

{w=1100 s=1928 e=1497 n=1354 c=N}

{w=1165 s=890 e=1165 n=890 c=N}

{w=1165 s=1054 e=1054 n=1054 c=B}

{w=1165 s=1100 e=1165 n=1354 c=B}

{w=1165 s=1165 e=1165 n=1165 c=B}

{w=1165 s=1177 e=1177 n=1054 c=B}

{w=1165 s=1354 e=1165 n=1165 c=N}

{w=1165 s=1497 e=1165 n=1497 c=B}

{w=1165 s=1928 e=1100 n=1100 c=N}

{w=1177 s=1100 e=1823 n=1177 c=N}

{w=1177 s=1165 e=1177 n=1165 c=B}

{w=1177 s=1177 e=1177 n=1177 c=B}

{w=1177 s=1823 e=1177 n=1177 c=B}

{w=1177 s=1928 e=1165 n=1165 c=N}

{w=1193 s=748 e=748 n=748 c=B}

{w=1193 s=890 e=1847 n=365 c=B}

{w=1193 s=1193 e=1193 n=1193 c=B}

{w=1193 s=1254 e=239 n=748 c=B}

{w=1193 s=1445 e=1193 n=1193 c=N}

{w=1193 s=1537 e=1847 n=1193 c=B}

{w=1193 s=1572 e=1572 n=1254 c=N}

{w=1193 s=1847 e=1847 n=1847 c=B}

{w=1193 s=1853 e=1193 n=1193 c=N}

{w=1193 s=2021 e=1537 n=1193 c=B}

{w=1196 s=1193 e=1193 n=1254 c=B}

{w=1253 s=890 e=890 n=516 c=B}

{w=1253 s=1193 e=1391 n=1391 c=N}

{w=1253 s=1391 e=1391 n=748 c=B}

{w=1254 s=239 e=748 n=1254 c=B}

{w=1254 s=748 e=1445 n=239 c=N}

{w=1254 s=1193 e=748 n=1445 c=N}

{w=1254 s=1196 e=1196 n=1254 c=B}

{w=1254 s=1254 e=1254 n=1254 c=N}

{w=1254 s=1445 e=1193 n=748 c=B}

{w=1254 s=1572 e=1193 n=1196 c=N}

{w=1254 s=1853 e=1966 n=1966 c=N}

{w=1254 s=1892 e=1853 n=1853 c=N}

{w=1254 s=1966 e=1853 n=1892 c=B}

{w=1354 s=1165 e=1165 n=1165 c=N}

{w=1354 s=1354 e=1354 n=1354 c=B}

{w=1354 s=1537 e=1354 n=1354 c=N}

{w=1354 s=1928 e=1541 n=1354 c=N}

{w=1391 s=262 e=262 n=262 c=N}

{w=1391 s=890 e=1193 n=262 c=N}

{w=1391 s=1193 e=1253 n=1253 c=N}

{w=1391 s=1253 e=1391 n=1391 c=N}

{w=1391 s=1391 e=1391 n=1391 c=B}

{w=1391 s=1847 e=890 n=890 c=N}

{w=1396 s=692 e=239 n=169 c=B}

{w=1396 s=944 e=169 n=944 c=N}

{w=1437 s=1437 e=1437 n=1437 c=N}

{w=1437 s=1823 e=1823 n=1823 c=N}

{w=1437 s=2035 e=1928 n=1928 c=N}

{w=1445 s=239 e=748 n=748 c=B}

{w=1445 s=748 e=748 n=748 c=N}

{w=1445 s=1193 e=1391 n=748 c=N}

{w=1483 s=1165 e=1165 n=1497 c=B}

{w=1497 s=1054 e=1054 n=1054 c=B}

{w=1497 s=1165 e=1497 n=1497 c=B}

{w=1497 s=1497 e=1497 n=1497 c=B}

{w=1497 s=1928 e=1165 n=1497 c=N}

{w=1537 s=1354 e=1354 n=1354 c=N}

{w=1537 s=1537 e=1537 n=1537 c=B}

{w=1537 s=1541 e=1928 n=1928 c=N}

{w=1537 s=1664 e=1928 n=1828 c=N}

{w=1537 s=1801 e=1928 n=1354 c=N}

{w=1537 s=1828 e=2029 n=2035 c=N}

{w=1537 s=1928 e=1801 n=1801 c=N}

{w=1537 s=1970 e=1537 n=1537 c=N}

{w=1537 s=2021 e=2021 n=1193 c=B}

{w=1537 s=2029 e=2035 n=1541 c=N}

{w=1537 s=2035 e=1928 n=2029 c=N}

{w=1541 s=1928 e=1664 n=1537 c=N}

{w=1572 s=1193 e=1853 n=1572 c=B}

{w=1572 s=1254 e=1853 n=1853 c=N}

{w=1572 s=1853 e=1193 n=1572 c=N}

{w=1664 s=1354 e=1165 n=890 c=N}

{w=1664 s=1928 e=1177 n=1354 c=N}

{w=1766 s=1766 e=1960 n=1966 c=N}

{w=1766 s=1958 e=1958 n=1958 c=N}

{w=1801 s=1823 e=1823 n=1801 c=N}

{w=1801 s=1928 e=1928 n=1928 c=N}

{w=1801 s=1960 e=1828 n=1970 c=B}

{w=1823 s=1437 e=1823 n=1823 c=N}

{w=1823 s=1801 e=1823 n=1863 c=N}

{w=1823 s=1823 e=1823 n=1823 c=B}

{w=1823 s=1863 e=1437 n=1437 c=N}

{w=1823 s=2029 e=2029 n=1437 c=N}

{w=1828 s=1766 e=1828 n=1958 c=N}

{w=1828 s=1828 e=1828 n=1828 c=N}

{w=1828 s=1853 e=1960 n=1853 c=N}

{w=1828 s=1960 e=1537 n=1537 c=N}

{w=1828 s=2035 e=2035 n=1928 c=N}

{w=1843 s=1537 e=1843 n=1664 c=N}

{w=1843 s=1928 e=1828 n=1928 c=N}

{w=1843 s=1960 e=1970 n=1970 c=N}

{w=1847 s=239 e=262 n=262 c=N}

{w=1847 s=262 e=1391 n=1847 c=B}

{w=1847 s=748 e=1391 n=239 c=B}

{w=1847 s=890 e=890 n=890 c=N}

{w=1847 s=1354 e=1537 n=1664 c=N}

{w=1847 s=1391 e=1391 n=748 c=N}

{w=1847 s=1537 e=1847 n=1847 c=B}

{w=1847 s=1664 e=1354 n=890 c=N}

{w=1847 s=1847 e=1847 n=1847 c=B}

{w=1853 s=1193 e=1193 n=1193 c=N}

{w=1853 s=1828 e=1958 n=1958 c=B}

{w=1853 s=1853 e=1853 n=1853 c=B}

{w=1853 s=1892 e=1892 n=1853 c=N}
{w=1853 s=1958 e=1853 n=1853 c=N}
{w=1853 s=1960 e=2021 n=2021 c=N}
{w=1853 s=1966 e=1193 n=1853 c=B}
{w=1853 s=2021 e=2021 n=2021 c=N}
{w=1892 s=1193 e=1853 n=1853 c=B}
{w=1892 s=1853 e=1193 n=1892 c=N}
{w=1892 s=1892 e=1966 n=1966 c=B}
{w=1928 s=1100 e=1823 n=1843 c=N}
{w=1928 s=1165 e=1928 n=1177 c=N}
{w=1928 s=1177 e=1177 n=1165 c=N}
{w=1928 s=1437 e=1823 n=1823 c=N}
{w=1928 s=1801 e=1928 n=1928 c=N}
{w=1928 s=1823 e=1437 n=1100 c=N}
{w=1928 s=1828 e=1928 n=1928 c=N}
{w=1928 s=1843 e=1100 n=1177 c=N}
{w=1928 s=1928 e=1928 n=1928 c=B}
{w=1928 s=2029 e=1928 n=1801 c=N}
{w=1928 s=2035 e=2029 n=2029 c=N}
{w=1958 s=1766 e=1958 n=1958 c=N}
{w=1958 s=1828 e=1853 n=1853 c=N}
{w=1958 s=1853 e=1853 n=1853 c=N}
{w=1958 s=1958 e=1958 n=1958 c=B}
{w=1958 s=1960 e=1828 n=1766 c=N}
{w=1958 s=1966 e=1853 n=1254 c=N}
{w=1960 s=1537 e=1843 n=1537 c=N}
{w=1960 s=1766 e=1958 n=1960 c=N}
{w=1960 s=1853 e=1958 n=1958 c=B}
{w=1960 s=1958 e=1766 n=1766 c=N}
{w=1960 s=1960 e=1960 n=1960 c=N}
{w=1960 s=1970 e=1960 n=1970 c=N}
{w=1966 s=1193 e=1892 n=1853 c=N}
{w=1966 s=1254 e=1853 n=1254 c=N}
{w=1966 s=1853 e=1892 n=1892 c=N}
{w=1966 s=1960 e=1853 n=1766 c=N}
{w=1970 s=1537 e=1537 n=1537 c=N}
{w=1970 s=1960 e=1801 n=1537 c=N}
{w=1970 s=1970 e=1970 n=1970 c=B}
{w=2021 s=1193 e=1537 n=1537 c=N}
{w=2021 s=1537 e=1537 n=2021 c=B}
{w=2021 s=1843 e=1970 n=1853 c=N}
{w=2021 s=1853 e=1970 n=2021 c=B}
{w=2021 s=1960 e=1843 n=1843 c=B}
{w=2021 s=1970 e=1970 n=1193 c=B}
{w=2021 s=2021 e=1537 n=1970 c=N}
{w=2029 s=1823 e=1801 n=1823 c=N}
{w=2029 s=1928 e=1928 n=1928 c=N}
{w=2029 s=2035 e=1437 n=1801 c=B}
{w=2035 s=1437 e=1437 n=1928 c=N}
{w=2035 s=1928 e=1537 n=1928 c=N}
{w=2035 s=2035 e=2035 n=2035 c=N}

A.14. Triángulo de Sierpinski 64×64

Tipos de mosaicos = 1161

{w=0 s=0 e=1391 n=193 c=B}
{w=0 s=193 e=591 n=591 c=N}
{w=0 s=482 e=1391 n=1606 c=N}
{w=0 s=1606 e=1391 n=0 c=B}

{w=6 s=6 e=6 n=6 c=B}

{w=6 s=193 e=591 n=591 c=B}

{w=6 s=457 e=1345 n=457 c=N}

{w=6 s=591 e=591 n=6 c=B}

{w=6 s=1345 e=6 n=6 c=N}

{w=6 s=1373 e=591 n=2369 c=N}

{w=6 s=2369 e=457 n=457 c=N}

{w=94 s=907 e=512 n=512 c=B}

{w=99 s=99 e=99 n=99 c=B}

{w=99 s=114 e=907 n=114 c=B}

{w=99 s=512 e=512 n=99 c=B}

{w=99 s=885 e=764 n=375 c=B}

{w=99 s=907 e=114 n=99 c=B}

{w=114 s=907 e=907 n=114 c=B}

{w=128 s=800 e=800 n=1547 c=B}

{w=186 s=703 e=568 n=703 c=B}

{w=188 s=568 e=188 n=568 c=B}

{w=188 s=703 e=703 n=703 c=B}

{w=193 s=0 e=193 n=0 c=N}

{w=193 s=6 e=193 n=6 c=N}

{w=193 s=591 e=591 n=591 c=N}

{w=193 s=1345 e=193 n=1373 c=N}

{w=193 s=1373 e=193 n=193 c=N}

{w=193 s=1391 e=3411 n=591 c=N}

{w=193 s=1606 e=591 n=591 c=B}

{w=294 s=703 e=703 n=1583 c=B}

{w=343 s=6 e=343 n=343 c=B}

{w=343 s=343 e=343 n=343 c=B}

{w=343 s=457 e=457 n=457 c=N}

{w=343 s=591 e=591 n=343 c=B}

{w=362 s=842 e=842 n=362 c=B}

{w=362 s=1391 e=1391 n=752 c=B}

{w=375 s=99 e=99 n=99 c=B}

{w=375 s=375 e=375 n=375 c=B}

{w=375 s=885 e=99 n=99 c=B}

{w=414 s=414 e=414 n=414 c=B}

{w=414 s=429 e=568 n=414 c=B}

{w=414 s=568 e=568 n=414 c=B}

{w=414 s=800 e=414 n=414 c=B}

{w=414 s=922 e=800 n=414 c=B}

{w=429 s=429 e=429 n=429 c=B}

{w=429 s=568 e=568 n=568 c=B}

{w=429 s=703 e=186 n=568 c=B}

{w=429 s=800 e=703 n=429 c=B}

{w=429 s=1583 e=1583 n=568 c=B}

{w=429 s=1693 e=1693 n=800 c=B}

{w=457 s=6 e=1345 n=343 c=B}

{w=457 s=343 e=343 n=343 c=N}

{w=457 s=457 e=457 n=343 c=N}

{w=457 s=2369 e=1345 n=6 c=B}

{w=482 s=1884 e=1391 n=1884 c=N}

{w=512 s=99 e=907 n=512 c=B}

{w=512 s=907 e=907 n=99 c=B}

{w=568 s=429 e=568 n=568 c=B}

{w=568 s=568 e=568 n=568 c=B}

{w=568 s=703 e=188 n=568 c=B}

{w=568 s=800 e=429 n=568 c=B}

{w=591 s=0 e=1391 n=591 c=B}

{w=591 s=6 e=591 n=193 c=N}

{w=591 s=193 e=591 n=591 c=N}

{w=591 s=362 e=362 n=752 c=B}

{w=591 s=591 e=591 n=591 c=B}

{w=591 s=752 e=752 n=752 c=B}

{w=591 s=842 e=719 n=591 c=B}

{w=591 s=1373 e=193 n=1345 c=N}

{w=591 s=1391 e=1606 n=591 c=B}

{w=591 s=1606 e=193 n=6 c=B}

{w=614 s=375 e=375 n=375 c=B}

{w=614 s=614 e=614 n=614 c=B}

{w=614 s=885 e=614 n=614 c=B}

{w=614 s=1391 e=885 n=752 c=B}

{w=703 s=703 e=703 n=703 c=B}

{w=703 s=1693 e=1191 n=429 c=B}

{w=703 s=1856 e=703 n=703 c=B}

{w=703 s=2099 e=703 n=703 c=B}

{w=719 s=1391 e=842 n=362 c=B}

{w=752 s=362 e=752 n=752 c=B}

{w=752 s=614 e=614 n=614 c=B}

{w=752 s=752 e=752 n=752 c=B}

{w=752 s=842 e=362 n=362 c=B}

{w=752 s=885 e=885 n=614 c=B}

{w=752 s=1391 e=614 n=752 c=B}

{w=764 s=885 e=375 n=375 c=B}

{w=764 s=907 e=766 n=99 c=B}

{w=766 s=907 e=94 n=99 c=B}

{w=800 s=429 e=429 n=429 c=B}

{w=800 s=703 e=703 n=800 c=B}

{w=800 s=800 e=800 n=800 c=B}

{w=800 s=922 e=922 n=414 c=B}

{w=800 s=1191 e=1191 n=800 c=B}

{w=800 s=1563 e=800 n=800 c=B}

{w=800 s=1693 e=1693 n=429 c=B}

{w=800 s=1836 e=1693 n=1191 c=B}

{w=842 s=1391 e=1391 n=842 c=B}

{w=875 s=885 e=1391 n=375 c=B}

{w=875 s=1398 e=1398 n=885 c=B}

{w=875 s=1646 e=875 n=875 c=B}

{w=885 s=99 e=764 n=99 c=B}

{w=885 s=375 e=375 n=614 c=B}

{w=885 s=875 e=875 n=375 c=B}

{w=885 s=885 e=885 n=885 c=B}

{w=885 s=907 e=907 n=99 c=B}

{w=885 s=1391 e=885 n=885 c=B}

{w=885 s=1398 e=1546 n=885 c=B}

{w=885 s=1546 e=907 n=885 c=B}

{w=885 s=1646 e=875 n=1391 c=B}

{w=900 s=2693 e=2169 n=907 c=B}

{w=907 s=128 e=800 n=907 c=B}

{w=907 s=414 e=414 n=414 c=B}

{w=907 s=800 e=922 n=922 c=B}

{w=907 s=907 e=907 n=907 c=B}

{w=907 s=922 e=800 n=414 c=B}

{w=907 s=1420 e=800 n=1563 c=B}

{w=907 s=1547 e=800 n=128 c=B}

{w=907 s=1563 e=128 n=907 c=B}

{w=907 s=1684 e=1684 n=907 c=B}

{w=907 s=1939 e=800 n=2061 c=B}

{w=907 s=2061 e=800 n=1420 c=B}

{w=907 s=2693 e=1939 n=907 c=B}

{w=907 s=3230 e=1939 n=1684 c=B}

{w=922 s=800 e=800 n=922 c=B}

{w=990 s=1646 e=1378 n=1391 c=B}

{w=990 s=2004 e=1646 n=482 c=B}

{w=1000 s=1646 e=1646 n=1000 c=B}

{w=1057 s=2693 e=2693 n=1684 c=B}

{w=1065 s=703 e=800 n=568 c=B}

{w=1181 s=1181 e=2099 n=703 c=B}

{w=1181 s=1955 e=1181 n=1181 c=B}

{w=1181 s=2099 e=2099 n=2099 c=B}

{w=1191 s=703 e=294 n=429 c=B}

{w=1191 s=1693 e=1693 n=800 c=B}

{w=1224 s=1606 e=1606 n=1373 c=N}

{w=1224 s=2369 e=1373 n=2263 c=N}

{w=1345 s=6 e=6 n=6 c=N}

{w=1345 s=193 e=1345 n=6 c=N}

{w=1345 s=1345 e=1345 n=1345 c=B}

{w=1345 s=1373 e=591 n=1345 c=B}

{w=1345 s=2369 e=1345 n=1345 c=N}

{w=1355 s=1391 e=482 n=1606 c=B}

{w=1373 s=6 e=1373 n=2369 c=N}

{w=1373 s=1224 e=1606 n=1373 c=B}

{w=1373 s=1373 e=1373 n=1373 c=B}

{w=1373 s=1606 e=591 n=1345 c=B}

{w=1373 s=2369 e=1606 n=1224 c=B}

{w=1378 s=1378 e=1606 n=3411 c=B}

{w=1378 s=1646 e=1000 n=1391 c=B}

{w=1378 s=2004 e=1646 n=482 c=N}

{w=1378 s=2915 e=2915 n=2263 c=N}

{w=1378 s=3411 e=1378 n=1378 c=B}

{w=1391 s=482 e=990 n=1391 c=B}

{w=1391 s=591 e=3411 n=591 c=N}

{w=1391 s=875 e=885 n=875 c=B}

{w=1391 s=885 e=99 n=375 c=B}

{w=1391 s=1000 e=1646 n=1391 c=B}

{w=1391 s=1378 e=1378 n=1378 c=B}

{w=1391 s=1391 e=1391 n=1391 c=B}

{w=1391 s=1398 e=885 n=1391 c=B}

{w=1391 s=1646 e=1398 n=1391 c=B}

{w=1391 s=1884 e=1884 n=482 c=B}

{w=1391 s=2004 e=1391 n=2915 c=N}

{w=1391 s=2771 e=2915 n=2004 c=N}

{w=1391 s=2915 e=2004 n=3411 c=N}

{w=1391 s=3411 e=1391 n=1391 c=N}

{w=1398 s=907 e=907 n=907 c=B}

{w=1398 s=1398 e=1398 n=1398 c=B}

{w=1398 s=1646 e=1515 n=1391 c=B}

{w=1398 s=2285 e=2693 n=1398 c=B}

{w=1398 s=2693 e=3709 n=907 c=B}

{w=1515 s=1646 e=1646 n=1398 c=B}

{w=1546 s=907 e=907 n=1546 c=B}

{w=1546 s=1398 e=1398 n=885 c=B}

{w=1563 s=800 e=1693 n=1181 c=B}

{w=1563 s=1181 e=1693 n=1836 c=B}

{w=1563 s=1563 e=1563 n=1563 c=B}

{w=1563 s=1693 e=1563 n=1836 c=B}

{w=1563 s=1836 e=1693 n=800 c=B}

{w=1563 s=1939 e=1939 n=907 c=B}

{w=1563 s=1955 e=1955 n=1693 c=B}

{w=1563 s=2083 e=3349 n=1563 c=B}

{w=1563 s=2322 e=2083 n=1563 c=B}

{w=1563 s=2832 e=1955 n=2083 c=B}

{w=1563 s=3349 e=1955 n=1563 c=B}

{w=1583 s=703 e=1065 n=568 c=B}

{w=1606 s=0 e=193 n=193 c=N}

{w=1606 s=193 e=0 n=193 c=N}

{w=1606 s=482 e=0 n=0 c=N}

{w=1606 s=1224 e=1606 n=1373 c=N}

{w=1606 s=1355 e=2632 n=1606 c=B}

{w=1606 s=1373 e=2263 n=1373 c=B}

{w=1606 s=1378 e=2771 n=2771 c=B}

{w=1606 s=1391 e=842 n=591 c=B}

{w=1606 s=1606 e=1606 n=1606 c=B}

{w=1606 s=1884 e=0 n=482 c=N}

{w=1606 s=2263 e=1606 n=1606 c=N}

{w=1606 s=2369 e=1606 n=1606 c=N}

{w=1606 s=2632 e=2632 n=2263 c=N}

{w=1606 s=3405 e=3137 n=1606 c=N}

{w=1606 s=3655 e=1606 n=2632 c=N}

{w=1646 s=1398 e=1398 n=1398 c=B}

{w=1646 s=1646 e=1646 n=1646 c=B}

{w=1646 s=2004 e=1646 n=2004 c=B}

{w=1646 s=2285 e=2285 n=1646 c=B}

{w=1646 s=2915 e=1646 n=1646 c=N}

{w=1646 s=3055 e=2285 n=1646 c=B}

{w=1646 s=3318 e=3318 n=1646 c=B}

{w=1646 s=3953 e=2285 n=1646 c=N}

{w=1646 s=4583 e=2285 n=3953 c=N}

{w=1684 s=1939 e=1563 n=907 c=B}

{w=1684 s=2693 e=1939 n=3230 c=B}

{w=1693 s=703 e=703 n=703 c=B}

{w=1693 s=1181 e=1181 n=1693 c=B}

{w=1693 s=1693 e=1693 n=1693 c=B}

{w=1693 s=1955 e=2099 n=1693 c=B}

{w=1836 s=3251 e=3251 n=1836 c=B}

{w=1856 s=1856 e=1856 n=1856 c=B}

{w=1856 s=3251 e=2097 n=2099 c=B}

{w=1856 s=3774 e=1856 n=1856 c=B}

{w=1884 s=1391 e=1391 n=1391 c=N}

{w=1884 s=3405 e=1884 n=1884 c=N}

{w=1884 s=3411 e=1391 n=1884 c=N}

{w=1939 s=1563 e=1563 n=1563 c=B}

{w=1939 s=1939 e=1939 n=1939 c=B}

{w=1939 s=2693 e=1057 n=907 c=B}

{w=1939 s=3230 e=1939 n=1939 c=B}

{w=1955 s=1836 e=2090 n=2099 c=B}

{w=1955 s=1955 e=1955 n=1955 c=B}

{w=1955 s=2099 e=2099 n=2099 c=B}

{w=1955 s=2322 e=1955 n=2322 c=B}

{w=1955 s=2985 e=2985 n=1955 c=B}

{w=1955 s=3112 e=3251 n=1955 c=B}

{w=1955 s=3251 e=1836 n=1955 c=B}

{w=1955 s=3498 e=3251 n=1955 c=B}

{w=2004 s=1378 e=2004 n=2004 c=B}

{w=2004 s=1391 e=1391 n=1391 c=N}

{w=2004 s=2004 e=2004 n=2004 c=B}

{w=2004 s=2263 e=1646 n=2915 c=N}

{w=2004 s=2915 e=1646 n=2004 c=N}

{w=2083 s=2322 e=2832 n=1563 c=B}

{w=2090 s=3251 e=1856 n=2099 c=B}

{w=2097 s=3251 e=3251 n=2099 c=B}

{w=2099 s=703 e=703 n=703 c=B}

{w=2099 s=1856 e=1856 n=703 c=B}

{w=2099 s=1955 e=1181 n=1693 c=B}

{w=2099 s=2097 e=1856 n=2099 c=B}

{w=2099 s=2099 e=2099 n=2099 c=B}

{w=2099 s=2347 e=1856 n=2097 c=B}

{w=2099 s=3251 e=2347 n=2347 c=B}

{w=2157 s=1646 e=2693 n=1646 c=B}

{w=2169 s=2693 e=2674 n=907 c=B}

{w=2199 s=2322 e=3347 n=3230 c=B}

{w=2249 s=3137 e=3405 n=2369 c=N}

{w=2249 s=3405 e=1606 n=2369 c=N}

{w=2249 s=3915 e=3411 n=3405 c=N}

{w=2263 s=1606 e=1606 n=1224 c=B}

{w=2263 s=2915 e=1646 n=2915 c=N}

{w=2263 s=3405 e=2771 n=1355 c=N}

{w=2285 s=1646 e=3055 n=1398 c=B}

{w=2285 s=2285 e=2285 n=2285 c=B}

{w=2285 s=2693 e=2693 n=2693 c=B}

{w=2285 s=2915 e=2915 n=2915 c=N}

{w=2285 s=2937 e=4083 n=4583 c=B}

{w=2285 s=4083 e=2285 n=2285 c=N}

{w=2285 s=4583 e=2285 n=2285 c=N}

{w=2322 s=1955 e=1955 n=2983 c=B}

{w=2322 s=2322 e=2322 n=2322 c=B}

{w=2322 s=2983 e=1955 n=1955 c=B}

{w=2322 s=3091 e=3091 n=2322 c=B}

{w=2322 s=3112 e=2983 n=1955 c=B}

{w=2322 s=3602 e=2322 n=2322 c=B}

{w=2347 s=3251 e=1856 n=1856 c=B}

{w=2369 s=6 e=1345 n=2369 c=B}

{w=2369 s=1224 e=1373 n=1373 c=N}

{w=2369 s=1345 e=1345 n=1345 c=B}

{w=2369 s=1373 e=6 n=6 c=N}

{w=2369 s=1606 e=1224 n=1224 c=N}

{w=2369 s=2249 e=1606 n=2632 c=N}

{w=2369 s=2263 e=2369 n=6 c=N}

{w=2369 s=2369 e=2369 n=2369 c=N}

{w=2369 s=2632 e=1606 n=2369 c=B}

{w=2369 s=3137 e=1606 n=2249 c=B}

{w=2369 s=3405 e=3405 n=3405 c=N}

{w=2369 s=3655 e=2249 n=3137 c=N}

{w=2579 s=2322 e=2199 n=3230 c=B}

{w=2632 s=1355 e=1355 n=1606 c=N}

{w=2632 s=1606 e=1606 n=1606 c=N}

{w=2632 s=3405 e=2263 n=1355 c=N}

{w=2650 s=4583 e=4583 n=3411 c=N}

{w=2671 s=4583 e=5206 n=3525 c=N}

{w=2671 s=4689 e=2671 n=3411 c=N}

{w=2671 s=5206 e=4300 n=3411 c=N}

{w=2671 s=5741 e=4689 n=4689 c=N}

{w=2674 s=2693 e=2693 n=907 c=B}

{w=2693 s=1646 e=1646 n=3055 c=B}

{w=2693 s=2199 e=2579 n=3230 c=B}

{w=2693 s=2285 e=2693 n=2285 c=B}

{w=2693 s=2322 e=3709 n=2693 c=B}

{w=2693 s=2693 e=2693 n=2693 c=B}

{w=2693 s=3198 e=3602 n=2693 c=B}

{w=2693 s=3230 e=3230 n=3230 c=B}

{w=2693 s=3339 e=2322 n=3709 c=B}

{w=2693 s=3602 e=3198 n=3198 c=B}

{w=2693 s=3709 e=2322 n=2322 c=B}

{w=2693 s=4610 e=3198 n=2693 c=B}

{w=2693 s=4723 e=3198 n=2693 c=N}

{w=2693 s=5109 e=5109 n=2693 c=B}

{w=2761 s=2249 e=3405 n=3405 c=B}

{w=2761 s=2761 e=2761 n=2761 c=N}

{w=2761 s=3411 e=3411 n=3013 c=N}

{w=2761 s=3525 e=3405 n=2761 c=B}

{w=2761 s=3915 e=3525 n=3525 c=N}

{w=2771 s=1378 e=2004 n=2004 c=B}

{w=2771 s=3405 e=1884 n=1391 c=N}

{w=2832 s=2322 e=1955 n=2832 c=B}

{w=2887 s=3525 e=3525 n=3915 c=B}

{w=2887 s=5206 e=4551 n=4551 c=N}

{w=2915 s=1646 e=1646 n=1646 c=N}

{w=2915 s=2004 e=3411 n=1391 c=N}

{w=2915 s=2285 e=2915 n=2915 c=N}

{w=2915 s=2915 e=2915 n=2915 c=B}

{w=2915 s=3318 e=3953 n=3318 c=N}

{w=2915 s=4583 e=2937 n=2285 c=N}

{w=2937 s=4083 e=2285 n=2937 c=N}

{w=2937 s=4583 e=3953 n=2915 c=N}

{w=2937 s=5741 e=4083 n=2937 c=N}

{w=2983 s=3112 e=3112 n=2985 c=B}

{w=2985 s=3112 e=1955 n=1955 c=B}

{w=2985 s=3251 e=3251 n=3498 c=B}

{w=3013 s=2761 e=2761 n=2761 c=N}

{w=3013 s=3013 e=3411 n=3405 c=N}

{w=3013 s=3411 e=3411 n=3915 c=N}

{w=3013 s=3525 e=5188 n=3915 c=N}

{w=3013 s=3915 e=2761 n=3013 c=B}

{w=3055 s=1646 e=1398 n=1398 c=B}

{w=3091 s=3091 e=3091 n=3091 c=B}

{w=3091 s=3112 e=3112 n=3112 c=B}

{w=3091 s=3602 e=3091 n=3091 c=B}

{w=3091 s=4226 e=4226 n=3091 c=B}

{w=3112 s=3112 e=3112 n=3112 c=B}

{w=3112 s=3251 e=3251 n=3498 c=B}

{w=3112 s=3498 e=3251 n=3251 c=B}

{w=3112 s=3504 e=3504 n=3749 c=B}

{w=3112 s=3749 e=3990 n=3990 c=B}

{w=3112 s=3884 e=3504 n=3498 c=B}

{w=3112 s=3990 e=3749 n=3112 c=B}

{w=3112 s=4226 e=3504 n=3112 c=B}

{w=3112 s=4741 e=3602 n=5649 c=B}

{w=3137 s=2249 e=3655 n=3655 c=N}

{w=3137 s=2369 e=3405 n=3405 c=B}

{w=3137 s=3137 e=3137 n=3137 c=N}

{w=3137 s=3405 e=2632 n=3655 c=N}

{w=3137 s=3655 e=2249 n=3137 c=B}

{w=3198 s=3602 e=3602 n=3198 c=B}

{w=3198 s=4723 e=3602 n=5109 c=N}

{w=3230 s=1563 e=1563 n=1563 c=B}

{w=3230 s=2322 e=1955 n=1563 c=B}

{w=3230 s=3230 e=3230 n=3230 c=B}

{w=3251 s=1856 e=1856 n=1856 c=B}

{w=3251 s=2615 e=3774 n=1856 c=B}

{w=3251 s=2869 e=3774 n=4032 c=B}

{w=3251 s=3112 e=3251 n=2985 c=B}

{w=3251 s=3129 e=3774 n=2615 c=B}

{w=3251 s=3251 e=3251 n=3251 c=B}

{w=3251 s=3504 e=3251 n=3251 c=B}

{w=3251 s=3774 e=3774 n=3129 c=B}

{w=3251 s=4032 e=4032 n=3251 c=B}

{w=3318 s=1646 e=2157 n=1646 c=B}

{w=3318 s=3318 e=1646 n=3318 c=B}

{w=3318 s=4583 e=1646 n=1646 c=N}

{w=3339 s=3602 e=3602 n=3339 c=B}

{w=3347 s=2322 e=3230 n=1563 c=B}

{w=3349 s=1955 e=1955 n=3349 c=B}

{w=3405 s=1378 e=3411 n=3411 c=B}

{w=3405 s=1391 e=1391 n=1378 c=B}

{w=3405 s=1884 e=3411 n=3405 c=B}

{w=3405 s=2249 e=3915 n=2249 c=N}

{w=3405 s=2369 e=3405 n=3405 c=N}

{w=3405 s=2761 e=3405 n=2249 c=B}

{w=3405 s=3137 e=3405 n=3655 c=B}

{w=3405 s=3405 e=3405 n=3405 c=B}

{w=3405 s=3411 e=3411 n=4173 c=B}

{w=3405 s=3915 e=3013 n=2369 c=N}

{w=3405 s=4173 e=3405 n=1884 c=B}

{w=3411 s=591 e=1391 n=0 c=B}

{w=3411 s=1378 e=3411 n=3411 c=B}

{w=3411 s=1391 e=1391 n=591 c=N}

{w=3411 s=1646 e=2263 n=2915 c=N}

{w=3411 s=2004 e=1378 n=3411 c=N}

{w=3411 s=2263 e=2915 n=1646 c=N}

{w=3411 s=2285 e=2285 n=2263 c=N}

{w=3411 s=2650 e=2650 n=2937 c=N}

{w=3411 s=2671 e=3525 n=3525 c=N}

{w=3411 s=2771 e=2004 n=2771 c=B}

{w=3411 s=2915 e=2915 n=2285 c=N}

{w=3411 s=2937 e=4583 n=4054 c=N}

{w=3411 s=3013 e=3411 n=3405 c=B}

{w=3411 s=3411 e=3411 n=3411 c=B}

{w=3411 s=3418 e=4583 n=4358 c=N}

{w=3411 s=3525 e=3411 n=3411 c=N}

{w=3411 s=3810 e=3525 n=3525 c=B}

{w=3411 s=3915 e=3013 n=3915 c=N}

{w=3411 s=4054 e=4583 n=2915 c=N}

{w=3411 s=4168 e=3411 n=2671 c=N}

{w=3411 s=4173 e=4173 n=3405 c=B}

{w=3411 s=4300 e=3411 n=3411 c=N}

{w=3411 s=4358 e=4358 n=4583 c=N}

{w=3411 s=4433 e=4433 n=3418 c=N}

{w=3411 s=4583 e=4583 n=2650 c=N}

{w=3411 s=4689 e=4433 n=4433 c=N}

{w=3411 s=5206 e=4419 n=3411 c=N}

{w=3418 s=4433 e=4689 n=4583 c=N}

{w=3498 s=3498 e=3498 n=3498 c=B}

{w=3498 s=3504 e=4393 n=4776 c=B}

{w=3498 s=4393 e=4393 n=3504 c=B}

{w=3498 s=4741 e=5401 n=4741 c=B}

{w=3498 s=4776 e=4776 n=4776 c=B}

{w=3498 s=5031 e=5426 n=3498 c=B}

{w=3498 s=5426 e=4393 n=3498 c=B}

{w=3498 s=5670 e=6076 n=6049 c=B}

{w=3498 s=6049 e=6076 n=5031 c=B}

{w=3498 s=6076 e=6076 n=5670 c=B}

{w=3498 s=6179 e=6076 n=3498 c=B}

{w=3498 s=6295 e=3498 n=3498 c=N}

{w=3498 s=6424 e=3498 n=3498 c=N}

{w=3498 s=6437 e=6440 n=3498 c=B}

{w=3498 s=6440 e=6179 n=3498 c=B}

{w=3504 s=2869 e=4032 n=3504 c=B}

{w=3504 s=3498 e=3504 n=3504 c=B}

{w=3504 s=3504 e=3504 n=3504 c=B}

{w=3504 s=4032 e=4032 n=2869 c=B}

{w=3504 s=4776 e=4032 n=3504 c=B}

{w=3525 s=2887 e=3525 n=3525 c=N}

{w=3525 s=3013 e=3915 n=2761 c=N}

{w=3525 s=3411 e=3411 n=3411 c=N}

{w=3525 s=3525 e=3525 n=3525 c=B}

{w=3525 s=3810 e=3810 n=3411 c=N}

{w=3525 s=3915 e=2249 n=2249 c=N}

{w=3525 s=4050 e=3525 n=4300 c=N}

{w=3525 s=4168 e=3411 n=4050 c=N}

{w=3525 s=4300 e=3411 n=3810 c=N}

{w=3525 s=4551 e=3525 n=2887 c=N}

{w=3525 s=5188 e=4173 n=3525 c=N}

{w=3525 s=5206 e=4173 n=4551 c=N}

{w=3525 s=5569 e=5569 n=3525 c=N}

{w=3525 s=6210 e=6210 n=5569 c=N}

{w=3602 s=2322 e=2322 n=2322 c=B}

{w=3602 s=3091 e=3091 n=2322 c=B}

{w=3602 s=3112 e=3112 n=4226 c=B}

{w=3602 s=3198 e=3339 n=3198 c=B}

{w=3602 s=3339 e=3602 n=2322 c=B}

{w=3602 s=3602 e=3602 n=3602 c=B}

{w=3602 s=4109 e=4999 n=4999 c=B}

{w=3602 s=4226 e=3112 n=3602 c=B}

{w=3602 s=4255 e=4255 n=3602 c=B}

{w=3602 s=4372 e=4372 n=3602 c=B}

{w=3602 s=4610 e=3602 n=3602 c=B}

{w=3602 s=4741 e=4741 n=4255 c=B}

{w=3602 s=4976 e=3602 n=3602 c=N}

{w=3602 s=4999 e=4999 n=3602 c=B}

{w=3602 s=5133 e=4976 n=4610 c=N}

{w=3602 s=5401 e=3091 n=3602 c=B}

{w=3602 s=5649 e=5649 n=4255 c=B}

{w=3655 s=2761 e=3405 n=3137 c=N}

{w=3655 s=3137 e=3405 n=2369 c=N}

{w=3655 s=3405 e=2369 n=3137 c=N}

{w=3709 s=2322 e=2322 n=2199 c=B}

{w=3709 s=2693 e=900 n=907 c=B}

{w=3709 s=3602 e=2322 n=3339 c=B}

{w=3749 s=3504 e=3504 n=3884 c=B}

{w=3749 s=3749 e=3504 n=3990 c=B}

{w=3774 s=3774 e=3774 n=3774 c=B}

{w=3774 s=4032 e=3774 n=3774 c=B}

{w=3774 s=4797 e=4797 n=3774 c=B}

{w=3810 s=3411 e=4300 n=3810 c=N}

{w=3915 s=3013 e=3915 n=5188 c=B}

{w=3915 s=3405 e=3411 n=3405 c=B}

{w=3915 s=3411 e=3525 n=3525 c=B}

{w=3915 s=3525 e=3525 n=3411 c=B}

{w=3915 s=3915 e=3915 n=3915 c=B}

{w=3915 s=5188 e=3915 n=3915 c=N}

{w=3938 s=5206 e=5480 n=3938 c=B}

{w=3938 s=5480 e=5480 n=4689 c=B}

{w=3953 s=1646 e=1646 n=2915 c=B}

{w=3953 s=2915 e=1646 n=3318 c=N}

{w=3953 s=3318 e=1646 n=3318 c=B}

{w=3953 s=4583 e=3318 n=3318 c=N}

{w=3953 s=4723 e=4228 n=4083 c=N}

{w=3953 s=4866 e=4866 n=3953 c=B}

{w=3990 s=3504 e=3504 n=3749 c=B}

{w=4032 s=3774 e=3774 n=3774 c=B}

{w=4032 s=4032 e=4032 n=4032 c=B}

{w=4032 s=4393 e=4393 n=4032 c=B}

{w=4032 s=4776 e=4032 n=4776 c=B}

{w=4032 s=4797 e=3774 n=3774 c=B}

{w=4083 s=2285 e=2285 n=2285 c=N}

{w=4083 s=4083 e=4083 n=4083 c=B}

{w=4083 s=4583 e=4723 n=4083 c=B}

{w=4083 s=4723 e=4228 n=2285 c=N}

{w=4083 s=5109 e=4723 n=4723 c=N}

{w=4109 s=3602 e=3602 n=4372 c=B}

{w=4168 s=3525 e=2887 n=3915 c=N}

{w=4168 s=4583 e=5741 n=4358 c=N}

{w=4168 s=5206 e=2671 n=4168 c=N}

{w=4173 s=3411 e=1378 n=1391 c=B}

{w=4173 s=3525 e=4168 n=5188 c=N}

{w=4173 s=3915 e=3915 n=3915 c=N}

{w=4173 s=4173 e=4173 n=3525 c=B}

{w=4173 s=4551 e=4551 n=4551 c=N}

{w=4173 s=5188 e=5569 n=5569 c=B}

{w=4173 s=5206 e=4168 n=3525 c=N}

{w=4173 s=6210 e=6210 n=5188 c=B}

{w=4226 s=3112 e=3112 n=3091 c=B}

{w=4226 s=3498 e=3504 n=4226 c=B}

{w=4226 s=4741 e=4226 n=3112 c=B}

{w=4226 s=5649 e=4226 n=3112 c=B}

{w=4228 s=2693 e=2285 n=2693 c=N}

{w=4228 s=4723 e=5741 n=4083 c=N}

{w=4255 s=4226 e=4226 n=3112 c=B}

{w=4255 s=4255 e=4255 n=3112 c=B}

{w=4255 s=4741 e=3112 n=3602 c=B}

{w=4300 s=3411 e=3525 n=3411 c=N}

{w=4300 s=5206 e=3411 n=4300 c=N}

{w=4358 s=3411 e=3418 n=3418 c=N}

{w=4358 s=4358 e=4433 n=4433 c=N}

{w=4358 s=4583 e=4168 n=4433 c=N}

{w=4358 s=5741 e=4583 n=4583 c=N}

{w=4372 s=3091 e=3091 n=5401 c=B}

{w=4372 s=3602 e=4109 n=4372 c=B}

{w=4372 s=4372 e=3091 n=4372 c=B}

{w=4372 s=4741 e=4741 n=4226 c=B}
{w=4393 s=4032 e=4797 n=4797 c=B}
{w=4393 s=4393 e=4393 n=4393 c=B}
{w=4393 s=4543 e=4797 n=5426 c=B}
{w=4393 s=4797 e=4797 n=4032 c=B}
{w=4393 s=5054 e=5054 n=4543 c=B}
{w=4393 s=5306 e=5054 n=4393 c=B}
{w=4393 s=5426 e=4797 n=4393 c=B}
{w=4393 s=6076 e=4393 n=4393 c=B}
{w=4419 s=5206 e=2671 n=4168 c=N}
{w=4433 s=4358 e=4583 n=4358 c=N}
{w=4433 s=4433 e=4358 n=4358 c=B}
{w=4433 s=4583 e=4358 n=4689 c=N}
{w=4433 s=5741 e=2671 n=4583 c=B}
{w=4545 s=4708 e=5708 n=4552 c=N}
{w=4545 s=5700 e=4708 n=4708 c=B}
{w=4551 s=4173 e=4551 n=2887 c=B}
{w=4551 s=4551 e=3525 n=3525 c=N}
{w=4551 s=5206 e=3525 n=4173 c=N}
{w=4552 s=5188 e=5206 n=5188 c=N}
{w=4552 s=5206 e=5206 n=4552 c=N}
{w=4576 s=5480 e=5480 n=5206 c=B}
{w=4583 s=2285 e=2285 n=2285 c=N}
{w=4583 s=2671 e=4583 n=4583 c=N}
{w=4583 s=2915 e=2915 n=2915 c=N}
{w=4583 s=2937 e=5741 n=4083 c=N}
{w=4583 s=3411 e=4583 n=2915 c=B}
{w=4583 s=3418 e=4689 n=4583 c=B}
{w=4583 s=3602 e=2937 n=2285 c=N}
{w=4583 s=3953 e=4723 n=4583 c=B}
{w=4583 s=4083 e=4083 n=3602 c=N}
{w=4583 s=4358 e=4583 n=3411 c=B}
{w=4583 s=4433 e=4583 n=4433 c=B}
{w=4583 s=4583 e=4583 n=4583 c=B}
{w=4583 s=4689 e=4583 n=2671 c=N}
{w=4583 s=4723 e=4723 n=5741 c=B}
{w=4583 s=4729 e=3953 n=4583 c=B}
{w=4583 s=4866 e=4866 n=4729 c=B}
{w=4583 s=5480 e=4583 n=4583 c=B}
{w=4583 s=5741 e=2937 n=4583 c=B}
{w=4583 s=5994 e=5994 n=4583 c=B}
{w=4610 s=3602 e=3602 n=3602 c=B}
{w=4610 s=4610 e=4610 n=4610 c=B}
{w=4610 s=4723 e=3602 n=3602 c=N}
{w=4610 s=4741 e=4741 n=4741 c=N}
{w=4610 s=4976 e=4976 n=4723 c=B}
{w=4610 s=5133 e=3602 n=4976 c=N}
{w=4610 s=5393 e=4610 n=4610 c=N}
{w=4610 s=5772 e=4741 n=6017 c=N}
{w=4610 s=6017 e=6017 n=4741 c=N}
{w=4610 s=6027 e=4610 n=4610 c=N}
{w=4610 s=6150 e=5772 n=4610 c=B}
{w=4610 s=6923 e=4741 n=4610 c=N}
{w=4689 s=4583 e=4583 n=4583 c=N}
{w=4689 s=5741 e=4358 n=2671 c=B}
{w=4708 s=4708 e=4708 n=4708 c=B}
{w=4708 s=5069 e=7252 n=7252 c=B}
{w=4708 s=5188 e=4552 n=5188 c=N}
{w=4708 s=5206 e=5206 n=6366 c=N}
{w=4708 s=5209 e=5480 n=5480 c=N}
{w=4708 s=5314 e=5314 n=5206 c=N}
{w=4708 s=5480 e=5209 n=5314 c=N}
{w=4708 s=5700 e=4708 n=4708 c=N}
{w=4708 s=5962 e=5962 n=4708 c=B}
{w=4708 s=5981 e=6366 n=5700 c=N}
{w=4708 s=5987 e=5987 n=4708 c=B}

{w=4708 s=6087 e=5962 n=4708 c=N}

{w=4708 s=6366 e=6122 n=5209 c=N}

{w=4708 s=7252 e=4708 n=4708 c=B}

{w=4723 s=4083 e=4083 n=4083 c=N}

{w=4723 s=4610 e=4610 n=4610 c=N}

{w=4723 s=4723 e=4723 n=4723 c=B}

{w=4723 s=4866 e=4723 n=4723 c=B}

{w=4723 s=4999 e=5133 n=5133 c=N}

{w=4723 s=5109 e=5109 n=2693 c=B}

{w=4723 s=5114 e=5133 n=4999 c=N}

{w=4723 s=5133 e=4610 n=4610 c=N}

{w=4741 s=3498 e=3498 n=5790 c=B}

{w=4741 s=4610 e=4741 n=4741 c=N}

{w=4741 s=4741 e=4741 n=4741 c=B}

{w=4741 s=5401 e=5401 n=4741 c=N}

{w=4741 s=5649 e=5649 n=5649 c=B}

{w=4741 s=5772 e=4610 n=6150 c=B}

{w=4741 s=5790 e=5790 n=4741 c=B}

{w=4741 s=6017 e=4741 n=4610 c=B}

{w=4741 s=6027 e=6027 n=6440 c=N}

{w=4741 s=6424 e=3498 n=3498 c=N}

{w=4741 s=6440 e=3498 n=6424 c=N}

{w=4741 s=6923 e=6027 n=4741 c=N}

{w=4776 s=4393 e=4032 n=4776 c=B}

{w=4776 s=4776 e=4032 n=4776 c=B}

{w=4797 s=4797 e=4797 n=4797 c=B}

{w=4797 s=5054 e=4797 n=4797 c=B}

{w=4866 s=4610 e=5771 n=6526 c=N}

{w=4866 s=4723 e=4723 n=4729 c=B}

{w=4866 s=4729 e=4723 n=4723 c=B}

{w=4866 s=4866 e=4866 n=4866 c=B}

{w=4866 s=4976 e=4866 n=4723 c=B}

{w=4866 s=5111 e=5755 n=4866 c=B}

{w=4866 s=5114 e=5114 n=5114 c=N}

{w=4866 s=5755 e=4866 n=4976 c=B}

{w=4866 s=5771 e=6027 n=4610 c=N}

{w=4866 s=5988 e=5111 n=4866 c=B}

{w=4866 s=6027 e=5771 n=5114 c=N}

{w=4866 s=6152 e=6526 n=4866 c=B}

{w=4866 s=6526 e=4610 n=6027 c=N}

{w=4933 s=4933 e=4933 n=4933 c=B}

{w=4933 s=5700 e=4933 n=5700 c=B}

{w=4933 s=7745 e=5962 n=5962 c=N}

{w=4933 s=7884 e=7745 n=5700 c=N}

{w=4933 s=7888 e=7884 n=5700 c=N}

{w=4976 s=3602 e=3602 n=4999 c=B}

{w=4976 s=4610 e=4976 n=3602 c=N}

{w=4976 s=4741 e=5411 n=4976 c=B}

{w=4976 s=4976 e=4976 n=3602 c=B}

{w=4976 s=5133 e=3602 n=4976 c=N}

{w=4999 s=3602 e=3602 n=4109 c=B}

{w=4999 s=4109 e=4372 n=3602 c=B}

{w=5054 s=5054 e=5054 n=5054 c=B}

{w=5054 s=5306 e=5054 n=5306 c=B}

{w=5054 s=6076 e=5054 n=5054 c=B}

{w=5069 s=4708 e=4708 n=5700 c=B}

{w=5069 s=6087 e=6087 n=7252 c=N}

{w=5109 s=2693 e=2693 n=2693 c=N}

{w=5109 s=3602 e=3602 n=4610 c=B}

{w=5109 s=4723 e=2693 n=5109 c=N}

{w=5111 s=4866 e=4866 n=5111 c=B}

{w=5111 s=5988 e=5988 n=4866 c=B}

{w=5114 s=4610 e=4741 n=5133 c=N}

{w=5114 s=4866 e=5133 n=5114 c=B}

{w=5114 s=5114 e=5133 n=4866 c=N}

{w=5114 s=5771 e=4610 n=5114 c=B}

{w=5133 s=3602 e=3602 n=3602 c=N}
{w=5133 s=4610 e=5114 n=5114 c=N}
{w=5133 s=4741 e=4976 n=3602 c=N}
{w=5133 s=5114 e=5133 n=5133 c=N}
{w=5133 s=5133 e=5133 n=5133 c=B}
{w=5188 s=3525 e=4173 n=3013 c=N}
{w=5188 s=3915 e=3915 n=3013 c=N}
{w=5188 s=4545 e=5188 n=5188 c=N}
{w=5188 s=4552 e=5206 n=5188 c=B}
{w=5188 s=4708 e=4545 n=5206 c=N}
{w=5188 s=5188 e=5188 n=5188 c=N}
{w=5188 s=5206 e=5569 n=4173 c=N}
{w=5188 s=5569 e=3525 n=5569 c=N}
{w=5188 s=6210 e=2887 n=4173 c=N}
{w=5206 s=3938 e=3938 n=5206 c=B}
{w=5206 s=4552 e=6356 n=5206 c=N}
{w=5206 s=4576 e=4576 n=3938 c=B}
{w=5206 s=4583 e=4433 n=3411 c=N}
{w=5206 s=4689 e=4583 n=4689 c=B}
{w=5206 s=4708 e=5708 n=5569 c=N}
{w=5206 s=5206 e=5206 n=5206 c=B}
{w=5206 s=5209 e=5209 n=5209 c=N}
{w=5206 s=5314 e=5480 n=5206 c=B}
{w=5206 s=5480 e=5480 n=4576 c=B}
{w=5206 s=5569 e=5206 n=5206 c=N}
{w=5206 s=5700 e=5206 n=5700 c=B}
{w=5206 s=5981 e=5987 n=5206 c=N}
{w=5206 s=5987 e=5981 n=5209 c=N}
{w=5206 s=6122 e=5480 n=5314 c=B}
{w=5206 s=6356 e=5569 n=5569 c=N}
{w=5206 s=6366 e=5206 n=5206 c=N}
{w=5206 s=6886 e=7144 n=5988 c=N}
{w=5209 s=4708 e=5994 n=4708 c=B}
{w=5209 s=5206 e=5209 n=5209 c=N}
{w=5209 s=5209 e=5209 n=5209 c=B}
{w=5209 s=5314 e=5209 n=6122 c=B}
{w=5209 s=5981 e=5209 n=5209 c=N}
{w=5209 s=5987 e=4708 n=5209 c=B}
{w=5209 s=5994 e=5994 n=5480 c=B}
{w=5209 s=6258 e=6258 n=5209 c=B}
{w=5209 s=6366 e=5209 n=5209 c=N}
{w=5306 s=5306 e=5054 n=6076 c=B}
{w=5306 s=6076 e=5054 n=5306 c=B}
{w=5314 s=5209 e=6122 n=5206 c=B}
{w=5393 s=6027 e=4610 n=5393 c=N}
{w=5401 s=3498 e=3498 n=4741 c=B}
{w=5401 s=4741 e=5401 n=5401 c=N}
{w=5401 s=5401 e=3498 n=4741 c=B}
{w=5411 s=4741 e=5771 n=3602 c=N}
{w=5426 s=6076 e=4393 n=5426 c=B}
{w=5480 s=4583 e=4583 n=4583 c=B}
{w=5480 s=4708 e=5209 n=5209 c=B}
{w=5480 s=5209 e=5480 n=5480 c=B}
{w=5480 s=5480 e=5480 n=5480 c=B}
{w=5480 s=5994 e=4583 n=5480 c=B}
{w=5480 s=6122 e=6366 n=6366 c=B}
{w=5569 s=3525 e=3525 n=3525 c=N}
{w=5569 s=4173 e=5206 n=4173 c=N}
{w=5569 s=5188 e=5188 n=3525 c=N}
{w=5569 s=5206 e=5206 n=5206 c=N}
{w=5569 s=5569 e=5569 n=5569 c=N}
{w=5569 s=5962 e=7252 n=6087 c=N}
{w=5569 s=6210 e=4173 n=5188 c=B}
{w=5649 s=3498 e=3498 n=3498 c=B}
{w=5649 s=4255 e=5649 n=4255 c=B}
{w=5649 s=4741 e=4372 n=4255 c=B}

{w=5679 s=6076 e=6076 n=5679 c=B}
{w=5700 s=4708 e=4708 n=4708 c=N}
{w=5700 s=4933 e=5962 n=5962 c=N}
{w=5700 s=5069 e=5069 n=5700 c=B}
{w=5700 s=5700 e=5700 n=5700 c=N}
{w=5700 s=5962 e=5569 n=7252 c=N}
{w=5700 s=5981 e=5206 n=6366 c=N}
{w=5700 s=6087 e=4708 n=5069 c=N}
{w=5700 s=7252 e=5069 n=6087 c=B}
{w=5700 s=7389 e=5700 n=5700 c=B}
{w=5700 s=7745 e=7745 n=4933 c=B}
{w=5700 s=7884 e=5700 n=5700 c=N}
{w=5708 s=4708 e=5206 n=5206 c=N}
{w=5708 s=5569 e=5206 n=6356 c=N}
{w=5729 s=5988 e=5988 n=5994 c=B}
{w=5741 s=4083 e=4083 n=4083 c=N}
{w=5741 s=4583 e=5741 n=5741 c=N}
{w=5741 s=4723 e=5109 n=5109 c=N}
{w=5741 s=5741 e=3953 n=4583 c=N}
{w=5755 s=4866 e=4866 n=5755 c=B}
{w=5771 s=4610 e=4610 n=5771 c=N}
{w=5771 s=4741 e=4741 n=3602 c=B}
{w=5771 s=6027 e=5393 n=4610 c=N}
{w=5772 s=4610 e=5772 n=5772 c=N}
{w=5772 s=4741 e=4741 n=4741 c=N}
{w=5772 s=5772 e=6793 n=6027 c=B}
{w=5772 s=6150 e=5772 n=5772 c=N}
{w=5772 s=6923 e=4741 n=6923 c=N}
{w=5790 s=3498 e=3498 n=5649 c=B}
{w=5880 s=5988 e=5729 n=5994 c=B}
{w=5962 s=5206 e=6366 n=5206 c=N}
{w=5962 s=5700 e=6340 n=5962 c=B}
{w=5962 s=5962 e=5962 n=5962 c=B}
{w=5962 s=5981 e=5981 n=5981 c=B}
{w=5962 s=6087 e=6366 n=5962 c=B}
{w=5962 s=7389 e=6729 n=5981 c=B}
{w=5962 s=7884 e=4933 n=5700 c=N}
{w=5981 s=5209 e=6258 n=5209 c=B}
{w=5981 s=5962 e=6087 n=4708 c=N}
{w=5981 s=5981 e=5981 n=5981 c=B}
{w=5981 s=5987 e=5987 n=5981 c=B}
{w=5981 s=6258 e=5209 n=5981 c=B}
{w=5981 s=7389 e=6258 n=5981 c=B}
{w=5987 s=5209 e=5987 n=5209 c=N}
{w=5987 s=5729 e=5987 n=5987 c=B}
{w=5987 s=5981 e=5987 n=5987 c=N}
{w=5987 s=5987 e=5987 n=5987 c=B}
{w=5987 s=5988 e=5988 n=5729 c=B}
{w=5987 s=6258 e=5987 n=5987 c=N}
{w=5988 s=4866 e=4866 n=4866 c=B}
{w=5988 s=5988 e=5988 n=5988 c=B}
{w=5988 s=5998 e=6773 n=6792 c=B}
{w=5988 s=6152 e=6152 n=4866 c=B}
{w=5988 s=6258 e=5988 n=5987 c=B}
{w=5988 s=6281 e=6152 n=6890 c=B}
{w=5988 s=6773 e=6773 n=6281 c=B}
{w=5988 s=6792 e=6773 n=6892 c=B}
{w=5988 s=6886 e=5988 n=5988 c=N}
{w=5988 s=6890 e=6281 n=5988 c=B}
{w=5988 s=6892 e=6773 n=5988 c=B}
{w=5988 s=7144 e=5988 n=5988 c=N}
{w=5988 s=7527 e=6773 n=5988 c=N}
{w=5988 s=7653 e=6773 n=5998 c=B}
{w=5994 s=4866 e=4866 n=4866 c=B}
{w=5994 s=5729 e=5880 n=5987 c=B}
{w=5994 s=5987 e=5994 n=5994 c=B}

{w=5994 s=5988 e=5988 n=4866 c=B}
{w=5994 s=5994 e=5994 n=5994 c=B}
{w=5998 s=6773 e=6773 n=7653 c=B}
{w=6017 s=4741 e=5401 n=6017 c=N}
{w=6027 s=3498 e=6440 n=6424 c=B}
{w=6027 s=4610 e=4610 n=4610 c=N}
{w=6027 s=5772 e=5772 n=4610 c=N}
{w=6027 s=6027 e=6027 n=6027 c=B}
{w=6027 s=6150 e=6526 n=6027 c=N}
{w=6027 s=6295 e=3498 n=6027 c=N}
{w=6027 s=6923 e=4741 n=4610 c=N}
{w=6027 s=7302 e=6027 n=6027 c=N}
{w=6076 s=5054 e=5679 n=6076 c=B}
{w=6076 s=5306 e=5306 n=5306 c=B}
{w=6076 s=5679 e=6076 n=5306 c=B}
{w=6076 s=6076 e=6076 n=6076 c=B}
{w=6076 s=6316 e=6076 n=6440 c=B}
{w=6076 s=6440 e=6076 n=6076 c=B}
{w=6076 s=6821 e=6076 n=5054 c=B}
{w=6076 s=7602 e=7724 n=6076 c=N}
{w=6076 s=7610 e=6076 n=6076 c=B}
{w=6076 s=7872 e=7594 n=7594 c=N}
{w=6087 s=5962 e=5981 n=5700 c=N}
{w=6087 s=5981 e=6232 n=6087 c=N}
{w=6087 s=7252 e=4708 n=5069 c=B}
{w=6122 s=5206 e=5209 n=6122 c=N}
{w=6122 s=5209 e=5480 n=5480 c=B}
{w=6122 s=6122 e=6122 n=6122 c=B}
{w=6150 s=6027 e=6150 n=6150 c=N}
{w=6150 s=6150 e=6150 n=6150 c=B}
{w=6150 s=6922 e=6150 n=6150 c=N}
{w=6150 s=6923 e=6923 n=6923 c=B}
{w=6150 s=7317 e=6923 n=6923 c=B}
{w=6150 s=7693 e=7173 n=6922 c=N}
{w=6152 s=6027 e=6027 n=6027 c=N}
{w=6152 s=6150 e=6922 n=7173 c=N}
{w=6152 s=6152 e=6152 n=6152 c=B}
{w=6152 s=6773 e=6152 n=6152 c=B}
{w=6152 s=6792 e=6152 n=6152 c=B}
{w=6152 s=6922 e=6150 n=6925 c=N}
{w=6152 s=6925 e=6150 n=6027 c=N}
{w=6152 s=7173 e=6150 n=7302 c=N}
{w=6152 s=7302 e=7173 n=6922 c=N}
{w=6179 s=6076 e=6076 n=6179 c=B}
{w=6179 s=6440 e=6076 n=6440 c=B}
{w=6210 s=4173 e=5206 n=6210 c=B}
{w=6210 s=5188 e=4173 n=6210 c=N}
{w=6210 s=5206 e=5206 n=6210 c=B}
{w=6210 s=5569 e=6210 n=6210 c=N}
{w=6232 s=5981 e=4708 n=6087 c=N}
{w=6258 s=5206 e=5988 n=7292 c=N}
{w=6258 s=5729 e=5988 n=5987 c=N}
{w=6258 s=5987 e=5206 n=5206 c=N}
{w=6258 s=5988 e=5988 n=6258 c=N}
{w=6258 s=6258 e=6258 n=6258 c=B}
{w=6258 s=6886 e=6886 n=6892 c=N}
{w=6258 s=6892 e=6886 n=5987 c=N}
{w=6258 s=7273 e=7273 n=6258 c=B}
{w=6258 s=7292 e=6258 n=5729 c=N}
{w=6258 s=7389 e=6258 n=7389 c=B}
{w=6258 s=7527 e=7652 n=7652 c=N}
{w=6258 s=7652 e=7652 n=6886 c=N}
{w=6281 s=6152 e=6152 n=5988 c=B}
{w=6281 s=6281 e=6281 n=6281 c=B}
{w=6281 s=7317 e=7320 n=6295 c=B}
{w=6281 s=7444 e=6281 n=6281 c=N}

{w=6281 s=7456 e=7456 n=7456 c=N}

{w=6281 s=7694 e=7456 n=7456 c=N}

{w=6281 s=7844 e=7220 n=7694 c=N}

{w=6295 s=3498 e=6440 n=6437 c=N}

{w=6295 s=6295 e=6295 n=6295 c=B}

{w=6295 s=6437 e=6437 n=3498 c=N}

{w=6295 s=6440 e=6437 n=6437 c=N}

{w=6295 s=7220 e=7444 n=3498 c=N}

{w=6295 s=7317 e=7317 n=6295 c=B}

{w=6295 s=7444 e=6440 n=7220 c=N}

{w=6340 s=5700 e=5962 n=5962 c=N}

{w=6340 s=5962 e=5962 n=5962 c=N}

{w=6340 s=7389 e=8132 n=5981 c=B}

{w=6356 s=4545 e=5700 n=6356 c=N}

{w=6356 s=5206 e=5708 n=5206 c=B}

{w=6356 s=5700 e=4545 n=4545 c=N}

{w=6356 s=6356 e=4708 n=4545 c=N}

{w=6366 s=5209 e=5209 n=5314 c=N}

{w=6366 s=5700 e=5206 n=5206 c=N}

{w=6366 s=5981 e=5700 n=5700 c=N}

{w=6437 s=6440 e=6440 n=6440 c=N}

{w=6437 s=7872 e=6440 n=6440 c=N}

{w=6440 s=3498 e=3498 n=3498 c=N}

{w=6440 s=6076 e=6076 n=6821 c=B}

{w=6440 s=6316 e=6440 n=6316 c=B}

{w=6440 s=6437 e=6440 n=6440 c=N}

{w=6440 s=6440 e=6440 n=6440 c=B}

{w=6440 s=6821 e=6076 n=6440 c=B}

{w=6440 s=7317 e=6281 n=6295 c=B}

{w=6440 s=7594 e=7610 n=7714 c=N}

{w=6440 s=7610 e=6076 n=6316 c=B}

{w=6440 s=7714 e=7610 n=6437 c=N}

{w=6440 s=7872 e=7714 n=6437 c=N}

{w=6526 s=6027 e=6152 n=5771 c=N}

{w=6526 s=6150 e=7302 n=7302 c=N}

{w=6526 s=6152 e=6526 n=4866 c=B}

{w=6548 s=7444 e=7872 n=7872 c=N}

{w=6548 s=7582 e=7973 n=7872 c=N}

{w=6548 s=7602 e=7582 n=7582 c=N}

{w=6729 s=7389 e=6340 n=5981 c=B}

{w=6773 s=5988 e=5998 n=5988 c=B}

{w=6773 s=6152 e=6152 n=6152 c=B}

{w=6773 s=6773 e=6773 n=6773 c=B}

{w=6773 s=6792 e=6152 n=6152 c=B}

{w=6773 s=7170 e=7170 n=7653 c=B}

{w=6773 s=7527 e=5988 n=7527 c=N}

{w=6773 s=7653 e=6792 n=6773 c=B}

{w=6773 s=8049 e=7170 n=7527 c=N}

{w=6792 s=6792 e=6792 n=6792 c=B}

{w=6792 s=6922 e=6922 n=7173 c=N}

{w=6792 s=7170 e=6792 n=6792 c=B}

{w=6792 s=7173 e=7173 n=6150 c=N}

{w=6792 s=7302 e=6925 n=6922 c=N}

{w=6792 s=7653 e=6792 n=7653 c=B}

{w=6792 s=7693 e=7302 n=6792 c=N}

{w=6793 s=5772 e=4741 n=4610 c=N}

{w=6886 s=5988 e=5988 n=5988 c=N}

{w=6886 s=6258 e=6886 n=6886 c=N}

{w=6886 s=6773 e=5988 n=7527 c=N}

{w=6886 s=6886 e=6886 n=6886 c=B}

{w=6886 s=7527 e=6886 n=5988 c=N}

{w=6886 s=7652 e=6886 n=6886 c=N}

{w=6892 s=7527 e=5988 n=6773 c=N}

{w=6922 s=6150 e=6150 n=6150 c=N}

{w=6922 s=6792 e=6922 n=6922 c=N}

{w=6922 s=6922 e=6922 n=6922 c=B}

{w=6922 s=7419 e=7693 n=6150 c=N}

{w=6922 s=7693 e=6150 n=7419 c=N}

{w=6923 s=6150 e=6295 n=6295 c=B}

{w=6923 s=6295 e=6295 n=6923 c=B}

{w=6923 s=6440 e=6295 n=7320 c=B}

{w=6923 s=6923 e=6923 n=6923 c=B}

{w=6923 s=7317 e=6440 n=6440 c=B}

{w=6923 s=7320 e=6295 n=6150 c=B}

{w=6923 s=7693 e=7419 n=7693 c=N}

{w=6925 s=7532 e=7693 n=7532 c=N}

{w=6925 s=7693 e=6792 n=6922 c=N}

{w=7004 s=5700 e=5962 n=5962 c=N}

{w=7004 s=5962 e=6340 n=6340 c=N}

{w=7004 s=6340 e=5962 n=5700 c=N}

{w=7112 s=7112 e=7888 n=7112 c=N}

{w=7112 s=7884 e=7888 n=7888 c=N}

{w=7114 s=7114 e=7114 n=7114 c=B}

{w=7114 s=7273 e=7273 n=7654 c=N}

{w=7114 s=7654 e=7273 n=7273 c=N}

{w=7144 s=6886 e=5988 n=7144 c=N}

{w=7144 s=7668 e=8022 n=8049 c=N}

{w=7170 s=7170 e=7170 n=7170 c=B}

{w=7170 s=7302 e=7532 n=7532 c=N}

{w=7170 s=7419 e=7302 n=7532 c=N}

{w=7170 s=7527 e=7535 n=7527 c=N}

{w=7170 s=7532 e=7693 n=7302 c=N}

{w=7170 s=7654 e=7668 n=7654 c=N}

{w=7170 s=7668 e=8049 n=8049 c=B}

{w=7170 s=7693 e=7532 n=7419 c=N}

{w=7170 s=7809 e=7419 n=7693 c=N}

{w=7170 s=8049 e=7170 n=7170 c=N}

{w=7170 s=8155 e=7532 n=7809 c=N}

{w=7173 s=6150 e=6922 n=6027 c=B}

{w=7173 s=6922 e=6922 n=6922 c=N}

{w=7173 s=7693 e=7317 n=6150 c=N}

{w=7181 s=7317 e=6295 n=6295 c=B}

{w=7220 s=6281 e=6281 n=6281 c=N}

{w=7220 s=7456 e=7456 n=7317 c=B}

{w=7220 s=7844 e=7456 n=7844 c=B}

{w=7220 s=7872 e=7456 n=6440 c=N}

{w=7252 s=4708 e=5700 n=5700 c=N}

{w=7252 s=5962 e=6087 n=7252 c=N}

{w=7265 s=7114 e=7273 n=7389 c=N}

{w=7265 s=7273 e=7654 n=7265 c=N}

{w=7265 s=8039 e=8022 n=7273 c=N}

{w=7273 s=7114 e=7273 n=7273 c=N}

{w=7273 s=7265 e=7527 n=8039 c=N}

{w=7273 s=7273 e=7273 n=7273 c=B}

{w=7273 s=7527 e=7527 n=7535 c=N}

{w=7273 s=7535 e=7535 n=7652 c=N}

{w=7273 s=7652 e=7652 n=7652 c=N}

{w=7273 s=7654 e=7114 n=7114 c=B}

{w=7273 s=8039 e=7273 n=7527 c=N}

{w=7302 s=6150 e=5772 n=6027 c=N}

{w=7302 s=7532 e=6925 n=7693 c=N}

{w=7302 s=7693 e=6923 n=6922 c=N}

{w=7317 s=6281 e=7456 n=7693 c=N}

{w=7317 s=6437 e=6437 n=7444 c=N}

{w=7317 s=6440 e=7872 n=7872 c=N}

{w=7317 s=7317 e=7317 n=7317 c=B}

{w=7317 s=7456 e=7220 n=7317 c=B}

{w=7317 s=7582 e=7317 n=7693 c=B}

{w=7317 s=7693 e=7317 n=7317 c=N}

{w=7317 s=7872 e=6437 n=8177 c=N}

{w=7317 s=8177 e=6440 n=6437 c=N}

{w=7320 s=6076 e=7603 n=7610 c=B}

{w=7320 s=7317 e=7181 n=6295 c=B}

{w=7320 s=7467 e=7320 n=7320 c=B}

{w=7320 s=7602 e=7602 n=7467 c=B}

{w=7320 s=7603 e=7320 n=7603 c=B}

{w=7389 s=5700 e=7745 n=7745 c=B}

{w=7389 s=6258 e=6258 n=6258 c=B}

{w=7389 s=7114 e=7265 n=7632 c=N}

{w=7389 s=7265 e=7389 n=7389 c=N}

{w=7389 s=7273 e=7273 n=6258 c=B}

{w=7389 s=7389 e=7389 n=7389 c=B}

{w=7389 s=7632 e=7632 n=7265 c=B}

{w=7389 s=7884 e=7389 n=7389 c=N}

{w=7419 s=7532 e=7532 n=7532 c=N}

{w=7419 s=7693 e=6150 n=6150 c=N}

{w=7444 s=6281 e=7844 n=6281 c=N}

{w=7444 s=6440 e=6295 n=6440 c=N}

{w=7444 s=7444 e=7444 n=7444 c=B}

{w=7444 s=7582 e=7872 n=7444 c=N}

{w=7444 s=7693 e=7844 n=7844 c=N}

{w=7444 s=7844 e=7444 n=7444 c=N}

{w=7444 s=7872 e=7872 n=7872 c=N}

{w=7456 s=6281 e=7220 n=7317 c=N}

{w=7456 s=7444 e=7582 n=7872 c=N}

{w=7456 s=7456 e=7456 n=7456 c=B}

{w=7456 s=7582 e=7444 n=7444 c=N}

{w=7456 s=7844 e=7456 n=7456 c=N}

{w=7456 s=7872 e=7872 n=7872 c=N}

{w=7467 s=7602 e=7602 n=7872 c=B}

{w=7491 s=7874 e=8132 n=7874 c=N}

{w=7505 s=7505 e=7617 n=7888 c=N}

{w=7505 s=7874 e=7884 n=7884 c=B}

{w=7527 s=6773 e=7653 n=6773 c=N}

{w=7527 s=7144 e=8049 n=7665 c=N}

{w=7527 s=7170 e=7654 n=7654 c=N}

{w=7527 s=7273 e=7527 n=7527 c=N}

{w=7527 s=7527 e=7527 n=7527 c=B}

{w=7527 s=7535 e=7527 n=7535 c=B}

{w=7527 s=7654 e=7527 n=7527 c=N}

{w=7527 s=7665 e=6773 n=6773 c=N}

{w=7527 s=7668 e=7665 n=8049 c=N}

{w=7527 s=8039 e=7527 n=7170 c=N}

{w=7527 s=8049 e=7144 n=7144 c=N}

{w=7532 s=7419 e=7532 n=7532 c=N}

{w=7532 s=7532 e=7532 n=7532 c=B}

{w=7532 s=7693 e=7693 n=7693 c=N}

{w=7532 s=8155 e=7693 n=7693 c=N}

{w=7535 s=7527 e=7527 n=7527 c=N}

{w=7535 s=7535 e=7170 n=7535 c=N}

{w=7582 s=6281 e=7317 n=7582 c=N}

{w=7582 s=6548 e=7582 n=7444 c=N}

{w=7582 s=7444 e=7444 n=7456 c=N}

{w=7582 s=7582 e=7582 n=7582 c=N}

{w=7582 s=7872 e=7602 n=7872 c=N}

{w=7582 s=7973 e=6548 n=6548 c=N}

{w=7594 s=7872 e=7610 n=7594 c=N}

{w=7602 s=6076 e=7724 n=6076 c=N}

{w=7602 s=7320 e=7602 n=7724 c=N}

{w=7602 s=7582 e=7872 n=7872 c=N}

{w=7602 s=7602 e=7602 n=7602 c=N}

{w=7602 s=7724 e=6076 n=7872 c=B}

{w=7602 s=7872 e=7602 n=6076 c=B}

{w=7603 s=6076 e=6076 n=7610 c=N}

{w=7603 s=7320 e=7872 n=7603 c=N}

{w=7603 s=7603 e=7603 n=7603 c=N}

{w=7603 s=7610 e=7610 n=7610 c=B}

{w=7603 s=7872 e=7320 n=7603 c=N}

{w=7610 s=6076 e=6076 n=6076 c=B}
{w=7610 s=7594 e=7872 n=7872 c=B}
{w=7610 s=7603 e=7872 n=7872 c=B}
{w=7610 s=7610 e=7610 n=7610 c=B}
{w=7610 s=7872 e=7610 n=7610 c=N}
{w=7617 s=7114 e=7114 n=7114 c=N}
{w=7617 s=7617 e=7654 n=7884 c=N}
{w=7617 s=7654 e=7617 n=7114 c=B}
{w=7617 s=7874 e=7505 n=7884 c=N}
{w=7632 s=5700 e=7389 n=7389 c=N}
{w=7632 s=5962 e=7632 n=7888 c=N}
{w=7632 s=7273 e=7884 n=7389 c=B}
{w=7632 s=7389 e=7632 n=7389 c=N}
{w=7632 s=7884 e=5962 n=4933 c=N}
{w=7652 s=6258 e=6258 n=6892 c=N}
{w=7652 s=6892 e=6886 n=6258 c=B}
{w=7652 s=7273 e=7527 n=7652 c=N}
{w=7652 s=7527 e=6892 n=6886 c=N}
{w=7652 s=7535 e=7535 n=7273 c=B}
{w=7652 s=7652 e=7527 n=6258 c=B}
{w=7653 s=6773 e=6773 n=7653 c=N}
{w=7653 s=7170 e=7170 n=7653 c=B}
{w=7653 s=7527 e=8049 n=6773 c=B}
{w=7653 s=7653 e=7653 n=7653 c=B}
{w=7654 s=7114 e=7654 n=7114 c=N}
{w=7654 s=7265 e=7265 n=7527 c=N}
{w=7654 s=7273 e=7265 n=7273 c=N}
{w=7654 s=7527 e=7527 n=7527 c=N}
{w=7654 s=7654 e=7654 n=7654 c=N}
{w=7665 s=7668 e=7668 n=7668 c=B}
{w=7668 s=7170 e=7668 n=8049 c=N}
{w=7668 s=7527 e=7668 n=7527 c=N}
{w=7668 s=7668 e=7668 n=7668 c=N}
{w=7668 s=8049 e=8049 n=8049 c=B}
{w=7668 s=8155 e=8155 n=7170 c=N}
{w=7693 s=6150 e=6150 n=6150 c=N}
{w=7693 s=6281 e=6281 n=7220 c=N}
{w=7693 s=7220 e=7582 n=7317 c=N}
{w=7693 s=7317 e=7317 n=7317 c=N}
{w=7693 s=7444 e=7220 n=6281 c=N}
{w=7693 s=7532 e=7693 n=7693 c=N}
{w=7693 s=7693 e=7693 n=7693 c=B}
{w=7693 s=7844 e=6281 n=7444 c=N}
{w=7693 s=8155 e=7693 n=7532 c=B}
{w=7714 s=7872 e=6076 n=6440 c=N}
{w=7724 s=6076 e=6076 n=6076 c=N}
{w=7724 s=7602 e=6076 n=6076 c=B}
{w=7745 s=5962 e=7389 n=5962 c=N}
{w=7745 s=7389 e=7884 n=5962 c=B}
{w=7745 s=7745 e=7745 n=7745 c=N}
{w=7745 s=7884 e=7389 n=7745 c=N}
{w=7809 s=8155 e=8155 n=7419 c=N}
{w=7844 s=6281 e=7444 n=6281 c=N}
{w=7844 s=7444 e=7582 n=7456 c=N}
{w=7844 s=7456 e=7844 n=7844 c=N}
{w=7844 s=7844 e=7844 n=7844 c=N}
{w=7844 s=8155 e=7444 n=7693 c=N}
{w=7872 s=6548 e=7872 n=7872 c=N}
{w=7872 s=7320 e=7320 n=7603 c=N}
{w=7872 s=7444 e=6548 n=6548 c=N}
{w=7872 s=7456 e=7444 n=6548 c=B}
{w=7872 s=7467 e=7603 n=7320 c=N}
{w=7872 s=7582 e=6548 n=7444 c=B}
{w=7872 s=7602 e=7467 n=7467 c=N}
{w=7872 s=7603 e=7603 n=7610 c=N}
{w=7872 s=7610 e=7610 n=7610 c=N}

{w=7872 s=7872 e=7872 n=7872 c=B}

{w=7872 s=7973 e=7872 n=7872 c=N}

{w=7874 s=7505 e=7505 n=7884 c=N}

{w=7874 s=7617 e=7884 n=7884 c=N}

{w=7874 s=7874 e=7874 n=7874 c=N}

{w=7874 s=7884 e=7884 n=7888 c=B}

{w=7874 s=7888 e=7112 n=7884 c=N}

{w=7884 s=4933 e=7632 n=7884 c=B}

{w=7884 s=5700 e=4933 n=5700 c=N}

{w=7884 s=7112 e=7884 n=5962 c=N}

{w=7884 s=7114 e=7617 n=7617 c=N}

{w=7884 s=7273 e=7273 n=7389 c=B}

{w=7884 s=7389 e=7389 n=7389 c=N}

{w=7884 s=7505 e=7888 n=7888 c=N}

{w=7884 s=7617 e=7114 n=7389 c=N}

{w=7884 s=7654 e=7617 n=7114 c=N}

{w=7884 s=7874 e=7884 n=7505 c=N}

{w=7884 s=7884 e=7884 n=7884 c=B}

{w=7884 s=7888 e=7884 n=7884 c=N}

{w=7888 s=7874 e=7884 n=7112 c=N}

{w=7888 s=7884 e=7884 n=7884 c=N}

{w=7973 s=6548 e=7602 n=7973 c=B}

{w=8022 s=7527 e=8022 n=7527 c=N}

{w=8022 s=7668 e=8049 n=8049 c=B}

{w=8022 s=8022 e=8022 n=8039 c=N}

{w=8022 s=8039 e=7170 n=7527 c=N}

{w=8049 s=7170 e=7170 n=7170 c=N}

{w=8049 s=7668 e=7170 n=7170 c=N}

{w=8049 s=8049 e=8049 n=8049 c=B}

{w=8132 s=7389 e=5981 n=5981 c=B}

{w=8132 s=7874 e=7617 n=7617 c=B}

{w=8155 s=7532 e=8155 n=7532 c=N}

{w=8155 s=7582 e=7844 n=7532 c=N}

{w=8155 s=7809 e=7809 n=7532 c=N}

{w=8155 s=8155 e=8155 n=8155 c=N}