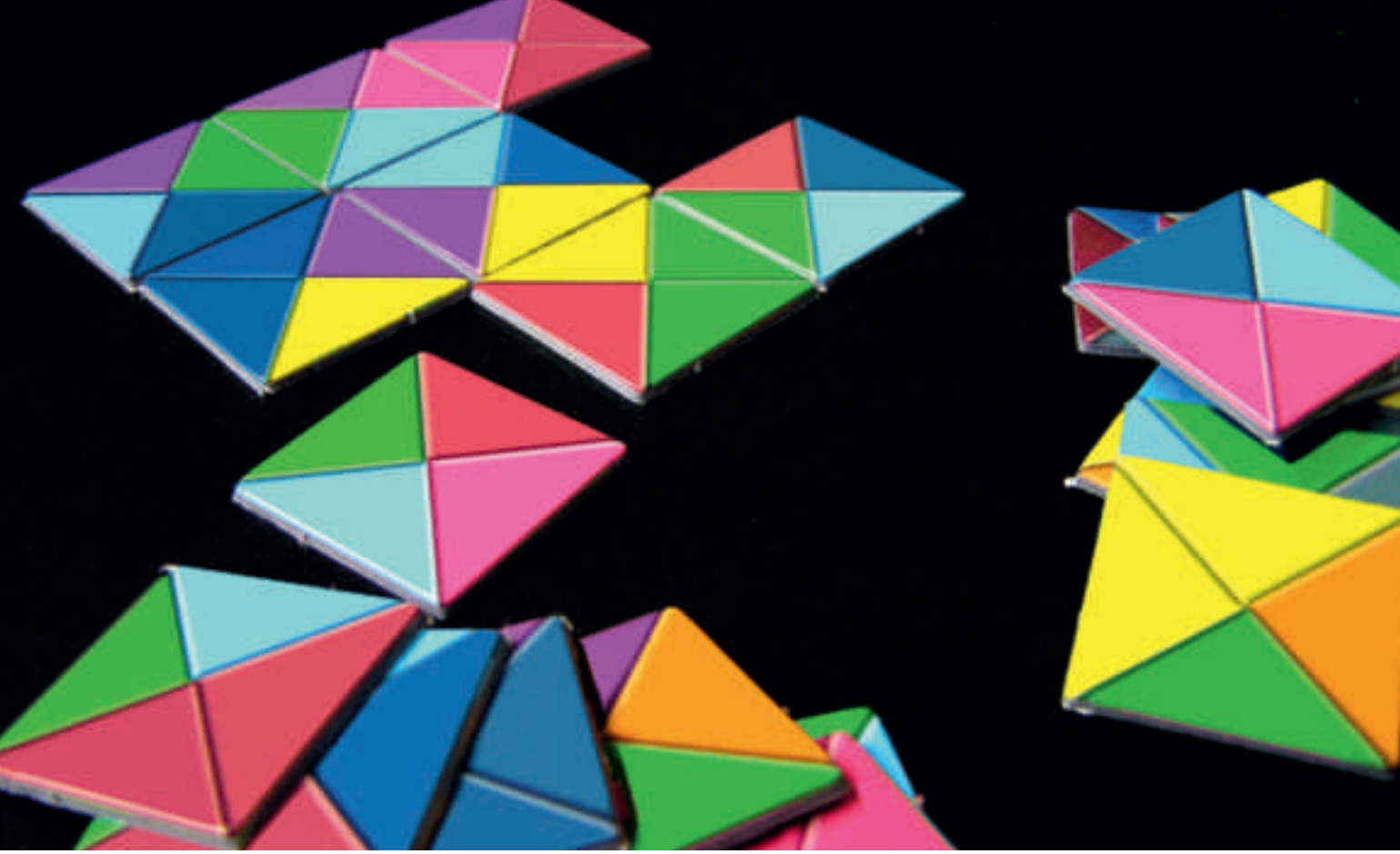


Un Puzle de 2 millones de dólares

Alberto Martín Goñi

Ingeniero Industrial del ICAI, Promoción 2009. Backoffice Comercial de Movianto España.

Jesús María Latorre Canteli

Ingeniero Industrial del ICAI (1995) y Doctor Ingeniero Industrial del ICAI (2007). Es investigador en el Instituto de Investigación Tecnológica.

Eugenio Fco. Sánchez Úbeda

Doctor Ingeniero Industrial del ICAI, Promoción 1991. Profesor de ICAI e Investigador en el IIT.

Palabras clave: Puzles, GRID, Problemas NP-Completo.

Resumen:

Este artículo estudia un tipo de puzles matemáticos, cuyo exponente principal es el puzle comercial denominado Eternity II. Se analiza la elevada complejidad de este tipo de puzles, caracterizándola como función del número de piezas y colores diferentes. Para confirmar estos hallazgos teóricos se ha desarrollado una herramienta informática capaz de generar automáticamente este tipo de puzles y resolverlos utilizando varios equipos en un entorno grid. Los resultados prácticos coinciden con los deducidos teóricamente, mostrando que existe un rango de número de colores, dependiente del tamaño del puzle, donde las estrategias de búsqueda requieren más tiempo para alcanzar una solución.

Key words: *Puzzles, GRID, NP-Complete Problems.*

Abstract

This paper studies a type of edge-matching puzzle, whose main exponent is the commercial puzzle called Eternity II. The high complexity of this type of puzzles has been discussed, as well as characterized as a function of the number of pieces and the amount of different colors. In order to confirm these theoretical findings, a software tool has been developed for generating and solving this kind of puzzles using several computers in a grid computing environment. The practical results agree with those deduced from theory, showing that there is a range in the number of colors, which depends on the puzzle size, where the search strategies require more time for achieving a solution.

Introducción

En 2000 Alex Selby y Oliver Riordan ganaron un millón de libras al encontrar una solución del puzle denominado Eternity. Este puzle constaba de 209 piezas con diferentes formas poligonales cuyo ensamblaje debía formar un dodecágono y cuya dificultad debería haber requerido, aparentemente, mucho más tiempo para resolverlo que los 7 meses empleados por los ganadores utilizando dos ordenadores personales.

En junio de 2007 apareció un nuevo puzle, denominado Eternity II, diseñado por los ganadores del anterior puzle. Éste consta de 256 piezas cuadradas cuyo ensamblaje debe formar un cuadrado de 16 x 16, siendo la imagen a formar desconocida. Cada lado de cada pieza tiene uno de los 23 colores posibles, existiendo un color concreto reservado para el borde exterior del puzle. En la Figura 1 se muestra un ejemplo de un puzle con el mismo número de colores y piezas que el Eternity II.

Este tipo de puzle matemático se puede entender como una abstracción de los puzles tradicionales en los que la posición de una pieza queda determinada por el encaje exacto de sus lados (en lo relativo tanto a la imagen como al contorno de la pieza). De hecho, en [Demaine2007] se demuestra que un puzle tradicional se puede transformar fácilmente en este tipo de puzle matemático. Sin embargo, la gran diferencia con los puzles tradicionales radica en que no se dispone de una imagen o patrón de partida que guíe con mayor o menor claridad la búsqueda de la solución.

En la Figura 2 se muestran cuatro puzles. El primero puede considerarse como muy fácil si se conoce previamente la imagen a formar ya que esa información permite localizar rápidamente la posición de la mayoría de las piezas, siendo necesario recurrir a la información proporcionada por el contorno de la pieza en raras ocasiones. El segundo puzle es un ejemplo de una situación en la que conocer con anterioridad la imagen a formar aporta mucha menos información, siendo un puzle mucho más complicado que el anterior. Finalmente, en el tercer y cuarto puzle la imagen visual contiene la misma información que la del contorno de las piezas, siendo trivial su resolución si se conoce previamente dicha imagen. Este último se corresponde con un caso especialmente sencillo ya que, aunque no se conozca la imagen de la solución, su resolución es directa al no existir dos piezas iguales. Un puzle matemático de tipo Eternity II es una abstracción de estos últimos puzles, siendo el objetivo del juego encontrar precisamente la imagen que representa una colocación correcta de las piezas en el tablero.

Figura 1. Ejemplo de puzle tipo Eternity II con 256 piezas y 23 colores

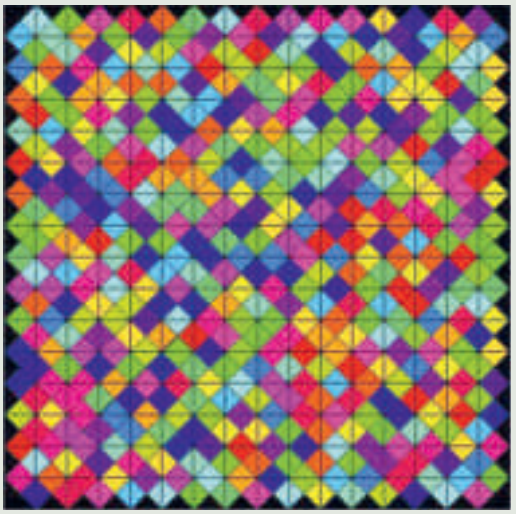
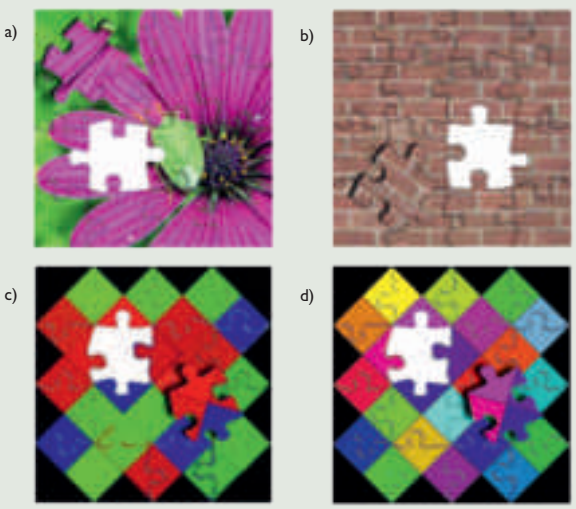


Figura 2: Puzles de menor a mayor complejidad: (a) puzle tradicional sencillo, (b) puzle tradicional más complejo, (c) puzle tipo Eternity II y (d) puzle tipo Eternity II trivial



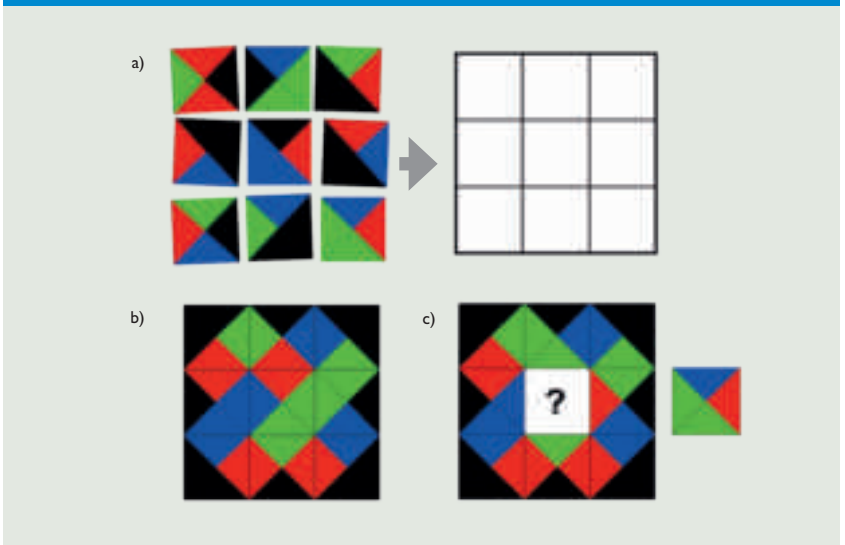
En realidad el puzle Eternity II es una variante del juego lúdico de ordenador denominado Tetravex, incluido en el entorno de escritorio Gnome de las distribuciones GNU/Linux y Unix (ver www.gnome.org). La diferencia fundamental reside en que el primero ha sido especialmente diseñado para que sea un problema intratable, incluso con la ayuda de un ordenador. Según los diseñadores del mismo, ni el ordenador actual más

potente podría encontrar una solución en todo el tiempo estimado de duración del universo.

Dificultad del problema

Resolver un puzle Eternity II de 4 colores y 9 piezas como el de la Figura 3 no debería requerir, en el peor de los casos, más de un minuto a una persona y unos pocos milisegundos a un ordenador. Sin embargo, 29 meses después de su lanzamiento, en diciembre de 2009, no se había encontrado una solución al puzle de 256 piezas y 23 colores, a pesar de tener asociado un premio de dos millones de dólares y de que se haya ido publicando la posición exacta de algunas piezas para reducir la complejidad del mismo. Esta búsqueda ha sido infructuosa debido a que la posición exacta de cada pieza en el tablero no se puede asegurar hasta tener completamente resuelto el puzle, y a medida que aumenta el tamaño del puzle existen muchas más posibilidades de tener que deshacer todo el trabajo realizado colocando piezas al llegar a un punto donde es imposible seguir. En la Figura 3 se muestra una posible solución, así como una situación en la

Figura 3: Ejemplo de puzle tipo Eternity II con 9 piezas y 4 colores: (a) Piezas inicialmente desordenadas y tablero, (b) Una posible solución, (c) Ejemplo de un intento fallido



que al intentar colocar la última pieza se descubre que no es posible completar el puzle correctamente.

Una manera sencilla de cuantificar cómo aumenta la dificultad del puzle con el número de piezas consiste en calcular el porcentaje de piezas del interior con respecto a las del borde y esquinas (es decir, comparar el perímetro del tablero con su superficie). Según la Figura 4 se puede afirmar que los puzles de menos de 49 piezas tienen más piezas en el contorno que en el interior. A partir de ese tamaño empieza a haber más piezas en el interior que en el contorno. Dado que las piezas del contorno tienen su orientación en la solución final totalmente fijada por el color negro del borde, se puede afirmar que cuantas

Figura 4. Porcentaje de piezas de cada tipo al variar el tamaño del puzle

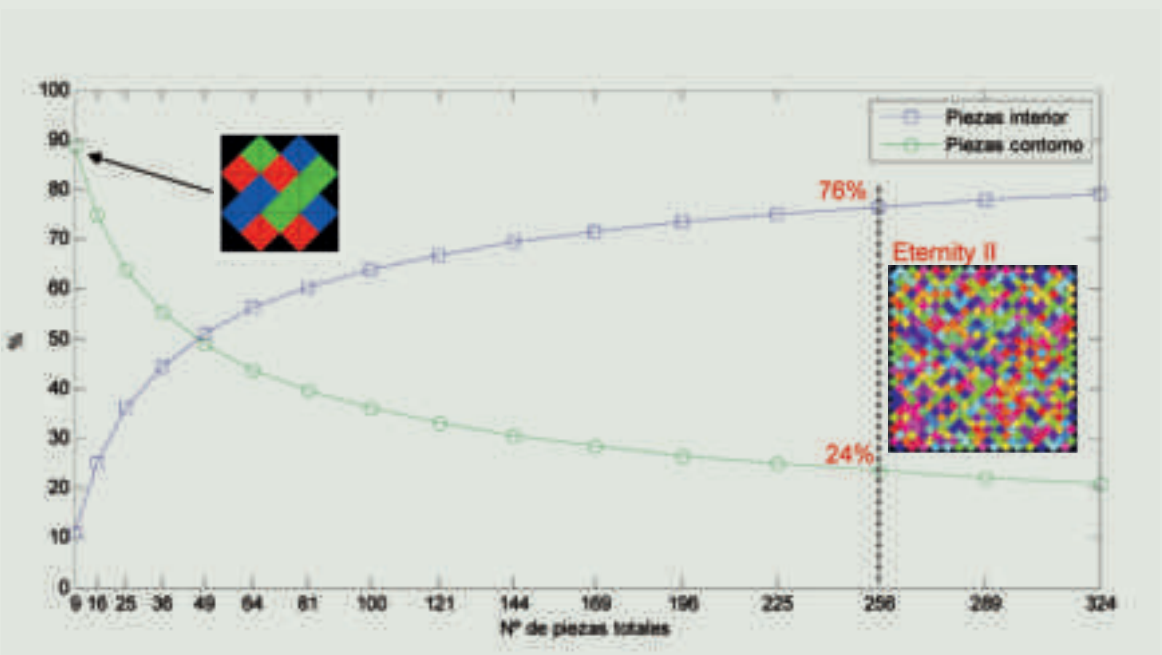
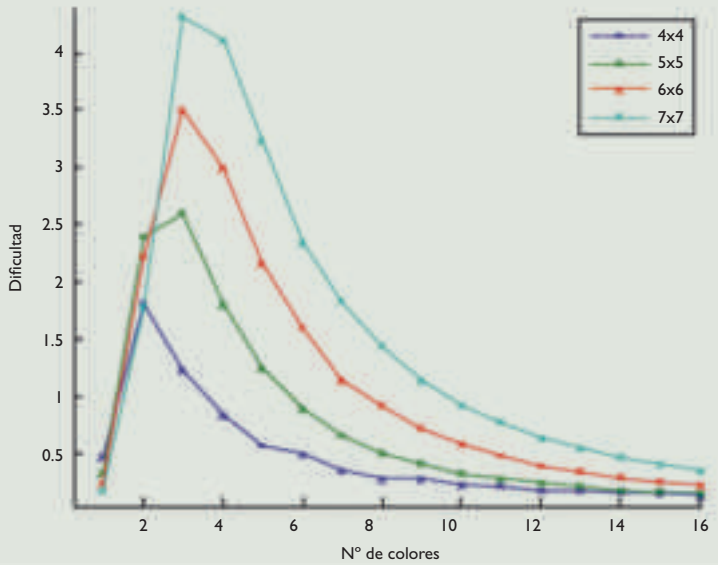


Figura 5. Variación del grado de dificultad con el número de piezas y colores



más piezas haya que colocar en el centro más complejo será.

Por otro lado, el número de colores diferentes en el puzle está directamente relacionado con la dificultad del mismo. Existen dos situaciones extremas donde el puzle no tiene ninguna dificultad. Si únicamente hay dos colores (el del borde exterior y el del interior), entonces simplemente hay que distinguir entre las piezas de las esquinas, laterales y del centro para poder colocarlas sin mayor complicación ya que su posición es indiferente. El caso contrario sería cuando en el puzle aparece el número máximo posible de colores diferentes y también sería muy sencillo de resolver ya que colocada una de las esquinas, la posición del resto de piezas estaría determinada de forma unívoca (ver Figura 2d). Para un puzle de N piezas, el número máximo de colores diferentes es $1+2\cdot(N-\sqrt{N})$ (ver [Martín2009]). Así, un puzle de 9 piezas sería trivial si consta de 2 ó 13 colores. Para un número de colores intermedio se pueden encontrar puzles de dificultad variable.

Realmente la dificultad del puzle depende de manera conjunta del número de piezas, del número de colores diferentes y de la distribución de los mismos en las piezas. En la Figura 5 se muestra una estimación de esta dificultad (obtenida a partir de varias simulaciones con diferente distribución de colores en las piezas), para cuatro tamaños de puzle y en función del número de colores [Martín2009]. Como se puede apreciar, el máximo de la curva se desplaza hacia la derecha y aumenta su valor a medida que crece el número de piezas del puzle.

Un puzle matemático como el Eternity II es extremadamente difícil de resolver. El número de piezas y de colores, así como su distribución espacial ha sido elegido cuidadosamente para que su solución no pueda ser encontrada en un tiempo razonable, ni utilizando la potencia de cálculo de los ordenadores actuales. La razón fundamental para ello es que no es posible diseñar un algoritmo capaz de encontrar la solución en un tiempo de ejecución que crezca de forma polinómica con el tamaño del problema. Si el tamaño es suficientemente grande, el tiempo de ejecución del algoritmo será inmenso. Más formalmente, en el campo de la complejidad computacional, a un problema de estas características se le denomina NP-Completo (del inglés *Non-Polynomial* [Garey1979]).

Existen muchos problemas NP-Completos, como el aparentemente sencillo problema de determinar si en un conjunto de *M* números enteros (positivos y negativos) existe un subconjunto no vacío de suma nula. Aunque es fácil verificar si una posible respuesta es correcta, el algoritmo de búsqueda requiere explorar todos los 2^M-1 subconjuntos posibles hasta encontrar uno que cumpla la condición. Esta dependencia exponencial con el tamaño del problema hace que el problema sea intratable en la práctica, incluso con valores de *M* pequeños.

El crecimiento exponencial de la complejidad escapa fácilmente a la intuición, por lo que conviene aclarar la magnitud del problema con cifras concretas. Según una conocida leyenda, si en cada una de las 64 casillas de un tablero de ajedrez se coloca el doble de los granos de arroz que en la casilla anterior (comenzando con un grano en la primera casilla), en el tablero habría un total de $2^{64}-1 \approx 18\cdot10^{18}$ granos de arroz. Exactamente 18.446.744.073.709.551.615 granos de arroz, el número más grande que se puede almacenar en un entero sin signo con una representación de 64 bits de los nuevos ordenadores personales. Suponiendo que un grano de arroz pesa 20 miligramos, (una tonelada tendrá 50 millones de granos) y teniendo en cuenta que la producción mundial de arroz en 2008 fue aproximadamente de 700 millones de toneladas, en el tablero de ajedrez tendría que estar todo el arroz producido en la Tierra durante los próximos 527 años, suponiendo una producción anual constante igual a la de 2008.

Al igual que en el caso del tablero de ajedrez, una excesiva ingenuidad puede llevar a infravalorar el crecimiento de la complejidad de un puzle Eternity II con el tamaño del mismo, albergando expectativas de resolución cuestionables. En [Takenaga2006] se demuestra teóricamente que, en general, la búsqueda de una solución de un puzle matemático como el Eternity II también es un problema NP-Completo, es decir, irresoluble a día de hoy.

La dificultad de este tipo de puzles radica en el tamaño del espacio de búsqueda y en el número de soluciones

posibles. Cuando el espacio de búsqueda es inmenso y el número de soluciones posibles es muy pequeño, como en el caso del Eternity II, se trata de encontrar una aguja muy pequeña en un pajar muy grande. Si se intentara resolver dicho puzle simplemente colocando las piezas totalmente al azar y comprobando a continuación si es la solución válida o no, el espacio de búsqueda a explorar sería de $4^N \cdot N!$, que para N=256 resulta en un tamaño aproximado de $2 \cdot 10^{660}$, un número astronómico de combinaciones. Encontrar una solución con este método de búsqueda de fuerza bruta sería al menos tan difícil como encontrar un átomo concreto en todo el universo observable.

Si se utiliza una estrategia de búsqueda un poco más inteligente, en la que las piezas se clasifican previamente en sus tres tipos básicos (esquinas, bordes y centro) y luego se colocan al azar en la zona correspondiente teniendo en cuenta el color del borde para evitar explorar soluciones que son infactibles a priori, el espacio de búsqueda se reduce significativamente, pero sigue siendo inmenso, del orden de 10^{560} para 256 piezas (ver [Martín2009], [Owen2007]).

Los algoritmos sofisticados especialmente diseñados para encontrar una solución a este problema se basan en ir colocando piezas de forma lógica y ordenada hasta llegar al final o encontrar una infactibilidad, en cuyo caso es necesario deshacer el camino recorrido y emprender uno nuevo. Determinar el espacio de búsqueda de estos algoritmos no es sencillo, pero según [Owen2007] podría ser del orden de 10^{47} para el Eternity II.

Modelado del problema

El primer paso para poder tratar automáticamente este problema consiste en encontrar una manera adecuada de expresarlo. Una forma compacta de representarlo que facilita los cálculos es la matriz. En dicha matriz cada fila representa una pieza del puzle, y para una misma fila las diferentes columnas almacenan el color de cada uno de los lados de la pieza representada. Este enfoque es extensible a piezas con mayor número de lados. Por ejemplo, también es posible cubrir una superficie con un patrón de piezas regulares empleando triángulos equiláteros y hexágonos, que con este esquema se representarían como matrices de tres o seis columnas, respectivamente. Por el contrario, si se considerasen patrones con piezas irregulares como las que se pueden encontrar en La Alhambra de Granada (Figura 6), debería encontrarse otra manera de representar las piezas que se ajustase mejor a esos perfiles.

Figura 6. Ejemplo de mosaico de La Alhambra (Patio de los Arrayanes)

Para identificar el color de un lado de una pieza, se emplea un número entero que lo representa de manera unívoca en todo el puzle. Por tanto, ese entero aparecerá en la matriz que representa el puzle en tantas ocasiones como veces se emplee ese color en las piezas del puzle. En la Tabla 1 se presenta la codificación de las piezas del ejemplo de puzle, que se muestra en la Figura 3(a) como si estuviesen colocadas en el tablero. Nótese que esta codificación permite representar piezas con bordes complejos, como los que aparecen en los puzles actuales con entrantes y salientes. Simplemente es necesario realizar la transformación de cada borde diferente al código de color correspondiente.

La comprobación del ajuste de las piezas utilizando esta representación se reduce a comparar los valores de las diferentes piezas entre sí. Por ejemplo, para comprobar si el lado izquierdo de la pieza 1 encaja con el lado derecho de la pieza 2 hay que comprobar si coincide el valor de la fila 1 y columna 4 con el valor de la fila 2 y columna 2. De forma similar, cuando se gira una pieza eso se refleja en un desplazamiento circular de los valores dentro de su fila, según el sentido de giro.

Con esta codificación matricial de las piezas y el tablero del puzle se tiene una representación muy eficaz, donde la comprobación de encaje de dos piezas supone la comparación de dos valores numéricos, y el giro de una pieza, un desplazamiento de valores dentro de una misma fila. Todas estas operaciones de tipo matricial son gestionadas de forma eficiente por los

ordenadores, y por ello el algoritmo de resolución puede dedicar el tiempo a la búsqueda de la solución.

Algoritmos de resolución

Para plantear la búsqueda de la solución es necesario saber con qué información se cuenta para guiar esa búsqueda. De manera local es posible conocer qué piezas encajan con las que ya están colocadas. Sin embargo, no es posible contar con una información global que indique la idoneidad de una pieza candidata frente a otra, ya que, como se ha comentado anteriormente, no se tiene una imagen que haya que reconstruir con los colores de las piezas.

Para garantizar que una pieza está correctamente colocada en el tablero debe obtenerse la solución completa, ya que hasta que todas las piezas estén colocadas no se puede asegurar que se ha encontrado la solución. Incluso tener más piezas encajadas en el puzle no supone más certeza de tener su solución, simplemente se está más próximo de saber si es solución o no. De hecho, el proceso de solución bien podría llegar a un tablero completo salvo por una pieza, y que la secuencia de colores en la pieza que falta por colocar no coincida con la secuencia de colores del hueco que queda en el puzle. Un ejemplo de esto último se puede ver en la Figura 3(c), donde la pieza que falta contiene los colores del último hueco, pero en orden incorrecto, y por ello no encaja. Como contraste, obsérvese que en la Figura 3(b) una recolocación de las piezas conduce a una situación final distinta, en la que la pieza central sí encaja, y por tanto supone una solución del puzle.

De esta manera, para garantizar que la decisión tomada en cada paso es correcta, debe comprobarse si el espacio de búsqueda que esas piezas determinan contiene la solución. Como se ha visto anteriormente, el número total de combinaciones de piezas (o posibles soluciones a comprobar) crece de forma exponencial con el número de piezas y no es posible explorarlo de forma exhaustiva en un tiempo razonable. En consecuencia, es necesario recurrir a criterios heurísticos que sirvan de guía aproximada, cuyas decisiones son posteriormente comprobadas durante el proceso de búsqueda. El resultado de estas comprobaciones puede indicar que hasta el momento se ha podido avanzar sin problemas, o que con las piezas colocadas hasta ese momento no se puede encontrar una pieza libre que encaje. En este último caso, alguna de las decisiones anteriormente tomadas no es correcta, y por tanto hay que revisarlas y escoger una alternativa distinta.

Tabla 1. Codificación de los colores de las piezas del puzle: 0 = negro, 1 = rojo, 2 = verde y 3 = azul

Código de color de los lados				Posición de la pieza	Pieza
Superior	Derecho	Inferior	Izquierdo		
1	0	1	2	(1,1)	1
3	2	2	0	(1,2)	2
2	1	0	0	(1,3)	3
0	0	3	1	(2,1)	4
0	1	3	3	(2,2)	5
1	3	0	0	(2,3)	6
2	0	3	1	(3,1)	7
3	0	0	1	(3,2)	8
3	1	2	2	(3,3)	9

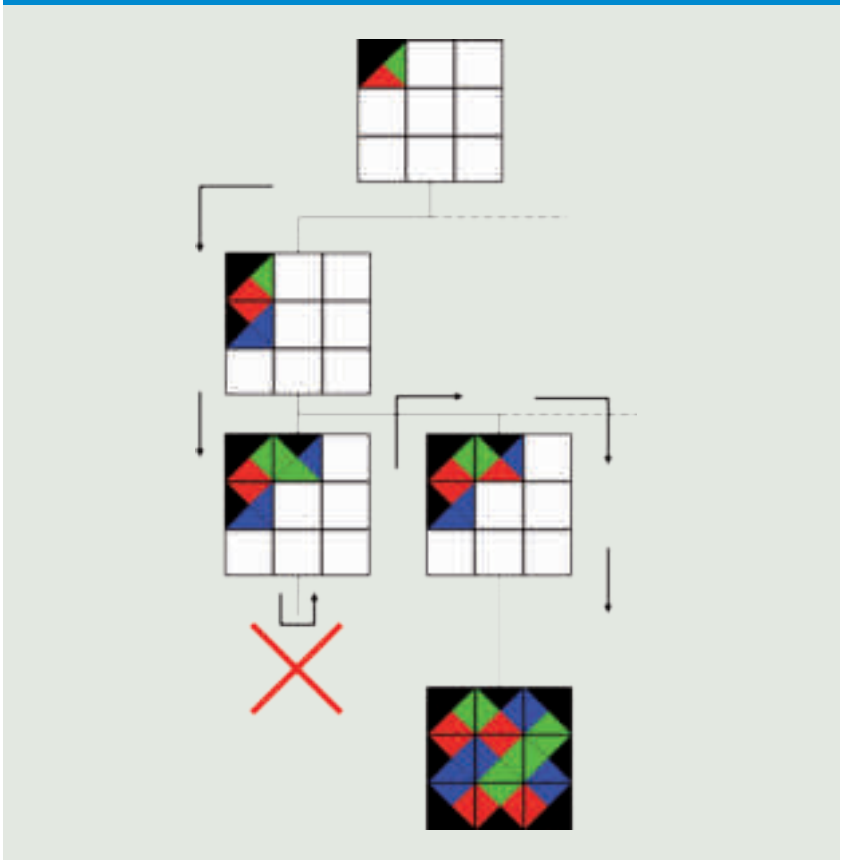
En las pruebas que se van a presentar en este artículo se ha empleado como criterio heurístico el colocar en cada momento una pieza en la posición en la que menos piezas encajen. Como en esa posición hay menos alternativas, cada una de ellas cuenta con mayores probabilidades de pertenecer a la solución. En cierto sentido, con esto se deja el camino más abierto para las futuras decisiones y es más probable que por ese camino se encuentre una solución.

Considerando todo lo anterior, el algoritmo de búsqueda de la solución que se ha diseñado incorpora los siguientes factores:

- En primer lugar, se utiliza un algoritmo voraz, es decir, al escoger qué pieza colocar y en qué posición, se toma una decisión óptima localmente. Es preciso escoger las piezas que encajen con las ya colocadas, evaluando la bondad de las posibles alternativas empleando el heurístico mencionado anteriormente: se selecciona la posición en la que menos piezas encajan, y por tanto, la posición en la que una pieza colocada tiene más posibilidades de ser la correcta.
- En segundo lugar, como puede ocurrir que alguna de las decisiones tomadas previamente sea incorrecta, el algoritmo debe permitir la vuelta atrás para considerar otras alternativas diferentes a las ya consideradas.

En la Figura 7 se muestra un ejemplo de este proceso de búsqueda para el puzle de la Figura 3(a), indicado por las flechas (de arriba a abajo y de izquierda a derecha). En el primer tablero se coloca la primera pieza, en la esquina superior izquierda. Posteriormente se escoge colocar la pieza que está debajo de la anterior, descendiendo por la rama de la izquierda en el diagrama que representa el proceso de búsqueda. En ese momento existen otras alternativas que se han señalado con una línea de trazos que se extiende hacia la derecha. Al seguir por la rama de la izquierda y colocar la tercera pieza del puzle a la derecha de la pieza de esquina, se llega a una vía muerta en que ya no es posible colocar la pieza central. Por ello, se deben reconsiderar las

Figura 7. Ejemplo de búsqueda de la solución (condensada)



decisiones tomadas anteriormente. En concreto, se toma una alternativa a la última pieza que se decidió poner colocando, a la derecha de la que ocupa la esquina, otra posible como tercera pieza. Siguiendo este proceso de búsqueda por ese camino, se llega finalmente a una solución del puzle.

Implementación utilizando un GRID

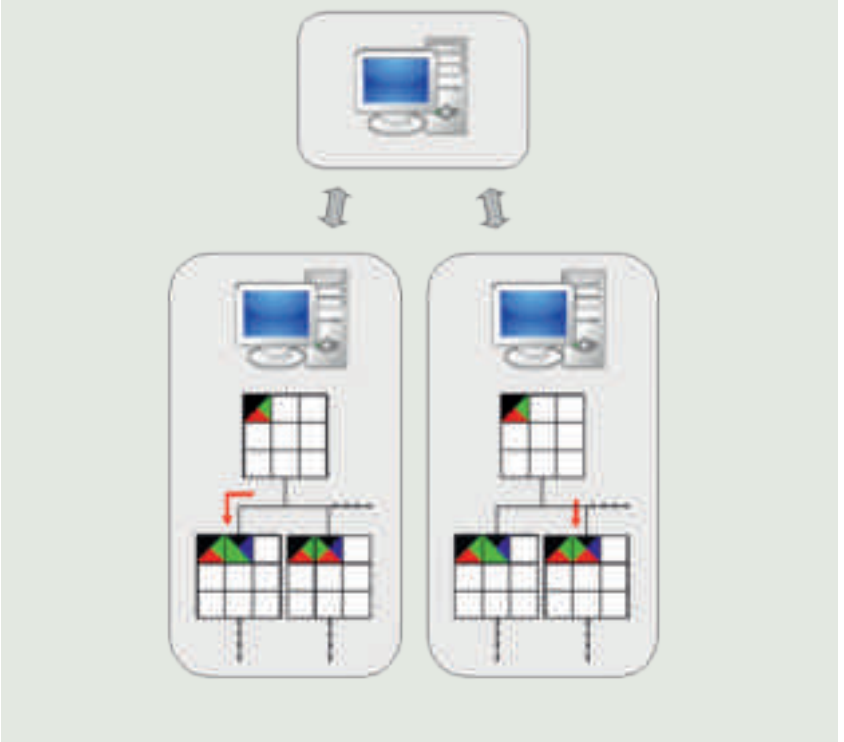
Un entorno grid se puede definir como un sistema de cálculo formado por varios equipos interconectados, dispersos geográficamente y que actúan en paralelo [Foster1998]. Este tipo de sistemas se emplea en aquellos procesos de cálculo donde es posible trocear el problema a gran escala en varios subproblemas y emplear cada uno de los equipos para resolverlos independientemente. Si se coordinan correctamente los equipos, es posible reducir el tiempo total de procesamiento de forma proporcional al número de equipos de los que conste el grid.

Cuando se emplea un grid para buscar la solución del puzle, es necesario adaptar el algoritmo de búsqueda para que permita:

- Avanzar en la búsqueda de la solución de forma independiente en cada ordenador, sin que se produzcan solapes en las posibles soluciones exploradas por cada uno.
- Interrumpir y reanudar la búsqueda, analizando cada vez un número dado de posibles soluciones. De esta forma se pueden asignar flexiblemente estas tareas a los equipos del grid, y parar cuando uno de ellos encuentre la solución.

En la Figura 8 se muestra esquemáticamente cómo se ejecutaría una unidad de trabajo en el grid. El equipo en la parte superior de la figura es el gestor del grid, que envía al resto de equipos (servidores de cálculo) la solución parcial inicial de la que debe partir cada uno. Éstos dan un número de pasos de búsqueda de la solución (colocando piezas o buscando alternativas a las ya colocadas), ensayando la colocación de piezas distintas en cada equipo. Si no encuentran la solución al final del

Figura 8. Implementación en GRID del algoritmo de búsqueda (esquema)



número de soluciones que pueden analizar, devuelven al gestor del grid la situación en la que han quedado. Éste vuelve a asignar esas soluciones parciales devueltas como posiciones iniciales a los ordenadores que hayan quedado libres, y así continúa hasta que algún servidor de cálculo finalmente encuentra la solución.

Resultados experimentales

El enfoque de búsqueda expuesto anteriormente ha sido validado de forma experimental utilizando un grid gestionado por NetSolve bajo Linux [Seymour2005]. Para ello se ha desarrollado en lenguaje C, una herramienta de resolución de este tipo de puzles. Además, para poder disponer de puzles de dificultad variable, se ha completado con un generador automático de puzles aleatorios en el que se puede controlar tanto el tamaño como el número de colores de los mismos. La combinación de estas dos herramientas ha permitido realizar diferentes análisis sobre la complejidad del problema, todos ellos recogidos en [Martín2009].

Por ejemplo, en la Figura 9 se ha representado en escala logarítmica el tiempo necesario para encontrar la solución en un grid formado por varios ordenadores personales de doble núcleo al variar el tamaño del puzle y el número de colores. Este tiempo se ha calculado como promedio de los tiempos obtenidos con varios puzles generados con diferentes distribuciones de colores. Se puede observar que estos resultados son coherentes con los obtenidos teóricamente mediante simulación (Figura 5), así como con los resultados prácticos que aparecen en [Ansotegui2008a] y [Ansotegui2008b].

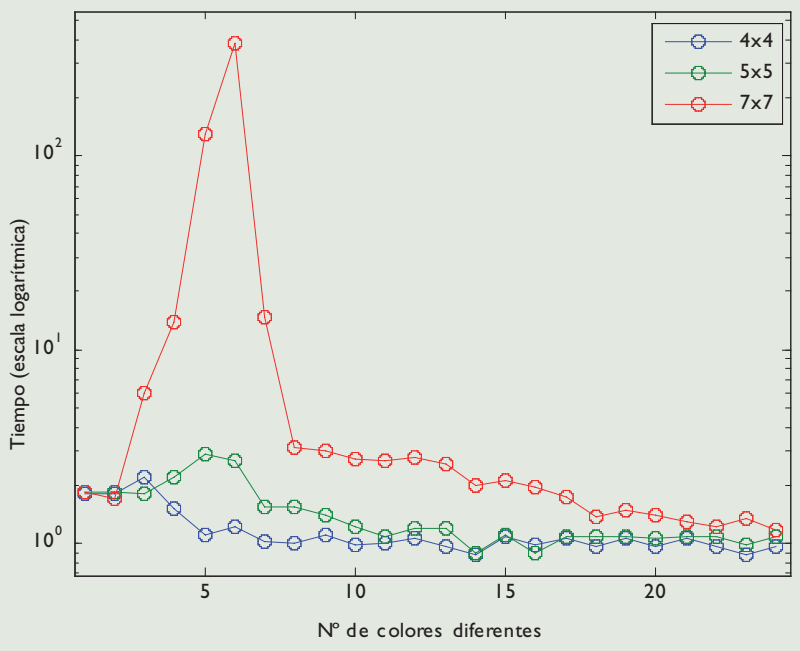
Conclusiones

En este artículo se ha presentado el problema de resolución de puzles y se ha estudiado qué parámetros influyen en su complejidad. Se ha determinado que influyen el número de piezas, el número de colores diferentes y su distribución entre las piezas. Además, se ha podido ver que existe un rango de colores, dependiente del tamaño del puzle, en el que las estrategias requieren más tiempo para alcanzar una solución.

Para resolver este problema se ha usado un grid, que permite disponer de una capacidad de cálculo adicional para resolver más rápidamente los puzles. Este entorno de cálculo es idóneo para el algoritmo de resolución empleado, que es de grano grueso, porque permite realizar la búsqueda simultánea considerando diferentes alternativas.

Todos los estudios realizados apuntan a que el puzle Eternity II ha sido especialmente diseñado para ser imposible de resolver con los recursos computacionales existentes en la actualidad, requiriendo pistas para reducir la dificultad del mismo y permitir así encontrar la solución. ∎

Figura 9. Variación del tiempo de búsqueda con el tamaño del puzle y el número de colores



Bibliografía

Ansotegui2008a Ansótegui, C., Béjar, R., Fernández, C., y Mateu, C., *"How Hard is a Commercial Puzzle: the Eternity II Challenge"*, en *Proc. of the 2008 Conference on Artificial Intelligence Research and Development*, Alsinet, T. , Puyol-Gruart, J., y Torras, C., (editores), vol. 184. páginas 99-108, IOS Press, Amsterdam, 2008.

Ansotegui2008b **Ansótegui, C., Béjar, R., Fern**àndez, C., Mateu, C., *"Edge matching puzzles as hard sat/csp benchmarks"*, en *Principles and Practice of Constraint Programming*, páginas 560-565, 2008.

Demaine2007 **Demaine, E., Demaine, M., "Jig**saw Puzzles, Edge Matching, and Polyomino Packing: Connections and Complexity", Kyoto International Conference on Computational Geometry and Graph Theory 2007, Kyoto, Japón, junio 2007.

Foster1998 Foster, I., y Kesselman, C., *"The Grid: Blueprint for a New Computing Infrastructure"*, Morgan Kaufmann, 1998.

Garey1979 Garey, M. and D. Johnson, *"Computers and Intractability; A Guide to the Theory of NP-Completeness"*, 1979.

Latorre2007 Latorre, J. M., *"Resolución distribuida de problemas de optimización estocástica. Aplicación al problema de coordinación hidrotérmica"*, tesis doctoral, Universidad Pontificia Comillas, 2007.

Martín2009 Martín, A., *"Análisis de estrategias de resolución de puzles en entornos grid"*, proyecto fin de carrera, Universidad Pontificia Comillas, 2009.

Owen2007 Página web de Brendan Owen sobre el puzle Eternity II: www.eternityii.mrowen.net/solving.html

Seymour2005 Seymour, K., YarKhan, A., Agrawal, S., Dongarra, J., *"NetSolve: Grid enabling scientific computing environments"*, en *Advances in Parallel Computing*, Volume 14, páginas 33-51, ed. Lucio Grandinetti, Holanda, 2005.

Takenaga2006 Takenaga, Y. y Walsh, T., *"TETRAVEX is NP-complete"*, en Information Processing Letters, 99, 5, páginas 171-174, septiembre 2006.