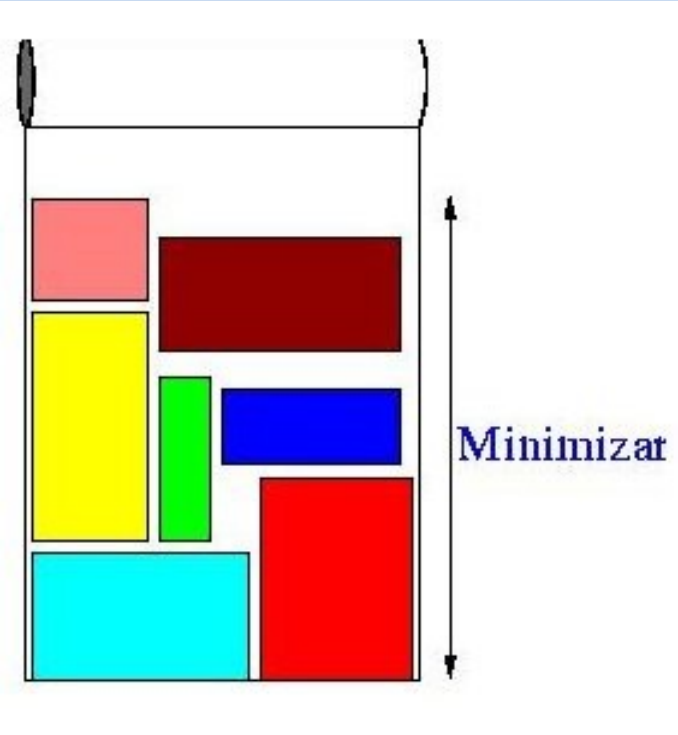
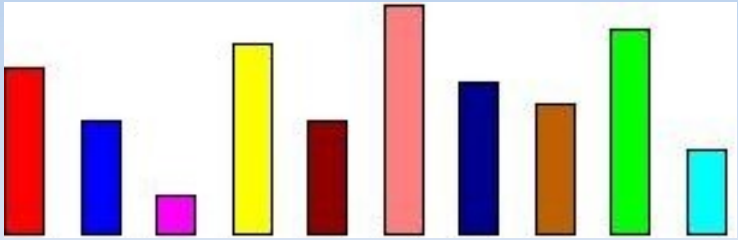
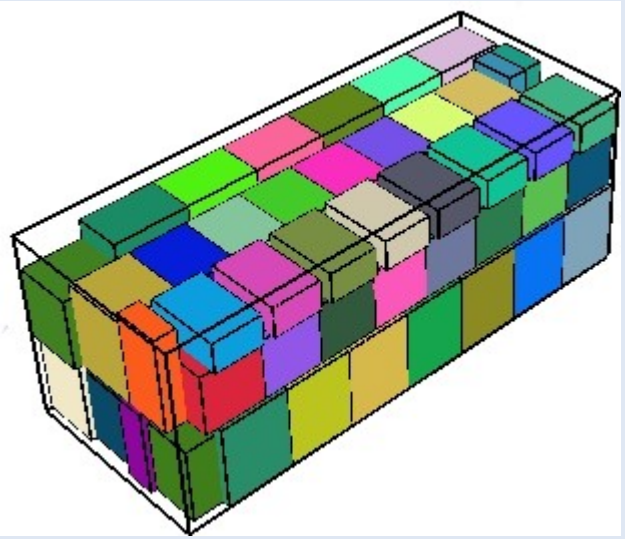
Cutting Stock Problem (CSP) - Unidimensional







Cutting Stock Problem (CSP) - Unidimensional

Funcion Objetivo: Minima Area

$$CS = \sum_{i=1}^{N} L_i$$

Cutting Stock Problem (CSP) - Unidimensional

- Solución inicial Greedy (Perturbada)
- Operadores de Vecindad:
  - Movimiento de las piezas
    - Para generar la vecindad se movera cada una de las piezas, a cualquiera de las posiciones libres en el tope.
- Generación de la vecindad
  - Tamaño de vecindad = # Piezas x Ancho

Cutting Stock Problem (CSP) - Unidimensional

```
class Piece {
 public:
  int large;
  Piece(int);
  Piece clone();
};
```

Cutting Stock Problem (CSP) - Unidimensional

- class Pattern {
- public:
- int width;
- int height;
- int num_pieces;
- list<Piece>* pieces;
- int area_ocup;
- int area_no_ocup;
- int lines;

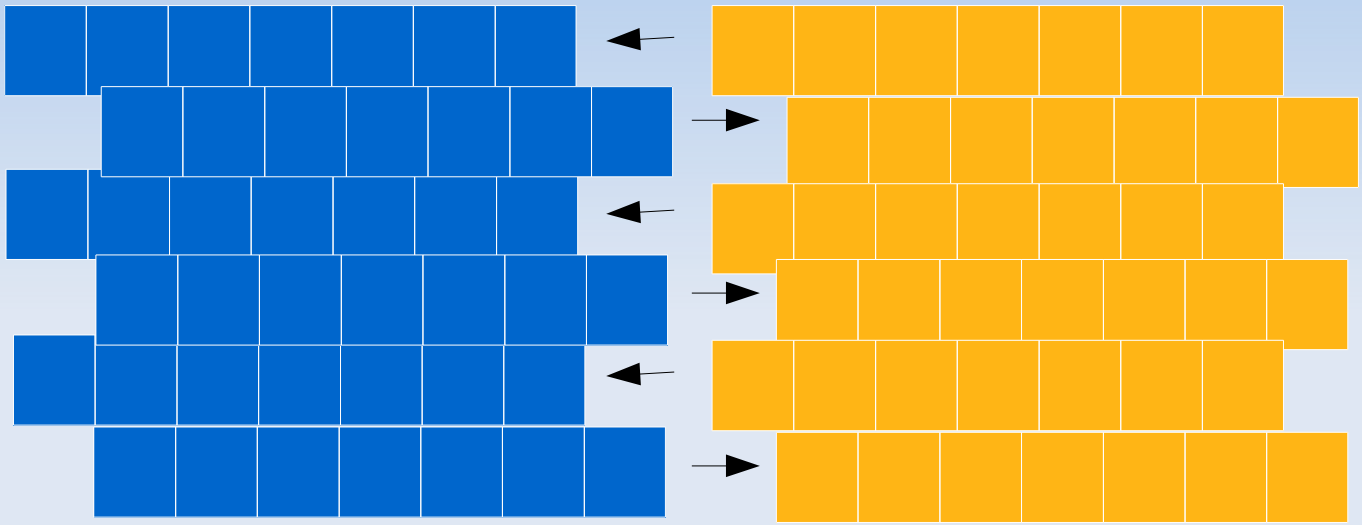
- Pattern();
- Pattern(int, int);
- Pattern(list<Piece *>, int, int);
- Pattern perturb(Pattern);
- int quality();
- Pattern clone();
- void remove(int);
- void add(int,Piece);
- list<Pattern> genVicinity();
- Pattern vicinityOperator();
- void updateHeight();

Cutting Stock Problem (CSP) - Unidimensional

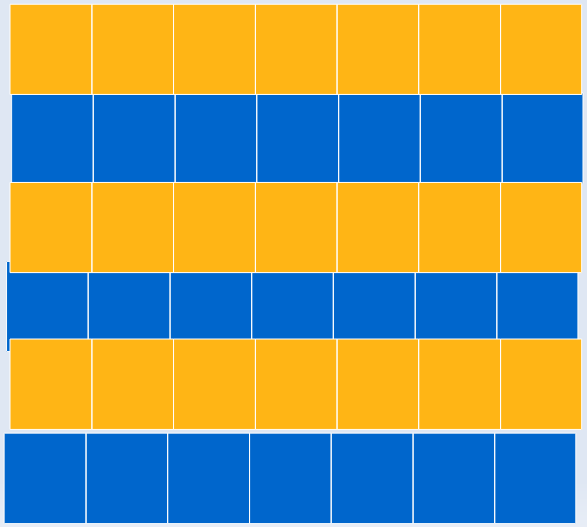
- Algoritmo Genetico

Cutting Stock Problem (CSP) - Unidimensional

- Crossover

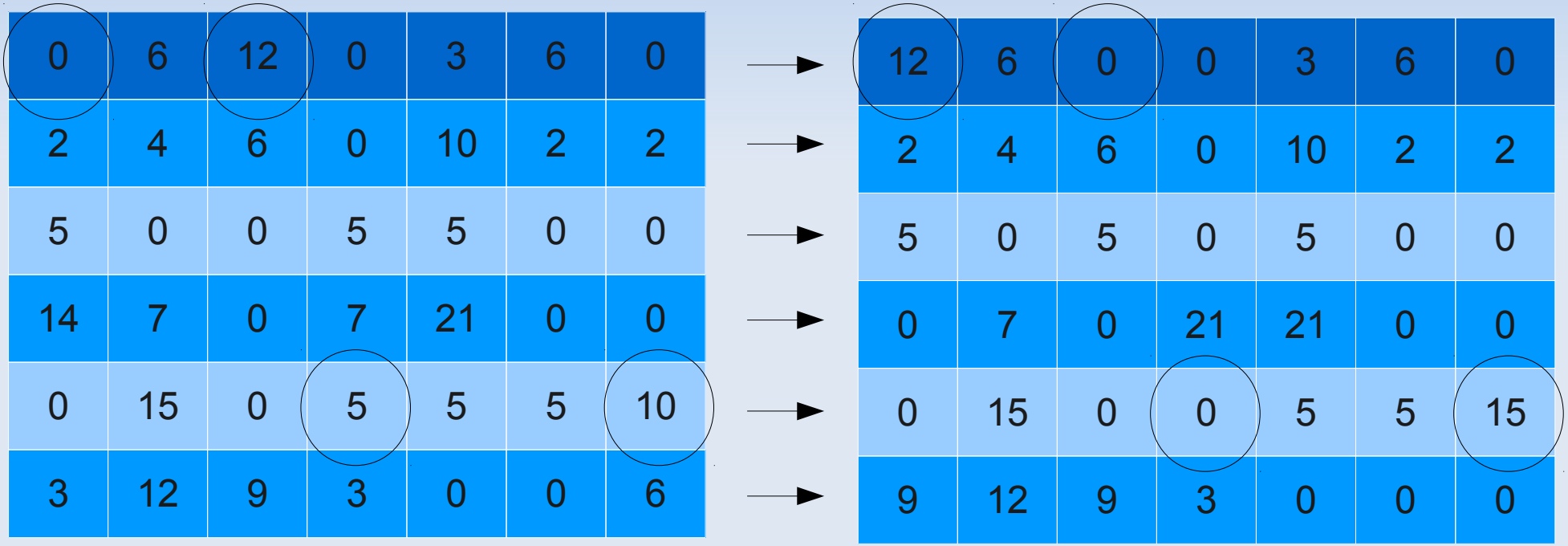


- Hijo =



Cutting Stock Problem (CSP) - Unidimensional

- Mutación con probabilidad 0.3



Cutting Stock Problem (CSP) - Unidimensional

- Resultados Simulated Anneling

	Optimo	Mejor Solución	Solución promedio	Peor solución	Tiempo promedio (seg)
55 piezas	12	12	16.33	22	0.07
66 piezas	5	5	9	13	0.078
100 piezas	4	4	8.26	12	0.112
150 piezas	3	3	14.33	23	0.175

- MaxIter = 20.0; Ro = 1.05; Temp = 100.0; Alfa = 0.8;

Cutting Stock Problem (CSP) - Unidimensional

- Resultados Algoritmo Genético

	Óptimo	Mejor Solución	Solución promedio	Peor solución	Tiempo promedio (seg)
55 piezas	12	12	23.33	42	1.383
66 piezas	5	5	13.26	21	1.268
100 piezas	4	4	19.733	28	1.126
150 piezas	3	23	44.66	73	1.774

Cutting Stock Problem (CSP) - Unidimensional