Un algoritmo genético híbrido para resolver el EternityII

Rico, Martin; Ros, Rodrigo
Directora: Prof. Dra. Irene Loiseau

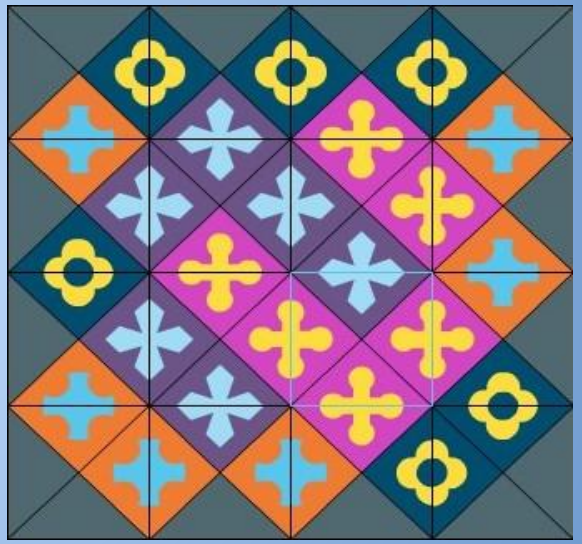
Temas

- **Introducción**
  - Eternity II
    - Historia
    - Descripción
  - Metaheurísticas
    - Algoritmos Evolutivos
    - Algoritmos Genéticos
    - Búsqueda Local
- **Algoritmo Propuesto**
  - Pseudocodigo
  - Explicación
- **Demo**
  - Demostración
  - Notas de implementación
- **Resultados**
  - Casos de prueba
  - Comportamiento observado
  - Resultados
  - Comparación con trabajos relacionados
- **Conclusiones**

Eternity II

- Juego de mesa, creado por Christopher Monckton.
- Tipo Edge-Matching de 256 piezas cuadradas (16x16).
- Cada pieza tiene sus bordes marcados con diferentes combinaciones de forma/color.
- Las piezas que se ubican en los bordes son de color gris.
- Existen 22 combinaciones de forma/color sin contar los bordes.


ETERNITY II
the amazing puzzle with an incredible prize
ETERNITY



Historia

En el año 1999 Christopher Monckton lanzó la primer versión, Eternity I.

Fue resuelto en el año 2000 por dos matemáticos de Cambridge, Alex Selby y Oliver Riordan.

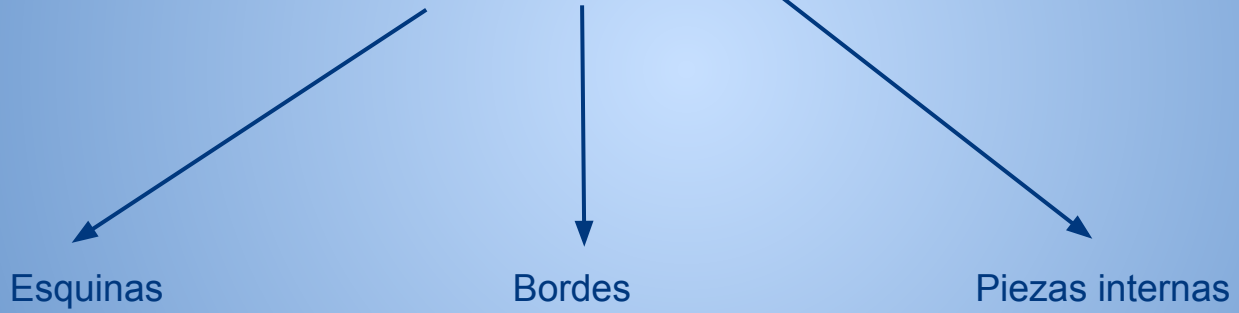
En 2007, Monckton lanzó la segunda versión del juego, Eternity II con un premio de 2 millones para quien lo pudiese resolver. Esta versión mejora las debilidades de la primer versión.

Hasta el momento no se conoce solución completa.

Descripción del problema

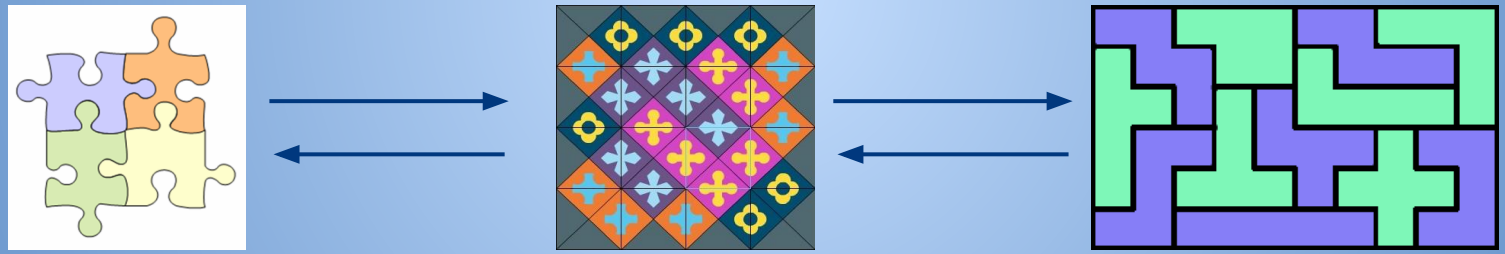
¿Qué tan complejo es el problema?

$$4! + 56! + 196! * 4^{196}$$



Aproximadamente $5{,}12 * 10^{483}$ posibles soluciones!

Erik y Martin Demaine demostraron que existe una equivalencia directa entre rompecabezas del tipo Jigsaw, Edge Matching y Polyomino Packing y que la resolución de todos ellos es NP-completo.



Metaheurísticas

Son métodos diseñados para encontrar buenas soluciones a problemas de optimización combinatoria en general. No hay una definición única para este término:

- “Las metaheurísticas son estrategias de alto nivel que guían una heurística específica del problema a resolver para mejorar su perfomance.”
- “Una metaheurística es un conjunto de conceptos que pueden ser usados para definir algoritmos heurísticos para un amplio espectro de problemas diferentes.”

Características

- Son estrategias que guían procesos de búsqueda.
- Sus conceptos se pueden describir con un gran nivel de abstracción. No son para un problema específico.
- En muchos casos son algoritmos no-determinísticos.
- Sus desarrollos y diseños suelen estar motivados por comportamientos naturales.
- No garantizan que una solución óptima sea encontrada.
- Las técnicas metaheurísticas van desde algoritmos simples de búsqueda local a complejos procesos de aprendizaje.

Algunas Técnicas

- Simulated Annealing
- Tabu Search
- GRASP
- Algoritmos evolutivos
- Colonia de hormigas
- Variable Neighborhood Search
- Iterated Local Search
- Etc.

- Híbridos

Algoritmos Evolutivos

Basados en la evolución natural. Consisten en generar conjuntos de posibles soluciones del problema que irán evolucionando iterativamente con la esperanza de que los nuevos conjuntos contengan soluciones mejores a las anteriores.

En el contexto de los algoritmos evolutivos, a estos conjuntos se los denomina *poblaciones* y a cada una de las soluciones se las denomina *individuo*. Para medir la calidad de un individuo, es decir, la calidad de una solución, se utiliza una *función de evaluación*, también llamada *fitness*.

Estructura general del algoritmo

1: *poblacion* ← generar una población inicial de individuos
2: evaluar el fitness de cada individuo de la *poblacion*
3: **mientras** algun criterio de parada **hacer**
4: *padres* ← seleccionar individuos de *poblacion*
5: *hijos* ← generar individuos a partir de operaciones geneticas sobre *padres*
6: evaluar el fitness de cada individuo en *hijos*
7: *poblacion* ← crear una nueva poblacion a partir de *poblacion* e *hijos*
8: **fin mientras**

Clasificación

Suelen clasificarse en cuatro grandes grupos según implementación y naturaleza del problema:

- Algoritmos Genéticos
- Programación Genética
- Programación Evolutiva
- Estrategias Evolutivas

Algoritmos Genéticos

- Utilizados con mayor frecuencia dentro de la familia de los algoritmos evolutivos.
- Suelen aplicarse a problemas de optimización combinatoria.
- Los individuos son denominados *cromosomas*. A cada parte o atributo del cromosoma se lo denomina *gen*.
- Las operaciones que dan lugar a nuevos cromosomas son:
  - *Mutación*: Consiste en la alteración de uno o varios genes del cromosoma.
  - *Cruzamiento (Crossover)*: Consiste en tomar más de una solución padre y generar, a partir de ellas, uno o más cromosomas hijos mediante el cruzamiento de genes.

Para construir un algoritmo genético específico para un problema se necesita:

- Representación genética de las soluciones (Cromosomas).
- Forma de generar la población inicial.
- Función de evaluación o fitness.
- Operadores genéticos que alteren la composición de los hijos. (Crossover, Mutación).
- Determinación de parámetros.

Búsqueda Local

- Son métodos metaheurísticos de optimización que, a diferencia de los anteriores, parten de una única solución.
- Comienzan con una solución candidata a un problema determinado y reiteradamente se mueven bajo algún criterio de optimización a una solución vecina hasta cumplir con algún criterio de terminación.
- La elección de la solución vecina se realiza utilizando información exclusivamente del vecindario de la solución candidata, de ahí el nombre de búsqueda local.

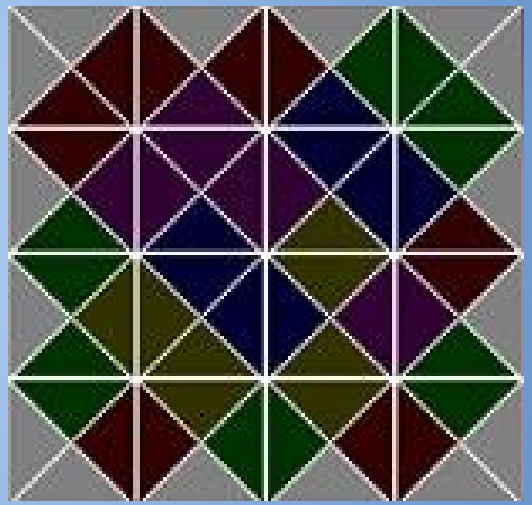
Algoritmo Propuesto

1: $iteracion \leftarrow 1$
2: $poblacion \leftarrow$ generar población inicial
3: evaluar el fitness de cada individuo de la $poblacion$
4: **mientras** $iteracion \leq MAXITERACIONES$ **hacer**
5: $padres \leftarrow$ seleccionar aleatoriamente individuos
6: **si** $iteracion$ mod $150 = 0$ **entonces**
7: $hijos \leftarrow$ realizar busqueda local sobre $padres$
8: **si no**
9: $hijos \leftarrow$ realizar operaciones geneticas sobre $padres$
10: **fin si**
11: evaluar el fitness de cada individuo en $hijos$
12: $poblacion \leftarrow$ crear una nueva poblacion a partir de $poblacion$ e $hijos$
13: $iteracion \leftarrow iteracion + 1$
14: **fin mientras**

Representación

- El tablero se representa con una matriz (cromosomas).
- Las piezas se representa con tuplas (genes).
- Las tuplas contiene números del 0 al 22 representando la combinación forma/color.
- Primer posición representa el borde superior, en la segunda al borde derecho, en la tercer el borde inferior y en la cuarta el borde izquierdo.
- El número 0 representa el borde del rompecabeza.

$$\mathbf{X} = \begin{pmatrix} <0,1,1,0> & <0,1,2,1> & <0,3,4,1> & <0,0,3,3> \\ <1,2,3,0> & <2,2,4,2> & <4,4,5,2> & <3,0,1,4> \\ <3,5,3,0> & <4,4,5,5> & <5,2,5,4> & <1,0,3,2> \\ <3,1,0,0> & <5,3,0,1> & <5,1,0,3> & <3,0,0,1> \end{pmatrix}$$



Población Inicial

- Las piezas se ubican de forma aleatoria en el tablero.
- Los bordes y las esquinas son ubicados de forma tal que coincidan con los bordes del tablero.
- Para no tener soluciones repetidas se fija la ficha de la esquina superior derecha.

Función de Evaluación

- Cuenta la cantidad de los bordes de las piezas adyacentes que coinciden en forma y color.
- No se toman en cuenta los bordes ni las esquinas.

$$Ft(X) = \sum_{\substack{1 \le i \le n \\ 1 \le j < n}} \left(iguales(\Pi_r(x_{i,j}), \Pi_l(x_{i,j+1})) \right) + \sum_{\substack{1 \le i < n \\ 1 \le j \le n}} \left(iguales(\Pi_b(x_{i,j}), \Pi_t(x_{i+1,j})) \right)$$

- El valor máximo de la función es 2 x n x (n - 1).
- Este valor representa la solución completa.
- En el caso de Eternity el valor de la solución completa es 480.

Selección

Para seleccionar los individuos de una población que serán utilizados como padres en la generación de la nueva población probamos dos técnicas:

- Roulette Wheel Selection
- Random Selection

Finalmente utilizamos Random Selection, ya que los fitness proporcionados por ambas implementaciones eran muy similares, pero Random Selection nos arrojaba mejores resultados en el rendimiento.

Operadores

Los operadores propuestos siempre generan soluciones válidas:

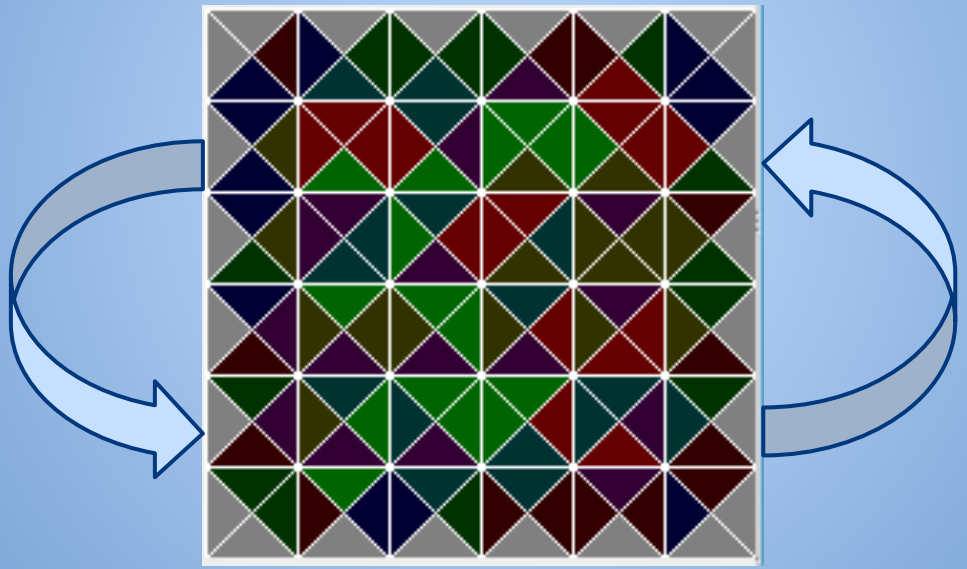
- No se mezclan piezas entre dos rompecabezas.
- Las piezas bordes y esquinas siempre se encuentran en esas ubicaciones.
- Las piezas bordes y esquinas se ubican de forma tal que sus bordes
  grises coincidan con los bordes del tablero.

Operadores: Mutación

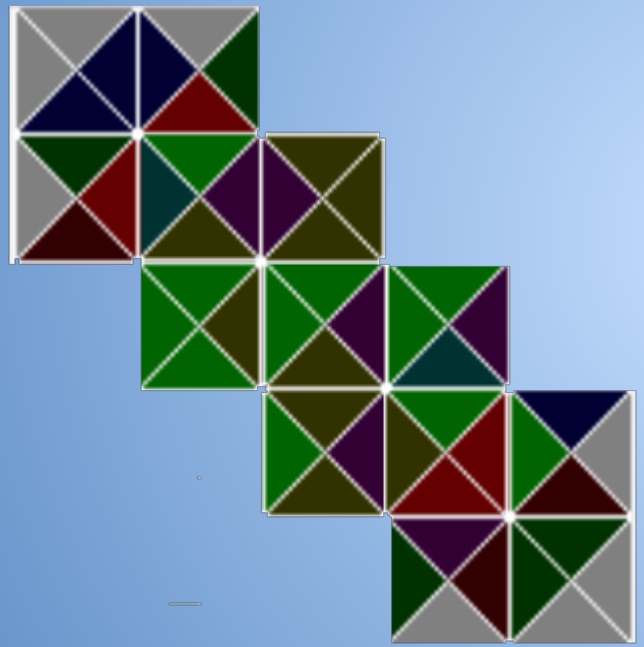
Se utilizan tres operadores de mutación. La elección del operador utilizado se determina en tiempo de ejecución basado en una probabilidad asignada a cada uno de los mismos.

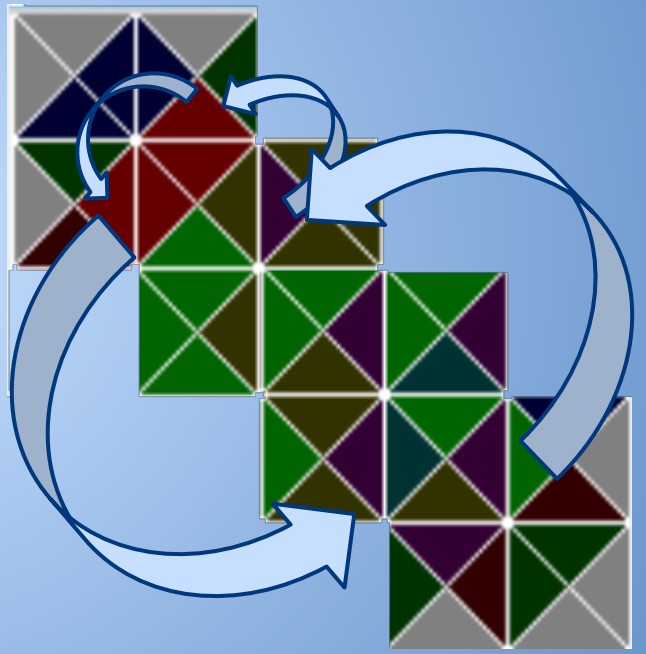
- Intercambio de Filas
- Intercambio de Columnas
- Intercambio de Piezas

Intercambio de Filas



Intercambio de Fichas





Operadores: Crossover

- Consiste en copiar una región de un tablero a otro.
- Se selecciona de forma aleatoria una región de un rompecabezas para ser copiada en el otro.
- El tamaño de la region es proporcional al tamaño del rompecabezas.
- Una vez seleccionada la región se modifica el otro rompecabezas para que la región seleccionada sea idéntica en los dos tableros.

Intercambio de región

Búsqueda local

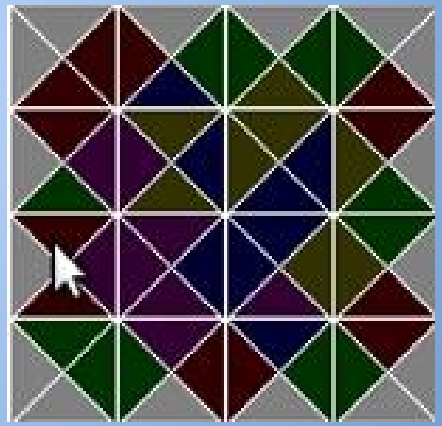
Cada cierto número de iteraciones se selecciona la mitad de los individuos (mediante Roulette Wheel Selection) para aplicarles búsqueda local.

La búsqueda local consiste en, de forma reiterada, intercambiar piezas dentro del tablero. Además las mismas se rotan para que se ajusten lo mejor posible a la nueva ubicación.

La selección de las piezas a intercambiar se hace mediante una matriz de peso. La pieza que menos se ajuste a su ubicación actual tiene más probabilidad de ser elegida.

$$w_{ij} = (sumarAdyacentesNoCoincidentes(x_{ij}) + 1)^C$$

$$\mathbf{W} = \begin{pmatrix} (0+1)^C & (1+1)^C & (0+1)^C & (0+1)^C \\ (1+1)^C & (3+1)^C & (2+1)^C & (1+1)^C \\ (2+1)^C & (1+1)^C & (1+1)^C & (0+1)^C \\ (1+1)^C & (0+1)^C & (1+1)^C & (0+1)^C \end{pmatrix}$$



Reemplazo

Una vez seleccionados los individuos y realizadas las operaciones sobre los mismos, se prosigue con la formación de la nueva generacion poblacional.
Para esto es necesario reemplazar algunos individuos de la población anterior con los recientemente obtenidos.

El reemplazo se realiza de forma elitista, es decir, se seleccionan de la población anterior los individuos que peores resultados tienen en cuanto a la función de evaluación y se los reemplaza por los nuevos, permitiendo de esta manera la convivencia de padres e hijos dentro de la población.

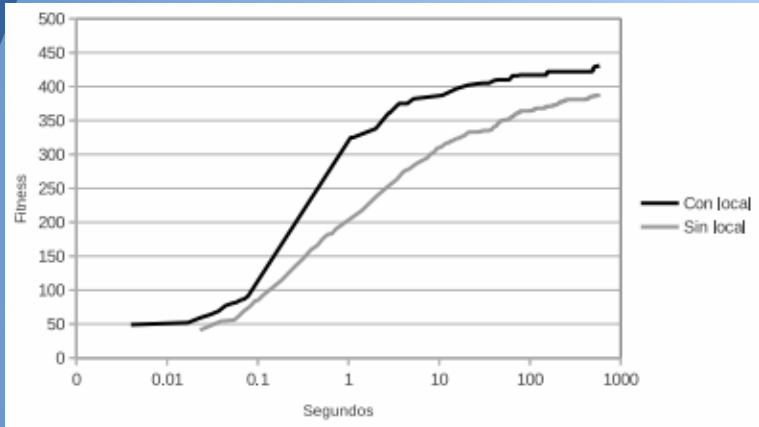
Parámentros

- Tamaño de la población: 15 individuos.
- Control de parada del algoritmo: por número de iteraciones.
- Probabilidad de mutación: 15 %.
  - Probabilidad de RowExchangeMutationOperator: 10 %.
  - Probabilidad de ColumnExchangeMutationOperator: 10 %.
  - Probabilidad de SwapAndRotateMutationOperator: 80 %.
- Tamaño de bloque de RegionExchangeCrossoverOperator: [n/2]x[n/2]
- Ejecucion de búsqueda local: cada 150 iteraciones
- Individuos a realizar búsqueda local por ejecución: 50% de la población.
- Iteraciones por cada búsqueda local: 2000.
- Threads por ejecucion de búsqueda local: cantidad de nucleos del CPU.
- Elitismo: 20% de la población.

Demo

Resultados

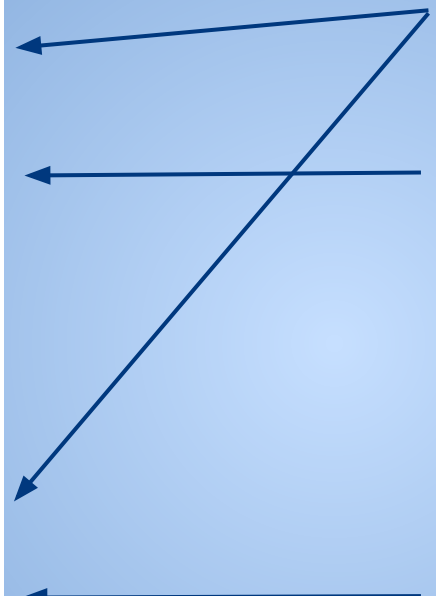
- Los casos de prueba utilizados para correr los experimentos fueron obtenidos de dos fuentes:
  - Sitio oficial de la “International Conference on Metaheuristics and Nature Inspired Computing (META)", edición 2010.
  - Red de intercambio de archivos peer-to-peer y luego validados con el checksum que ofrece E2Lab.
- Las pruebas fueron realizadas en una computadora con dos núcleos de 2.20Ghz cada uno y 4GB de RAM.



(b) Escala logarítmica

- Crecimiento muy marcado hasta los primeros 10 segundos.
- Se estabiliza aproximadamente luego de los 100 segundos.
- Algoritmo con búsqueda local consigue en todo momento mejores soluciones.
- Algoritmo con búsqueda local muestra incremento en forma de saltos.

Iteracion	Fitness	Similitud
9400	410	1
9500	413	0.4
9600	413	1
9700	413	0.87
9800	413	0.98
9900	413	1
10000	415	0.81
10100	415	1
10200	415	1
⋮	⋮	⋮
24900	418	1
25000	418	1
25100	419	0.17
25200	419	1
25300	420	0.93
25400	420	1
25500	420	1
25600	420	1



Salto con soluciones muy distintas.

Mismo fitness, distintas soluciones.

Salto con soluciones similares.

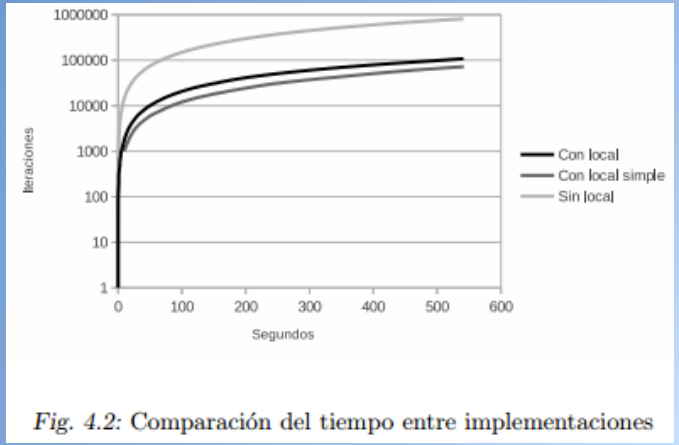


Fig. 4.2: Comparación del tiempo entre implementaciones

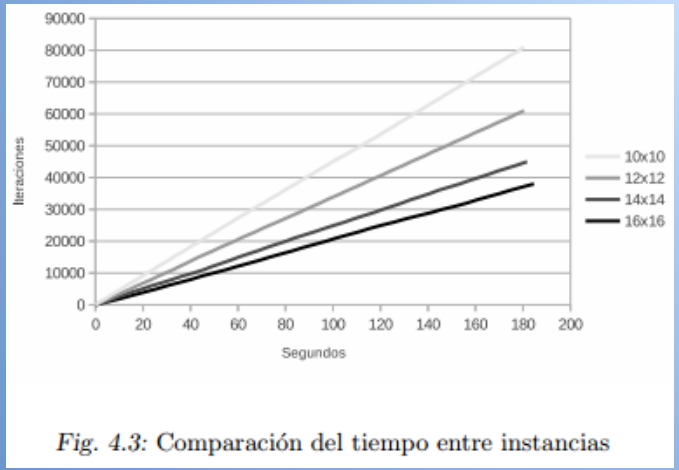


Fig. 4.3: Comparación del tiempo entre instancias

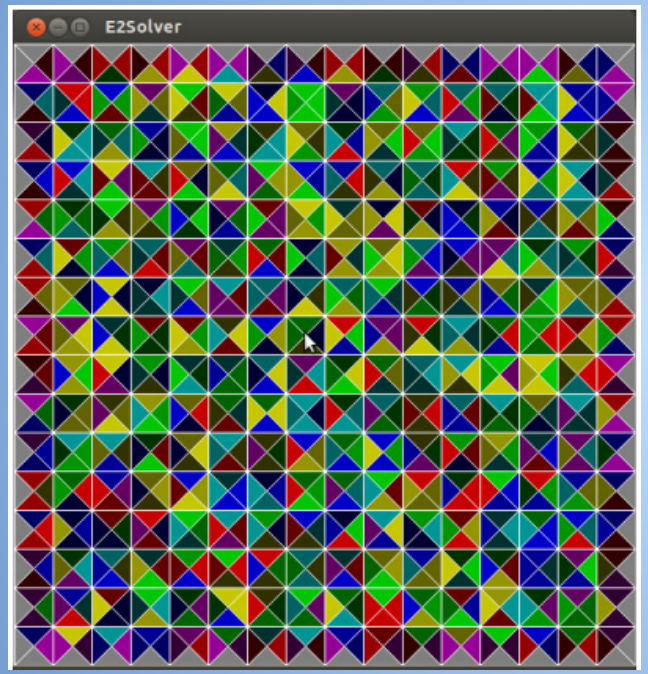
- Gran impacto en el rendimiento con bùsqueda local.
- La utilización de threads (2) mejora el rendimiento.
- Es posible mejorar el rendimiento aumentando la cantidad de threads sin necesidad de recurrir a una implementación distribuida.
- Comparando entre distintas instancias el rendimiento es el esperado.

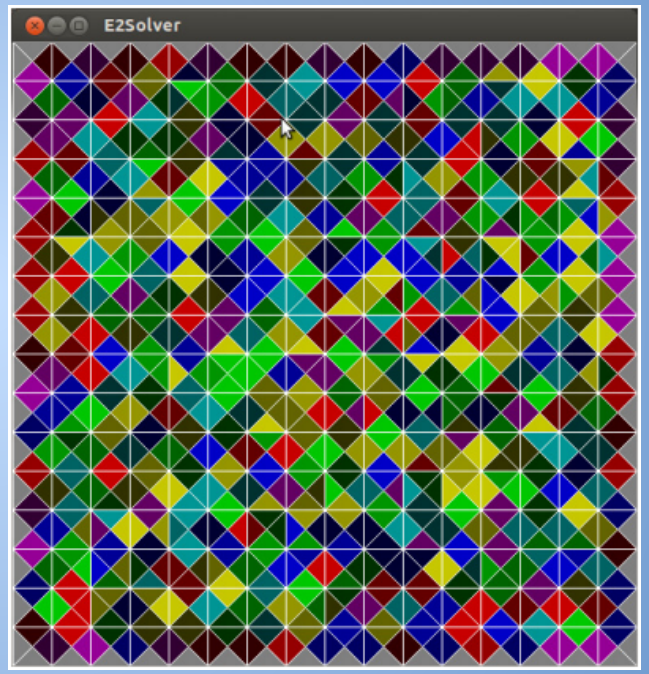
Resultados Finales

Instancia	Max Sol Posible	Iteraciones	Mejor Sol	Duracion (seg)	ES	VS
6x6	60	57041	60	42.7	59.8	0.40
7x7	84	128000	81	183.2	80.6	0.27
10x10	180	128000	165	281.9	162.6	2.49
12x12	264	128000	239	378.1	234.8	6.18
14x14	364	128000	321	531.0	318.2	5.07
16x16 [1]	480	128000	431	626.7	426.3	8.23

Tab. 4.2: Resultados finales en las diferentes instancias

Fig. 4.4: Mejor solución en instancia 6×6 (60/60)


E2Solver


E2Solver

Comparación con Trabajos Relacionados

La instancia utilizada para realizar la comparación fue la correspondiente al Eternity II.

Mejor Solución	Promedio	Técnica	Autores
396	392.5	GA[1]	Muñoz y col.
418	408.6	Tabu Search y SA[1]	Wei-Sin y col.
425	419.6	VNS[1]	Coelho y col.
431	**426.3**	**GA[1] y Búsqueda Local**	**Tesis**
458	≈ 440	Contraint Programming y VLNS[1]	Schaus y col.
459	456.4	Hiper Heurísticas	Vancroonenburg y col.

Tab. 4.4: Comparación con trabajos relacionados

Conclusiones

- El agregado de la búsqueda local mejora la calidad de las soluciones.
- La combinación de ambas técnicas parece complementarse de manera satisfactoria.
- El rendimiento del algoritmo se ve notablemente afectado al combinarlo. De todas formas el trade-off es favorable.
- Mejores resultados que otros autores o resultados similares en menor tiempo.
- Parece difícil disminuir el margen con las mejores soluciones.

Trabajos Futuros

- Operadores con conocimiento del dominio del problema. Por ejemplo, tener en cuenta bloques completos.
- Función de evaluación adaptativa.
- Algoritmo completo adaptativo.

¡Gracias!

Foto Actual?