Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Optimización de corte mediante algoritmos genéticos y enfriamiento simulado

Autor: Yuri Vladimir Huallpa Vargas

Asignatura: Técnicas de inteligencia artificial

2022/2023

I. Introducción

Hoy en día el mundo de la industria es una de las actividades económicas mas contaminantes tanto durante el proceso productivo o por los residuos que generan. De hecho, es la responsable de los problemas actuales como el calentamiento global, contaminación del suelo, contaminación del ecosistema acuático y más.

Muchos productos del sector industrial tienden a ser laminas muy grandes, que generalmente estos deben ser cortados o troceados para poder ser distribuidos y darles un nuevo uso, así como los papeles, las planchas metálicas, maderas, melaminas, y más. Realizar dicho proceso conlleva generar residuo y gastos adicionales sobre el precio inicial. Por lo tanto, encontrar alguna técnica que optimice o solucione dicho problema significaría menos pérdida económica y menor contaminación.

II. Planteamiento del problema

Encontrar un algoritmo que dado un conjunto de patrones de corte encuentre la forma de poder realizar dichos cortes e intentar minimizar el porcentaje de material desperdiciado.

A. Objetivos.

- Encontrar el rectángulo con el área más pequeña que contenga todos los patrones de corte o que abarque el mayor número posible, dicho proceso significa realizar un corte inmediatamente después de la anterior sin producir ningún espacio entre ellos. Esto se puede expresar de la siguiente forma:

$$\min_{(x_1,x_2,\dots,x_n),(y_1,y_2,\dots,y_n)} f\,(p_1, p_2, \dots, p_n) = 1 - \frac{\sum_{i=1}^{n} w_i * h_i}{W * H}$$

Donde $p_1, p_2, p_3, \dots, p_n$, son los patrones de corte a realizar, w_i, h_i son el ancho y largo de los patrones a cortar y W, H son las dimensiones del rectángulo mínimo que recubre a todos los patrones de corte.

B. Restricciones.

- Las dimensiones del rectángulo mínimo que recubre los patrones de corte no deben superar las dimensiones del material principal, el cual esta definido mediante la siguiente condición.

$$W \leq W_t \wedge H \leq H_t$$

Donde $W\ y\ H$ es el ancho y largo del material principal.

- Los cortes realizados deben ser cortes de guillotina. Un corte en guillotina “Guillotine Cutting” consiste en producir rectángulos más pequeños a partir de uno más grande; el corte inicia en un lado del material y terminar en extremo opuesto siguiendo un patrón lineal, y un corte por guillotina genera dos áreas o subespacios.

III. Solución del problema

Para resolver el problema se ha estudiado dos metaheurísticas de optimización, uno inspirado en la evolución biológica (algoritmos genéticos) y otro basado en el recalentamiento del metal y la cerámica (enfriamiento simulado).

A. Algoritmos genéticos.

Es una técnica de optimización que imita la evolución biológica como estrategia para encontrar soluciones, se sigue una serie de pasos (selección, mutación, cruce y remplazo); no necesariamente todos.

Primero se inicializan con un conjunto de soluciones posibles o completamente aleatorios el cual son denominados cromosomas, los cual deben ser seleccionados según una métrica de evaluación llamada aptitud o fitness el cual indica el grado de adaptabilidad del cromosoma en su entorno; la codificación del cromosoma es una parte crucial y no trivial ya que de ella depende en gran medida el éxito o el fracaso del algoritmo. Posteriormente mediante una serie de procesos aleatorios de combinación o remplazos se logra conseguir los individuos de la siguiente generación el cual deben de ser evaluados antes de ser aceptados, en caso no sean aptos serán eliminados y otros serán generados aleatoriamente. Este proceso se sigue iterativamente hasta conseguir una solución suficientemente buena, no necesariamente la mejor. Una descripción grafica de ello se puede observar en la Ilustración 1.

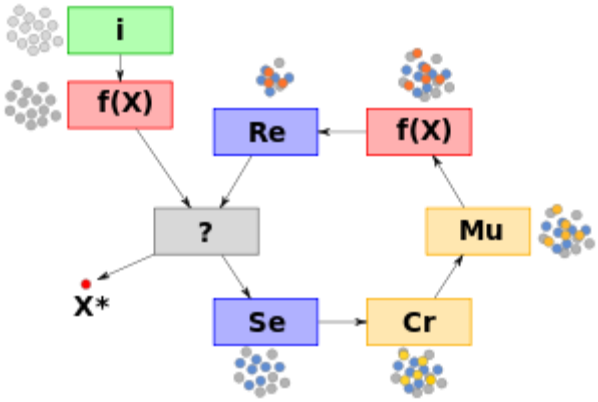


Ilustración 1. Grafo de estados correspondientes a un algoritmo genético, donde i: inicialización, f(X): evaluación, ?: condición de término, Se: selección, Cr: cruzamiento, Mu: mutación, Re: reemplazo, X: mejor solución.*
Fuente: https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico.

1. Diseño del algoritmo.

Para el diseño del algoritmo se debe de considerar una forma de inicializar una población o individuos, haber codificado el problema (genotipo), encontrar una manera de decodificar la posible solución (fenotipo) y un conjunto de parámetros

que nos permitan tener más flexibilidad a la hora de buscar una solución y evitar caer en óptimos locales.

Generalmente encontrar una codificación adecuada para el problema es la parte más difícil ya sea computacionalmente o temporalmente, estas suelen ser una secuencia de caracteres denominados “Gen” y que en conjunto son denominados cromosomas.

a) Población.

Al conjunto de cromosomas o individuos en una generación determinada se le conoce como población, generalmente se inicializa de forma aleatoria, no existe una regla que indique cuantos deben ser generados. Mientras más individuos más oportunidades tendrá el algoritmo de encontrar una solución suficientemente buena, pero esto a cambio de más poder de cómputo.

Los N cromosomas de una población deben de ser decodificables, el cromosoma debe tener su correspondiente fenotipo (solución) y no ser ambiguo.

Por naturaleza de algunos problemas a veces es necesario tener dos tipos de cromosomas que codifiquen la solución, como es el caso de la optimización de corte, donde se requiere un cromosoma que indiquen el patrón de corte seguido y otros que indiquen cual es la orientación de cada pieza antes de ser cortada.

b) Codificación.

Optimizar un corte no suele ser una tarea trivial; puesto que, se necesita tener en cuenta muchas variables como, la orientación de cada pieza, las dimensiones del material, un patrón que pueda ser seguido por una maquina industrial, etc.

El patrón de corte a realizar debe cumplir con las restricciones definidas en II.B. Por consiguiente, los corte por guillotina se puede representar de dos maneras distintas; horizontal y vertical, ver Ilustración 2. Donde cada línea roja indica la dirección del corte realizado.

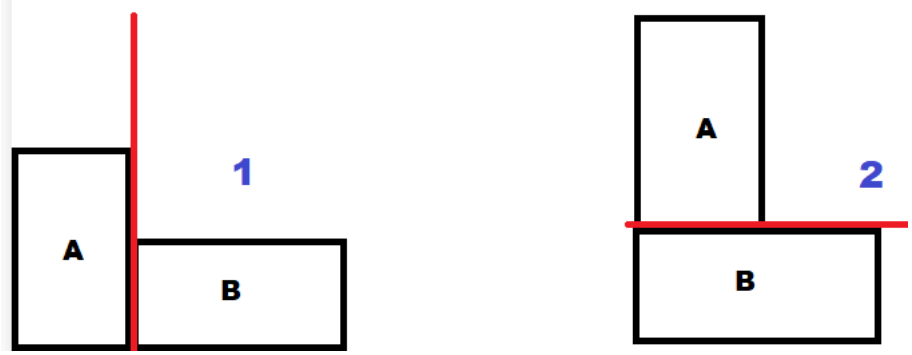


Ilustración 2. Patrones de corte generados mediante la técnica corte por guillotina; la imagen del lado izquierdo representa un patrón horizontal (H) y el lado izquierdo un patrón vertical (V).

Los patrones generados y la orientación individual de cada pieza son almacenados dentro de una estructura de datos llamado árbol binario, el cual almacena una representación exacta y sin ambigüedad del corte realizado. Por ejemplo, de la Ilustración 2 se puede generar los siguientes árboles. Primero, para el patrón

horizontal la raíz del árbol es el operador “H” (horizontal), su correspondiente hijo izquierdo es el rectángulo “A” con una orientación vertical y el hijo derecho es el rectángulo “B” con una orientación horizontal. Segundo, para el patrón vertical la raíz es el operador “V” (vertical), su hijo izquierdo es el rectángulo “B” con una orientación horizontal y su hijo derecho es el rectángulo “A” con una orientación vertical. Finalmente, solo es necesario realizar un recorrido en post orden para conseguir los cromosomas correspondientes: *“ABH” y “VH” para el primer árbol y “BAV” y “HV” para el segundo árbol; esta expresión también es conocida como notación polaca inversa.*

Una ilustración más compleja se puede visualizar en la Ilustración 3, donde al lado izquierdo se muestra el patrón de corte, al lado derecho el árbol binario correspondiente y únicamente es necesario realizar un recorrido en post orden para obtener los cromosomas de corte *“12V4V35V67VHH”* y de orientación *“VVVHHVH”*.

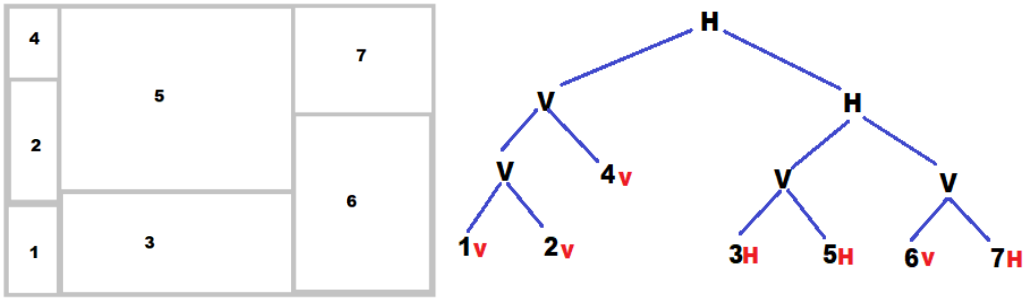


Ilustración 3.Corte de guillotina representado mediante un árbol binario.

c) Decodificación.

Decodificar el cromosoma a su respectivo fenotipo o solución suele ser relativamente sencillo por estar en notación post fija, únicamente se debe recorrer el cromosoma de corte de izquierda a derecha e ir operando mediante una pila según indique el gen en la que se encuentre, si el gen corresponde a una pieza, se asigna su orientación extrayendo la primera posición del cromosoma de orientación (cola) y se apila, y si corresponde a un gen operador, se extrae las dos últimas piezas de la pila y se opera con ella, la solución resultante nuevamente es apilada. Se realiza este proceso hasta consumir el cromosoma de corte y la solución resultante viene a ser un rectángulo que contiene todas las piezas en la menor área posible.

El proceso descrito se puede observar en la siguiente imagen.

Cromosoma de corte: Se denominará a la expresión generada a partir del árbol que almacena las orientaciones relativas de cada pieza dado un patrón de corte.
Cromosoma de posición: Se denominará al cromosoma formado por la orientación absolutas de cada pieza.

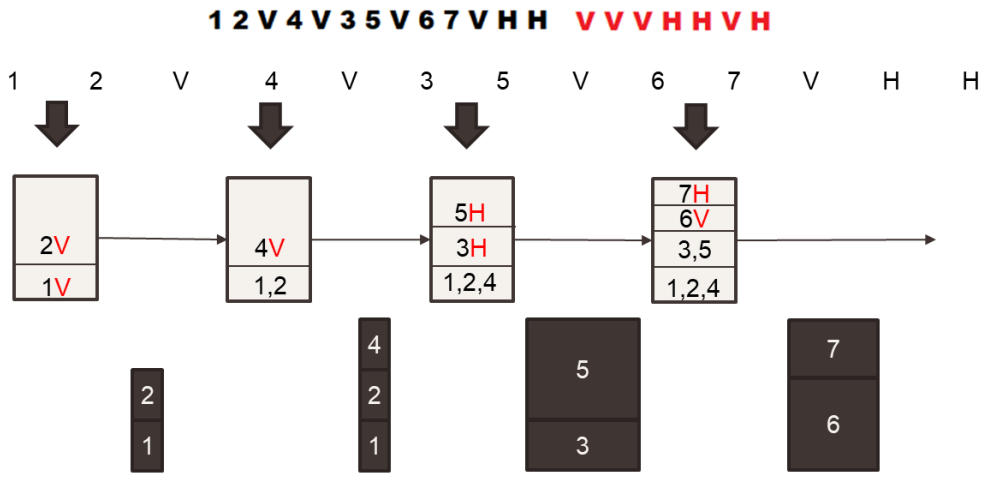


Ilustración 4. Decodificación del cromosoma de corte y orientación.

d) Fitness.

El fitness o aptitud de un cromosoma indica el grado de adaptabilidad de un cromosoma a su entorno. La función de aptitud esta definido mediante la siguiente ecuación.

$$fitness(C) = \frac{\sum_{i=1}^{n} w_i * h_i}{W * H}$$

Donde w_i, h_i es el ancho y largo de cada pieza, W, H es el ancho y largo del rectángulo que cubre todas las piezas. El cromosoma mas optimo es aquel cuyo fitness es 1.

e) Selección.

Los individuos más óptimos (padres) son elegidos según su correspondiente función de aptitud (selección elitista).

Optar por un gran número de padres puede provocar que el algoritmo tenga una convergencia prematura.

f) Cruce.

Es el proceso de intercambiar genes entre dos progenitores para formar nuevos descendientes, realizar muchos intercambios (mayor mutación) permite que el algoritmo realice una exploración más amplia de posibles soluciones, caso contrario el algoritmo intenta mejorar una solución posible.

El cruce de individuos se realiza mediante la técnica rueda de la ruleta en función de la aptitud del cromosoma, aquellos cromosomas con mejor aptitud tendrán mayor probabilidad de realizar un cruce. La probabilidad se define mediante la siguiente ecuación.

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Donde f_i es el fitness del cromosoma.

Por naturaleza del problema únicamente se considera el cromosoma de corte para generar los descendientes, además por ser cromosomas de genes compuestos es necesario primero dividirlos en dos subconjuntos; subconjunto de piezas y subconjunto de operadores. El cruce se realiza en el subconjunto de operadores, únicamente se intercambian aquellos operadores que se encuentran dentro del segmento generado aleatoriamente y se recombinan para generar los nuevos descendientes.

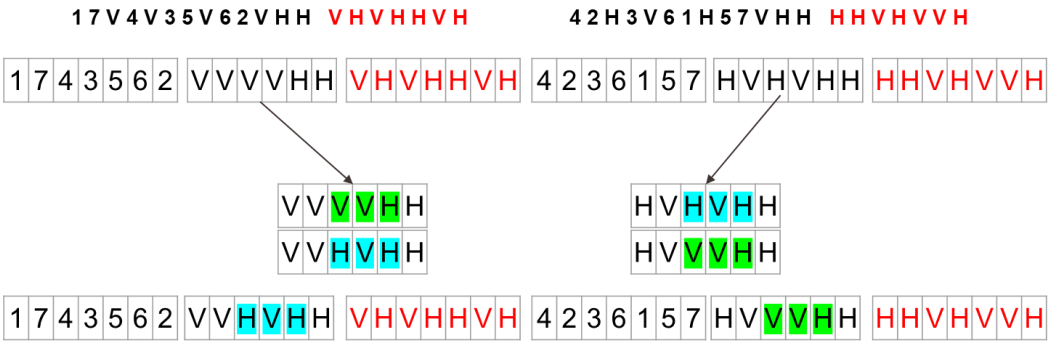


Ilustración 5. Generación de nuevos individuos.

g) Mutación.

La mutación nos permite introducir innovación en la búsqueda, permite explorar zonas de búsqueda distantes o locales y diversificar los individuos de una población. Se ha determinado las siguientes mutaciones en función de la diversidad de genes, por cambio de orientación, por cambio de operador y por intercambio de piezas.

La mutación se realiza si se satisface una condición de probabilidad p, además un cromosoma solo puede realizar una mutación según un numero aleatorio con una distribución uniforme.

I. Mutación por cambio de orientación:
   Se toma el cromosoma de orientación y se realiza un cambio bit-a-bit en función de un numero aleatorio y una condición de probabilidad.

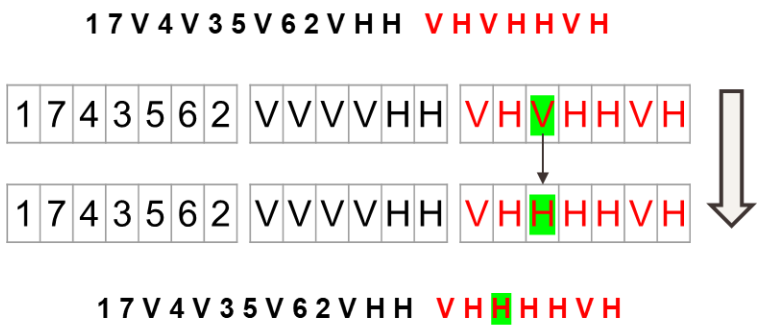


Ilustración 6. Mutación por cambio de orientación.

II. Mutación por cambio de operador:

Se toma el cromosoma de corte y se dividen en dos subconjuntos; subconjunto de piezas y subconjunto de operadores, se realiza un cambio bit-a-bit en función de un numero aleatorio y dos condiciones de probabilidades dentro del subconjunto de operadores.

Aquellos operadores cuyos vecino izquierdo y derecho es otro gen operador debe tener una mutación menos aleatoria; puesto que, este tipo de agrupación hace referencia a dos subconjuntos con muchos bloques relativamente bastante ordenados.

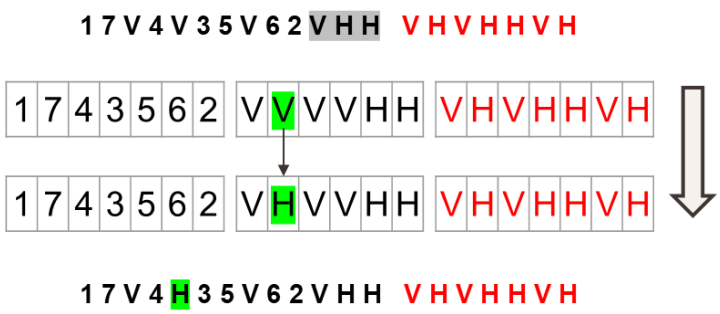


Ilustración 7. Mutación por cambio de operador.

III. Mutación por intercambio de piezas:

Se toma el cromosoma de corte y se dividen en dos subconjuntos; subconjunto de piezas y subconjunto de operadores, se realiza un intercambio reciproco en función de dos posiciones aleatorios y una condición de probabilidad dentro del subconjunto de piezas.

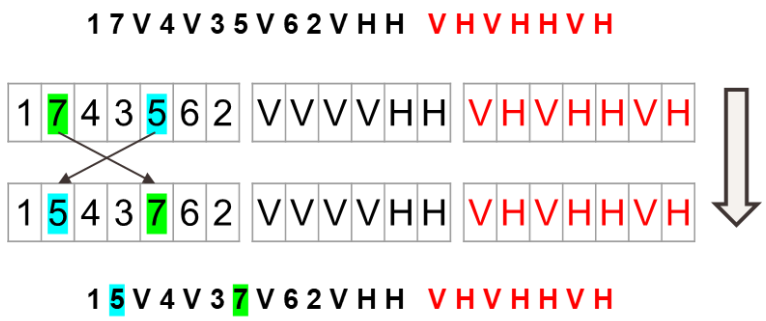


Ilustración 8. Mutación por intercambio de piezas.

h) Remplazo.

Consiste en sustituir una población inicial por los descendientes excepto aquellos N cromosomas elites. Los individuos que no cumplen con las restricciones descritas en II.B, son eliminados y remplazados por nuevos individuos generados aleatoriamente.

B. Enfriamiento simulado.

Algoritmo metaheurístico inspirado en el proceso de recocido del acero, proceso que consiste en variar la temperatura y luego enfriarla para variar las propiedades físicas del material. La temperatura hace que los átomos incrementen su energía causando que estos se reagrupen de mejor manera en un nuevo estado de menor energía.

1. Diseño del algoritmo.

El algoritmo de enfriamiento simulado inicia con una temperatura bastante alta, que va decrementando según las iteraciones realizadas; en cada iteración se evalúa un conjunto de transiciones posibles (vecinos) a partir de un estado S y mediante una probabilidad se decide entre realizar el cambio o permanecer en el mismo estado S. Mientras mayor es la temperatura mayor es la probabilidad de cambiar de un estado a otro sin considerar si esta es un estado de mayor energía (mala solución), a medida que la temperatura decrementa únicamente se aceptaran estados de menor energía.

Los vecinos están compuestos por todos los estados que se puede llegar a partir de S mediante algunos cambios realizados.

a) Temperatura.

La temperatura juega un rol muy importante en la exploración de posibles soluciones, mientras mayor sea la temperatura más amplia será el espacio de búsqueda, pero más tiempo tomara el algoritmo en encontrar una solución.

Según lo mencionado, la temperatura viene a ser un parámetro configurable y se define mediante la siguiente ecuación:

$$T(t) = \frac{t}{1 + kt}, \quad 0 < k < 1$$

Donde t varía en función de una constante k.

b) Energía o aptitud.

La energía de un vecino indica la calidad de la solución encontrada y se define mediante la siguiente ecuación.

$$energia(C) = \frac{\sum_{i=1}^{n} w_i * h_i}{W * H}$$

c) Estado inicial.

El estado inicial es el vecino con menor energía seleccionado de un conjunto de N vecinos generados aleatoriamente, el estado inicial debe cumplir las restricciones descritas en II.B. y cada vecino se codifica tal como se describe en III.A.1.b).

Por cuestiones técnicas a los cromosomas de corte se le denominara como vecinos y al cromosoma de orientación se le denominara simplemente como orientación.

d) Vecinos.

Los vecinos son estados generados a partir del estado inicial S y su correspondiente orientación S_o, existe 3 tipos de transformación mediante el cual se generan nuevos estados el cual se detalla a continuación.

I. Transformación por variación de la orientación:
Dado un estado inicial S y su correspondiente estado de orientación S_o, se toma S_o, se recorre bit-a-bit de izquierda a derecha, si se cumple una condición de probabilidad entonces el bit se cambia y se genera un nuevo estado S con su correspondiente estado de orientación S_o'.

S := vecino
S_o := orientación(S)
vecinos := {}
para i desde 0 hasta longitud(S_o):
  si se cumple la probabilidad p entonces:
    $S_o^{'}$:= Cambiar(S_o, i)
    Agregar(vecinos, S, $S_o^{'}$)

II. Transformación por variación del operador:
Dado un estado inicial S y su correspondiente estado de orientación S_o, se toma S y se divide en dos subconjuntos; subconjunto de piezas S_p y subconjunto de operadores S_{op}, se recorre bit-a-bit de izquierda a derecha en el subconjunto de operadores, si se cumple una condición de

probabilidad entonces el bit se cambia y se genera un nuevo estado S' y su correspondiente estado de orientación S_o.

S := vecino
S_p, S_{op} := Dividir(S)
S_o := Orientación(S)
vecinos := {}
para i desde 0 hasta longitud(S_{op}):
  si se cumple la probabilidad p entonces:
    $S_{op}^{'}$:= Cambiar(S_{op}, i)
    $S^{'}$:= Unir($S_p, S_{op}^{'}$)
    Agregar(vecinos, $S^{'}$, S_o)

III. Transformación por intercambio de piezas:
Dado un estado inicial S y su correspondiente estado de orientación S_o, se toma S y se divide en dos subconjuntos; subconjunto de piezas S_p y subconjunto de operadores S_{op}, se recorre bit-a-bit de izquierda a derecha en el subconjunto de piezas, si un bit cumple una condición de probabilidad entonces se realiza el intercambio del bit por otro bit del mismo subconjunto tomado aleatoriamente y se genera un nuevo estado S' y su correspondiente estado de orientación S_o.

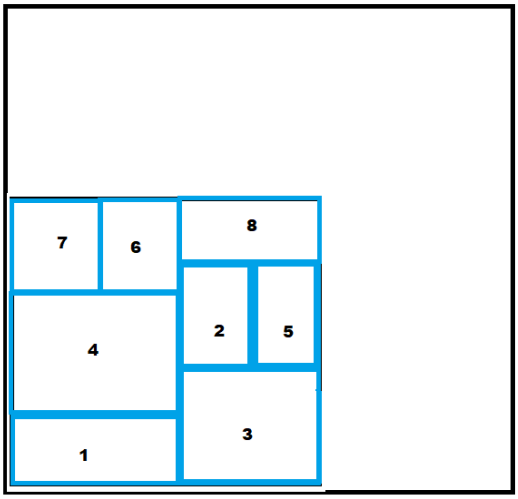
S := vecino
S_p, S_{op} := Dividir(S)
S_o := Orientación(S)
vecinos := {}
para i desde 0 hasta longitud(S_p):
  si se cumple la probabilidad p entonces:
    j := Generar_indice(longitud(S_p))
    $S_p^{'}$:= Intercambiar(S_p, i, j)
    $S^{'}$:= Unir($S_p^{'}, S_{op}$)
    Agregar(vecinos, $S^{'}$, S_o)

IV. Evaluación

Para la evaluación de los algoritmos desarrollados se han generado un set de pruebas generados aleatoriamente, en cada prueba se irán variando los parámetros y se irán incrementando el número de piezas.

Primero, se mostrará el desempeño de los algoritmos desarrollados en un set de piezas donde se conoce la solución óptima.

- Experimento 1:
  - Nro. piezas: 8
  - Piezas: [([4, 1], '1'), ([1, 2], '2'), ([4, 3], '3'), ([4, 3], '4'), ([1, 2], '5'), ([2, 2], '6'), ([2, 2], '7'), ([4, 1], '8')]



ID	Ancho	Alto
1	4	1
2	1	2
3	4	3
4	4	3
5	1	2
6	2	2
7	2	2
8	4	1

Ilustración 9. Solución óptima conocida, experimento 1.

  - Solución del algoritmo:

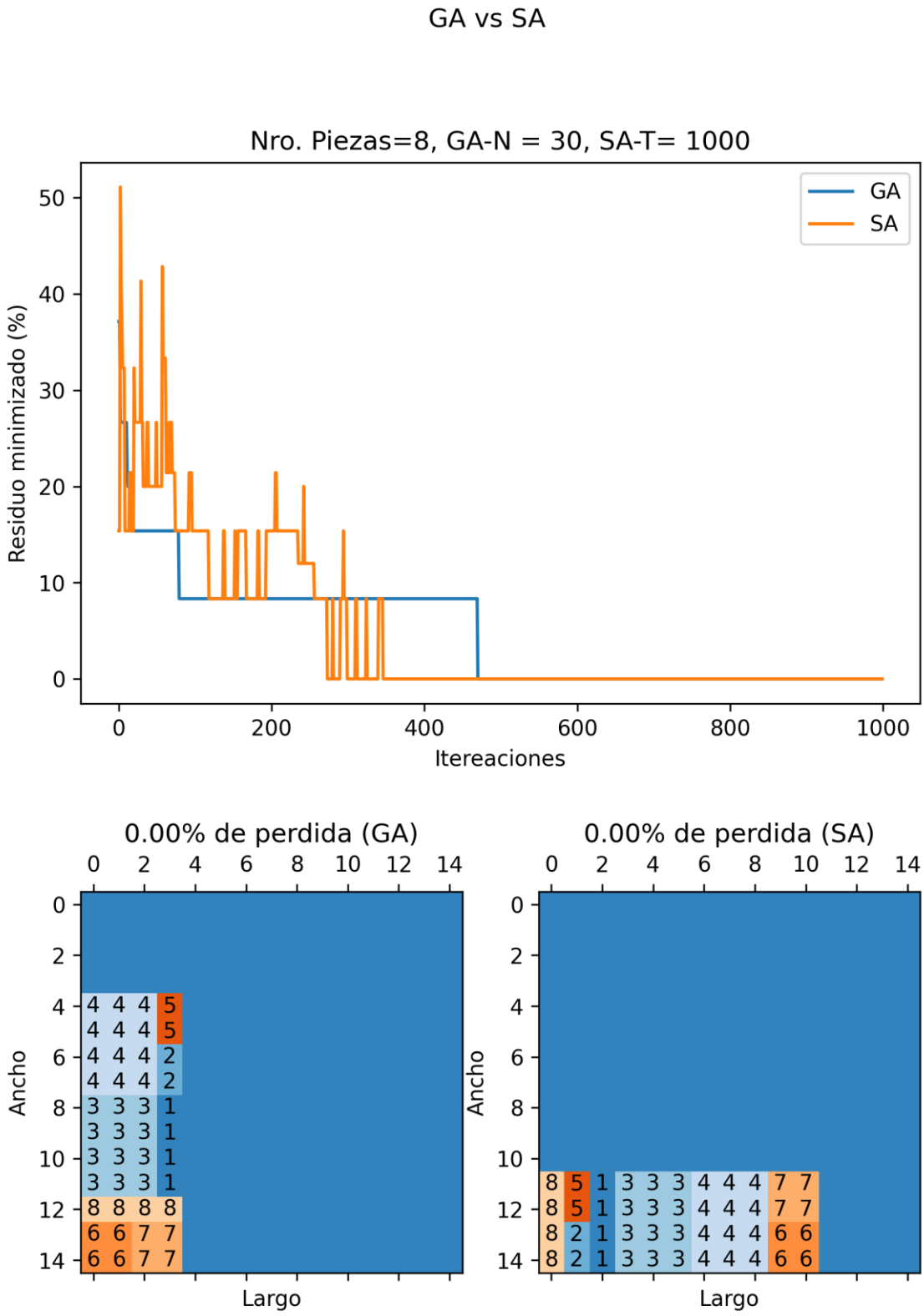


Ilustración 10. Resultados del experimento 1.

- Experimento 2:
  - Nro. piezas: 10
  - Piezas: [([4, 4], '1'), ([6, 2], '2'), ([6, 5], '3'), ([2, 1], '4'), ([3, 2], '5'), ([1, 3], '6'), ([2, 1], '7'), ([5, 2], '8'), ([3, 1], '9'), ([1, 1], '10')]
  - Solución del algoritmo:

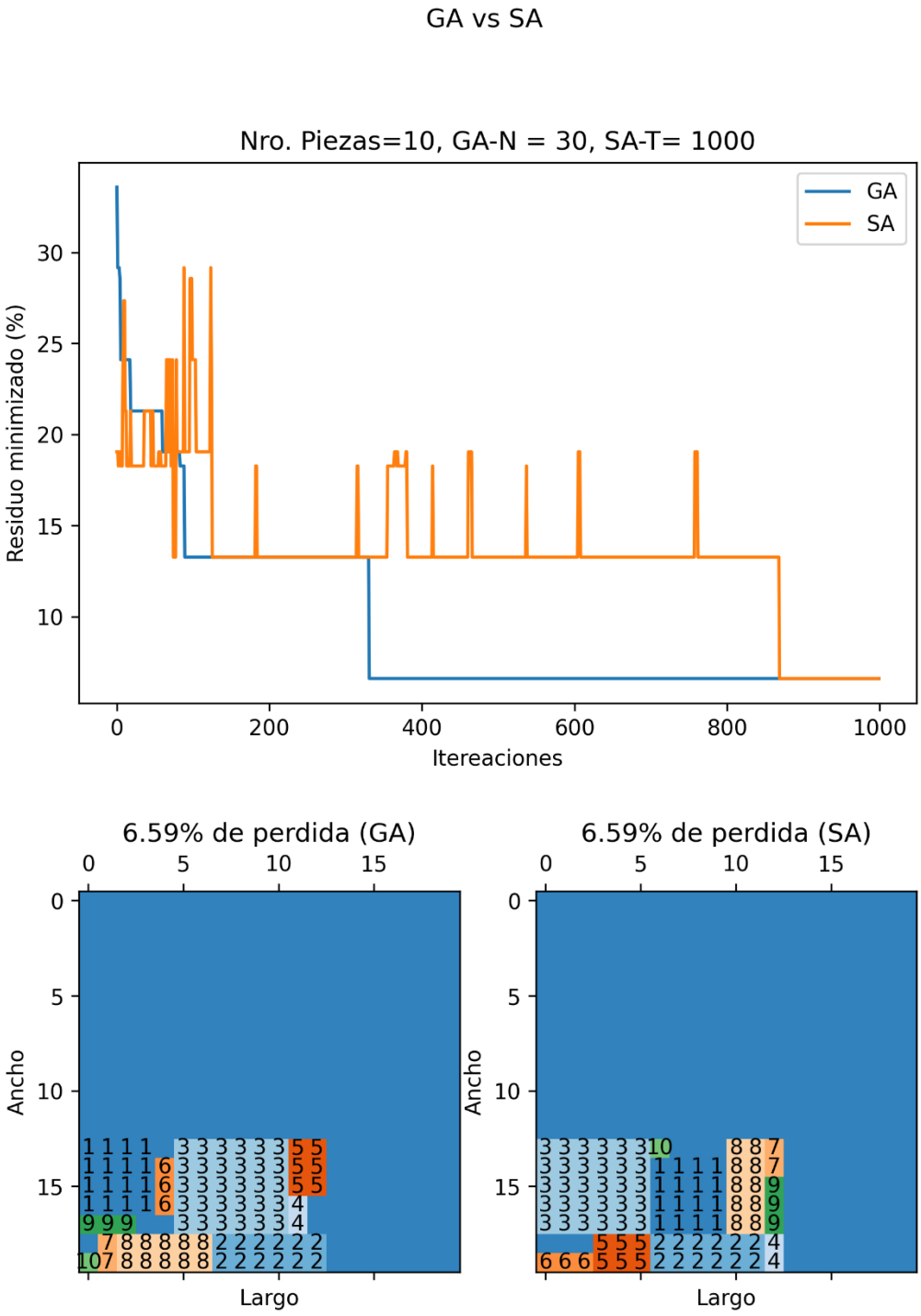


Ilustración 11. Resultados del experimento 2.

- Experimento 3:
  - Nro. Piezas: 10
  - Piezas: [([4, 4], '1'), ([6, 2], '2'), ([6, 5], '3'), ([2, 1], '4'), ([3, 2], '5'), ([1, 3], '6'), ([2, 1], '7'), ([5, 2], '8'), ([3, 1], '9'), ([1, 1], '10')]
  - Solución del algoritmo:

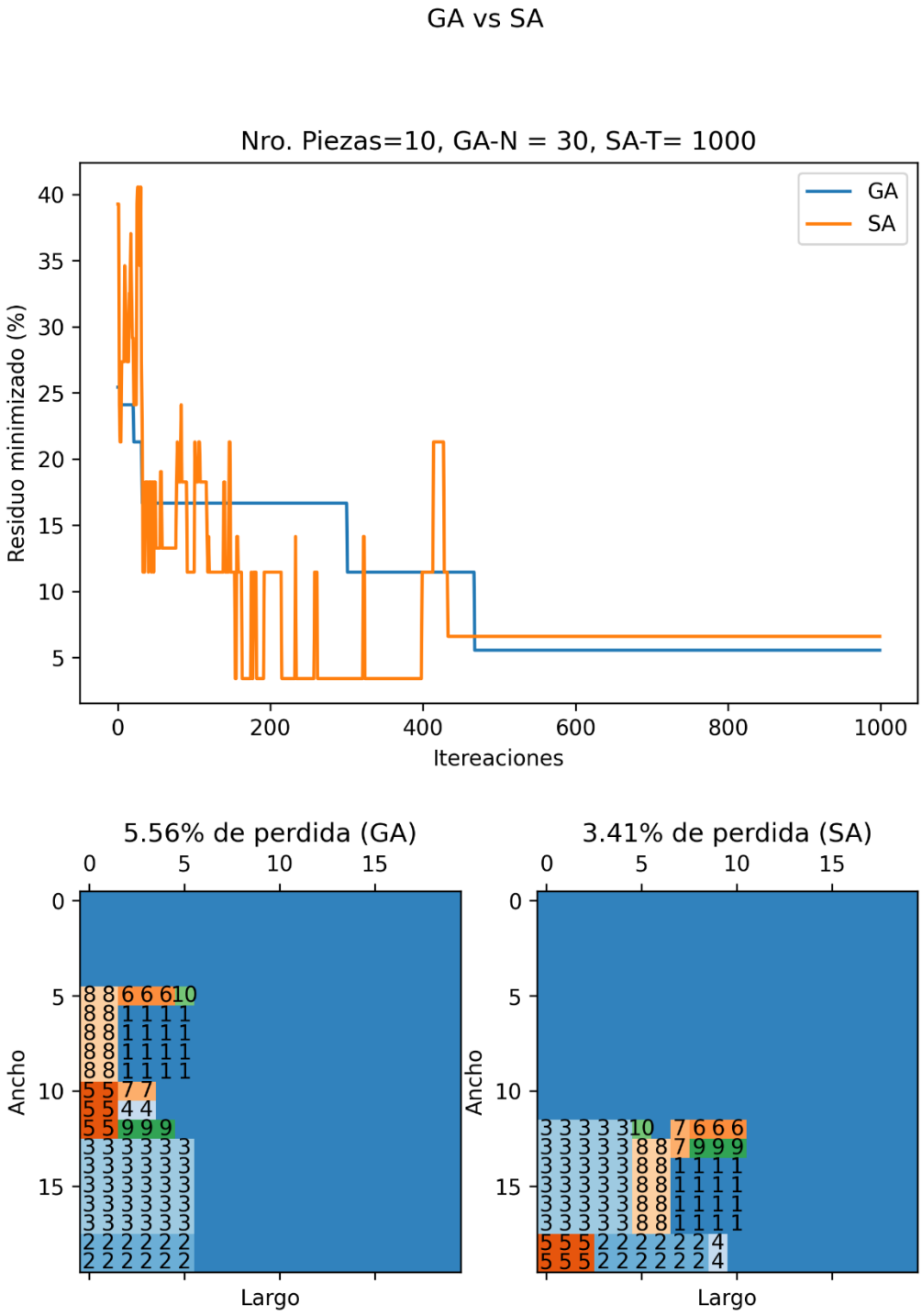


Ilustración 12. Resultados del experimento 3.

- Experimento 4:
  - Nro. Piezas: 20
  - Piezas: [([5, 10], '1'), ([10, 4], '2'), ([8, 8], '3'), ([10, 4], '4'), ([9, 3], '5'), ([2, 9], '6'), ([2, 10], '7'), ([6, 4], '8'), ([1, 7], '9'), ([10, 2], '10'), ([3, 7], '11'), ([5, 3], '12'), ([5, 8], '13'), ([1, 4], '14'), ([5, 8], '15'), ([1, 7], '16'), ([8, 5], '17'), ([10, 1], '18'), ([1, 5], '19'), ([10, 7], '20')]
  - Solución del algoritmo:

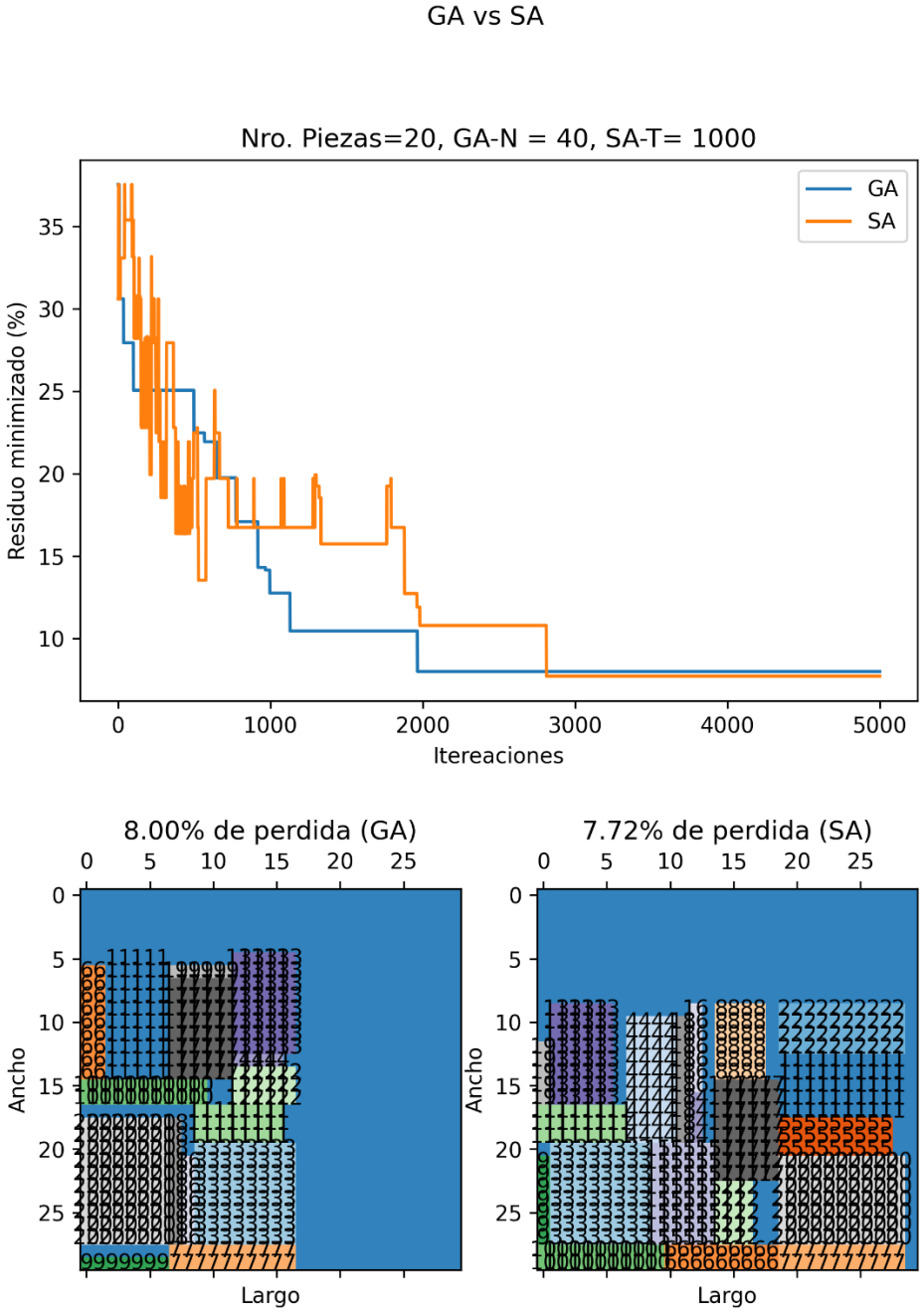


Ilustración 13. Resultados del experimento 4.

- Experimento 5:
  - Nro. Piezas: 20
  - Piezas: [([5, 10], '1'), ([10, 4], '2'), ([8, 8], '3'), ([10, 4], '4'), ([9, 3], '5'), ([2, 9], '6'), ([2, 10], '7'), ([6, 4], '8'), ([1, 7], '9'), ([10, 2], '10'), ([3, 7], '11'), ([5, 3], '12'), ([5, 8], '13'), ([1, 4], '14'), ([5, 8], '15'), ([1, 7], '16'), ([8, 5], '17'), ([10, 1], '18'), ([1, 5], '19'), ([10, 7], '20')]
  - Solución del algoritmo:

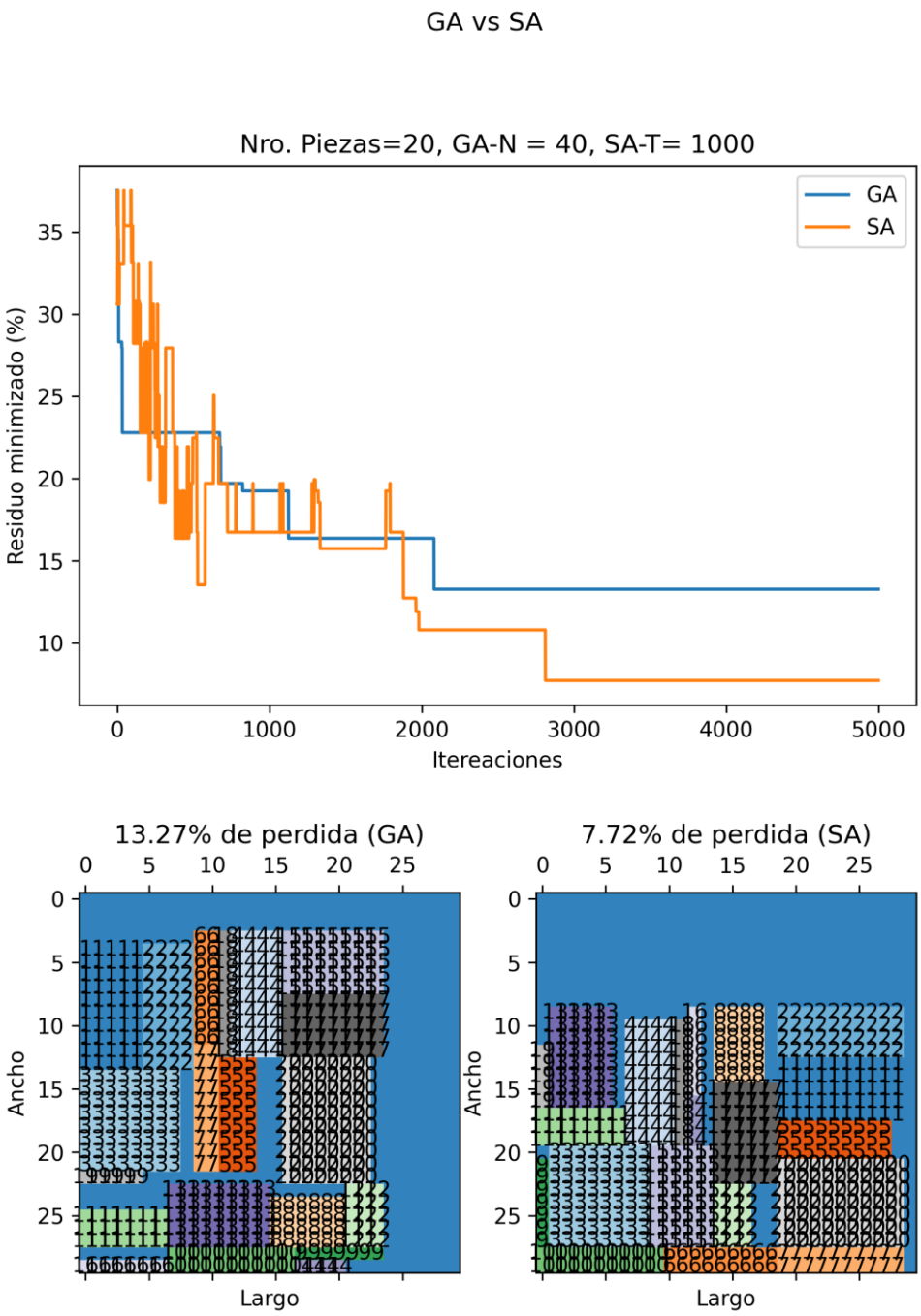


Ilustración 14. Resultados del experimento 5.

- Experimento 6:
  - Nro. Piezas: 20
  - Piezas: [([5, 10], '1'), ([10, 4], '2'), ([8, 8], '3'), ([10, 4], '4'), ([9, 3], '5'), ([2, 9], '6'), ([2, 10], '7'), ([6, 4], '8'), ([1, 7], '9'), ([10, 2], '10'), ([3, 7], '11'), ([5, 3], '12'), ([5, 8], '13'), ([1, 4], '14'), ([5, 8], '15'), ([1, 7], '16'), ([8, 5], '17'), ([10, 1], '18'), ([1, 5], '19'), ([10, 7], '20')]
  - Solución del algoritmo:

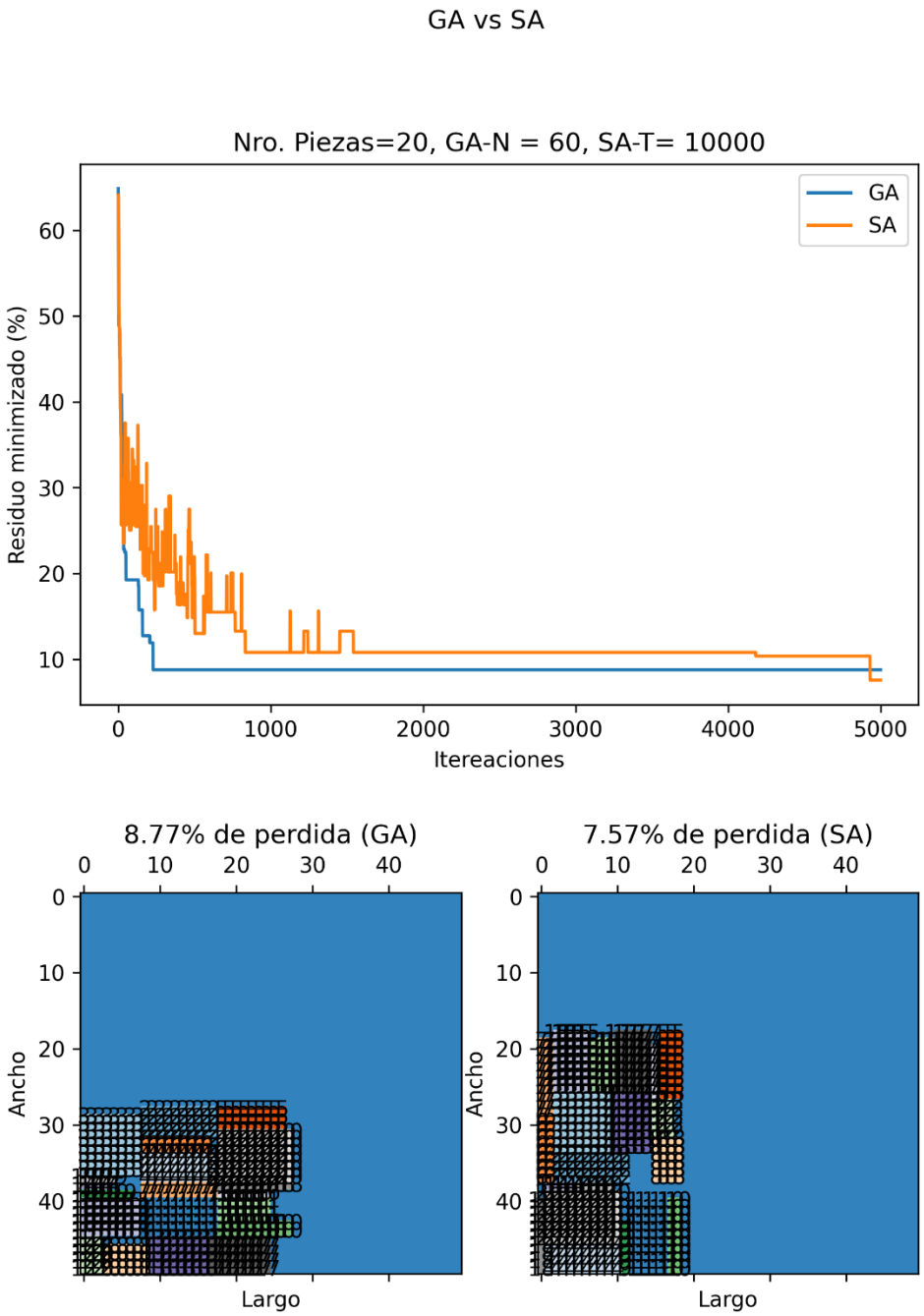


Ilustración 15. Resultados del experimento 6.

- Experimento 7:
  - Nro. Piezas: 20
  - Piezas: [([5, 10], '1'), ([10, 4], '2'), ([8, 8], '3'), ([10, 4], '4'), ([9, 3], '5'), ([2, 9], '6'), ([2, 10], '7'), ([6, 4], '8'), ([1, 7], '9'), ([10, 2], '10'), ([3, 7], '11'), ([5, 3], '12'), ([5, 8], '13'), ([1, 4], '14'), ([5, 8], '15'), ([1, 7], '16'), ([8, 5], '17'), ([10, 1], '18'), ([1, 5], '19'), ([10, 7], '20')]
  - Solución del algoritmo:

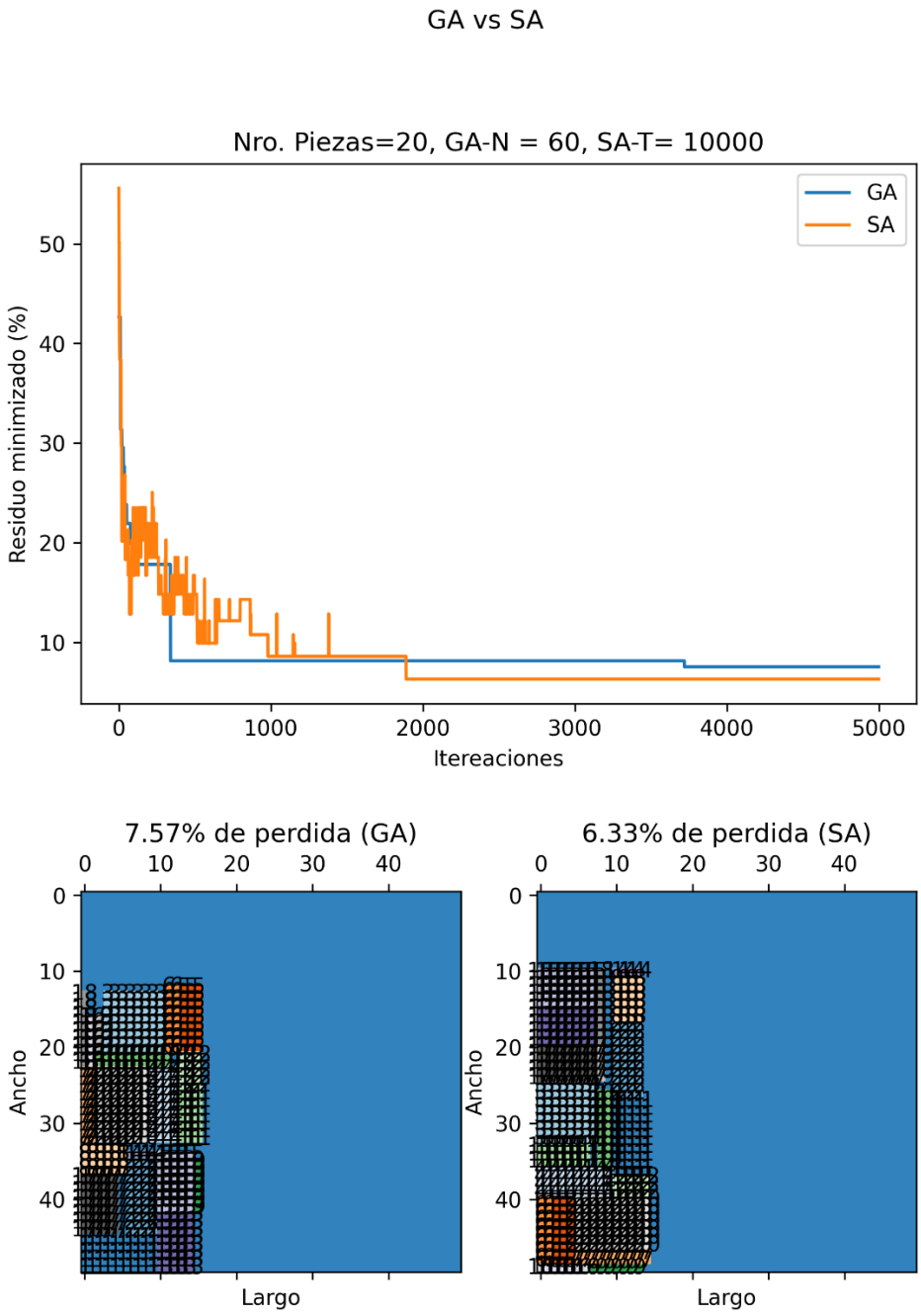


Ilustración 16. Resultados del experimento 7.

- Experimento 8:
  - Nro. Piezas: 30
  - Piezas: [([1, 4], '1'), ([5, 10], '2'), ([9, 1], '3'), ([8, 8], '4'), ([7, 8], '5'), ([4, 3], '6'), ([10, 7], '7'), ([6, 1], '8'), ([7, 4], '9'), ([1, 3], '10'), ([4, 4], '11'), ([9, 6], '12'), ([4, 3], '13'), ([9, 2], '14'), ([7, 5], '15'), ([10, 10], '16'), ([10, 8], '17'), ([8, 9], '18'), ([7, 8], '19'), ([7, 2], '20'), ([10, 5], '21'), ([4, 9], '22'), ([4, 9], '23'), ([4, 2], '24'), ([5, 8], '25'), ([5, 10], '26'), ([10, 10], '27'), ([9, 1], '28'), ([2, 10], '29'), ([3, 3], '30')]
  - Solución del algoritmo:

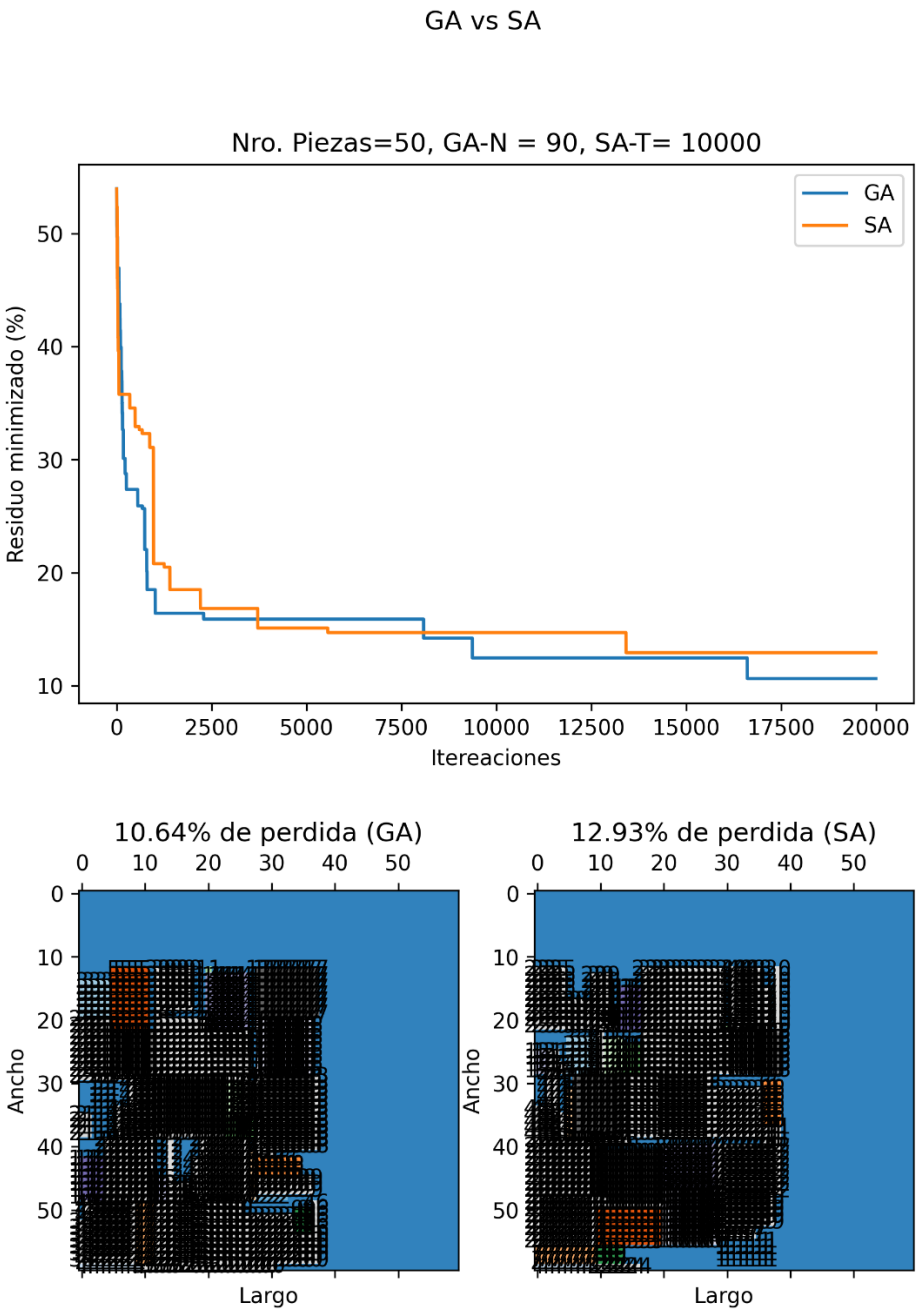


Ilustración 17. Resultados del experimento 8.

A. Tabla de resultados.

Lista de parámetros configurables:

- **Exp**: Experimento.
- **Pc**: Probabilidad de cruce.
- **Pr**: Probabilidad de rotación.
- **Pco**: Probabilidad de cambio de operador.
- **Pcoa**: Probabilidad de cambio de operador anidado.
- **Pi**: Probabilidad de intercambio de piezas.
- **T**: Temperatura
- **Elite**: Numero de padres considerados como mejores.
- **Espacio de corte**: Dimensiones del material a cortar.
- **Iteraciones**: Número de veces que se invoca a la función objetivo por experimento.
- **Perdida**: Porcentaje de material desperdiciado.

Algoritmo genético												
Exp	**pc**	**pr**	**pco**	**pcoa**	**pi**	**Población**	**Piezas**	**Tiempo (s)**	**Elites**	**Espacio de corte**	**Iteraciones**	**Perdida%**
1	0.9	0.3	0.3	0.1	0.5	30	8	5	2	15	1000	0
2	0.9	0.3	0.3	0.1	0.5	30	10	3.8	2	20	1000	6.59
3	0.9	0.3	0.3	0.5	0.7	30	10	3.8	2	20	1000	5.56
4	0.9	0.3	0.3	0.1	0.5	40	20	54	2	30	5000	8
5	0.9	0.3	0.3	0.1	0.5	40	20	56	4	30	5000	13.27
6	0.9	0.3	0.3	0.1	0.5	60	20	12	4	50	5000	8.77
7	0.9	0.5	0.5	0.5	0.5	60	20	83	4	50	5000	7.57
8	0.9	0.3	0.3	0.1	0.5	90	50	1800	4	60	20000	10.64

Tabla 1. Experimentación de parámetros para reducir los residuos de un patrón de corte utilizando algoritmo genético.

Enfriamiento simulado											
Exp	**pr**	**pco**	**pcoa**	**pi**	**Vecinos iniciales**	**t**	**Piezas**	**Tiempo (s)**	**Espacio de corte**	**Iteraciones**	**Perdida %**
1	0.3	0.2	0.1	0.3	30	1000	8	0.8	15	1000	0
2	0.3	0.2	0.1	0.3	30	1000	10	2.6	20	1000	6.59
3	0.3	0.2	0.5	0.7	30	1000	10	0.8	20	1000	3.41
4	0.3	0.3	0.1	0.3	40	1000	20	18	30	5000	7.72
5	0.3	0.3	0.1	0.3	40	1000	20	18	30	5000	7.72
6	0.3	0.3	0.1	0.3	60	10000	20	23	50	5000	7.57
7	0.5	0.5	0.5	0.5	60	10000	20	16	50	5000	6.33
8	0.5	0.5	0.5	0.5	100	10000	50	450	60	20000	12.93

Tabla 2. Experimentación de parámetros para reducir los residuos de un patrón de corte utilizando algoritmo de enfriamiento simulado.

V. Conclusiones

De los experimentos realizados se puede llegar a las siguientes conclusiones:

1) Ninguno de los dos algoritmos es mejor uno del otro, ambos se rigen en función de valores aleatorios y una misma función objetivo que guía las posibles soluciones. Por tal motivo, no siempre se encuentra una solución óptima, pero si una próxima a ella.
2) El éxito de un algoritmo genético depende en mayor medida del tamaño de la población y cambios ligeros en ella; mutaciones y cruces no muy caóticos; En cambio, el algoritmo de enfriamiento simulado se beneficia bastante de los eventos aleatorios.
3) Los algoritmos genéticos son computacionalmente más costosos en función del tamaño de la población; pero, se necesita menos iteraciones para obtener una solución suficientemente buena.
4) El algoritmo de enfriamiento simulado suele caer bastante en óptimos locales si no se configura adecuadamente la temperatura en función de la complejidad del problema. Valores mas altos de la temperatura suelen producir mejores resultados a costa de realizar mas iteraciones; sin embargo, esto no suele ser un problema debido a la simplicidad del algoritmo.

VI. Bibliografía

Lee, L. S. (2008). A genetic algorithms for two-dimensional bin packing problem. *MathDigest. Res. Bull. Inst. Math. Res.*, 34-39.

López, M. Á. (2021). *Optimización del trazado de patrones de corte.* Madrid: Universidad Autonoma de Madrid.

Onwubolu, G. a. (2003). A genetic algorithm approach for the cutting stock problem. *Journal of Intelligent Manufacturing*, 209-218.

Parmar, K. B. (2014). Cutting stock problem: A survey of evolutionary computing based solution. En *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)* (págs. 1-6).

Ruvalcaba Sánchez, L. G. (2014). *Heuristica de dos-etapas para el problema de corte de piezas con guillotinado bidimensional.* Dialnet.

Sisca Octarina, V. A. (2019). Gilmore and gomory model on two dimensional multiple stock size cutting stock problem. *Journal of Physics: Conference Series*, 012015.