



UNIVERSIDAD VERACRUZANA  
INSTITUTO DE INVESTIGACIONES EN  
INTELIGENCIA ARTIFICIAL

*Un método para el acomodo de objetos basado en intercambios ponderados, que minimiza el costo de acceso global a estos*

José Alberto López López

Asesor: Dr. Antonio Marín Hernández

7 de julio de 2024

# Contenido de la presentación

1 Introducción

2 Hipótesis y objetivos

3 Marco teórico

4 Metodología propuesta

5 Resultados

## 1 Introducción

## 2 Hipótesis y objetivos

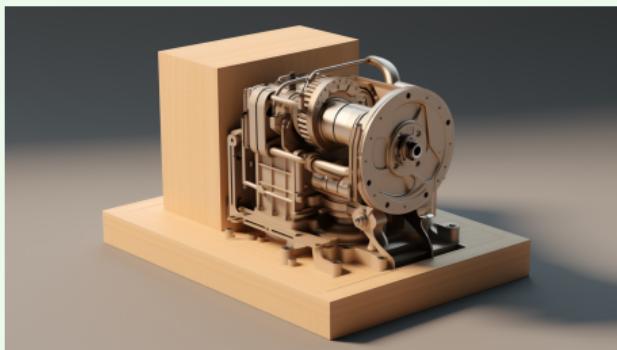
## 3 Marco teórico

## 4 Metodología propuesta

## 5 Resultados

# Problema a abordar

Problemas relacionados con disponer objetos en espacios delimitados:

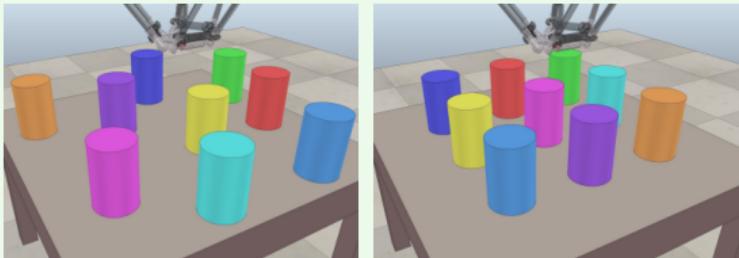


# Características generales del problema

En el presente trabajo se pretende desarrollar un método para el acomodo de objetos en un espacio delimitado, de tal forma que el costo para acceder (sujetar) a cada uno de estos se minimice.

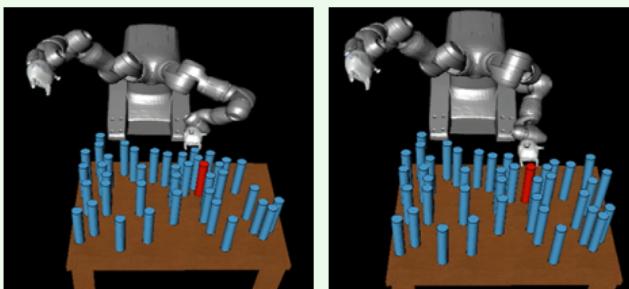


# Trabajos relacionados



Mover un objeto rodeado de obstáculos a una posición meta. Imagen tomada de [Pinto et al., 2018].

Reordenamiento donde se conoce la configuración final. Imagen tomada de [Han et al., 2017].



Tomar un objeto rodeado de obstáculos en desorden. Imagen tomada de [Akbari et al., 2019].

1 Introducción

2 Hipótesis y objetivos

3 Marco teórico

4 Metodología propuesta

5 Resultados

# Hipótesis

- 💡 Existe una metodología para el acomodo de objetos en un espacio delimitado que minimiza el número promedio de acciones necesarias para acceder (tomar) a cualquiera de ellos.

# Objetivo general

- ▶ Diseñar un algoritmo para el arreglo de objetos en un espacio delimitado, con la finalidad de que un manipulador sea capaz de acceder a cualquiera de los objetos, realizando un número mínimo de acciones o movimientos.

# Objetivos particulares

- ① Analizar la literatura relacionada para comparar los diferentes métodos utilizados en problemas similares.
- ② Seleccionar y adaptar las metodologías útiles para el problema planteado.
- ③ Definir las métricas de eficiencia para los algoritmos a implementar.
- ④ Proponer un procedimiento adecuado para encontrar arreglos eficientes de objetos.
- ⑤ Evaluar la metodología propuesta.

1 Introducción

2 Hipótesis y objetivos

3 Marco teórico

4 Metodología propuesta

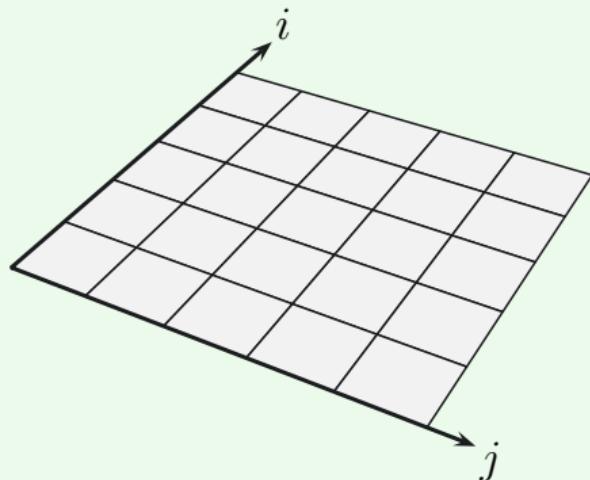
5 Resultados

# Espacio de trabajo

Se desea que el algoritmo propuesto se aplique a una gran variedad de circunstancias de la vida real, por lo cual se trató de que los fundamentos de este fueran lo más generales posible.

## Definición

Sea  $E$  un espacio discreto (malla) de  $n \times m$  localidades o celdas, cada una representada por  $e_{ij}$ .

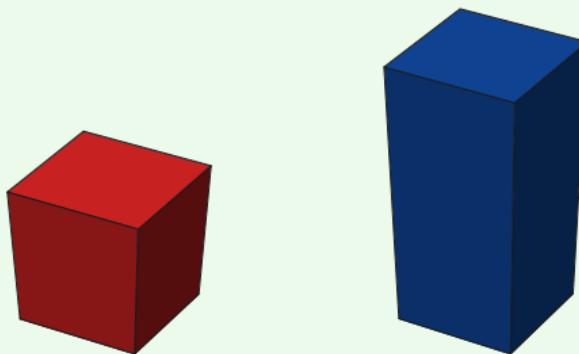


# Conjunto de objetos

Se desea colocar en  $E$  un grupo de objetos.

## Definición

Sea  $O = \{o_1, o_2, \dots, o_N\}$  el conjunto de objetos disponibles para colocar en las celdas de  $E$ , donde  $N$  es el número total de objetos y  $N \leq nm$ .



# Atributos de los objetos

## Definición

Sea  $A = \{a_1, a_2, \dots, a_{N_A}\}$  el conjunto de atributos que puede tener un objeto, por ejemplo el tamaño, color, forma, etc.

## Definición

Sea  $A_k = \{a_{k,1}, a_{k,2}, \dots, a_{k,N_A}\}$  el conjunto de atributos asociado a un determinado objeto  $o_k$ .

Es decir, si por ejemplo,  $o_1 \equiv \text{red cube}$  y  $o_2 \equiv \text{blue cylinder}$ , entonces:

$$A = \{\text{altura, color, ...}\}$$

$$A_1 = \{10\text{cm, rojo, ...}\}$$

$$A_2 = \{20\text{cm, azul, ...}\}$$

# Clases de objetos

## Definición

Se define  $C = \{C_1, C_2, \dots, C_{N_C}\}$  como el conjunto de clases distintas de objetos, donde cada clase representa un  $A_k$  distinto, siendo  $N_C$  el número total de clases o  $A_k$ 's diferentes.

Entonces, si  $o_1 \equiv$   y  $o_2 \equiv$  

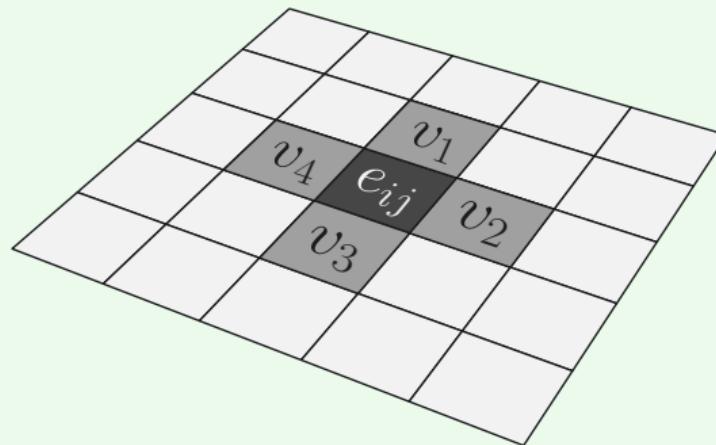
$$\text{clase}(o_1) = C_1 \equiv \text{atributos cubo}$$

$$\text{clase}(o_2) = C_2 \equiv \text{atributos prisma}$$

# Vecindad

## Definición

Se define una vecindad  $V_{ij} = \{v_1, v_2, \dots, v_{N_V}\}$ , como un conjunto de celdas, las cuales se consideran localidades vecinas a la localidad  $e_{ij}$ .

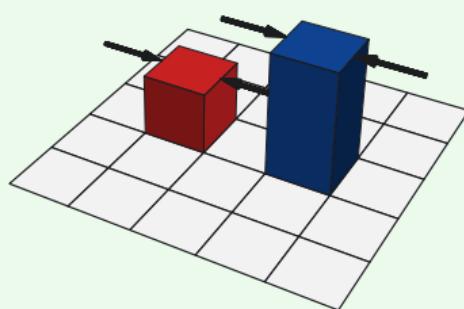


# Formas de sujeción

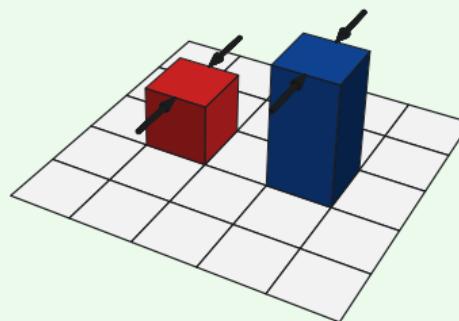
## Definición

Sea  $S_K = \{s_{K,1}, s_{K,2}, \dots, s_{K,N_S}\}$  el conjunto de formas de sujeción asociado a una clase  $C_K$ .

Por ejemplo, si  $S_1 = S_2 = \{H, V\}$ :



(a) Forma de sujeción  $H$ .



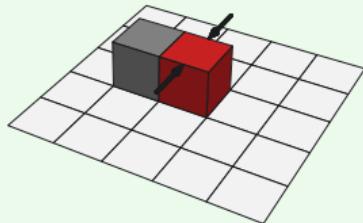
(b) Forma de sujeción  $V$ .

# Restricciones de sujeción

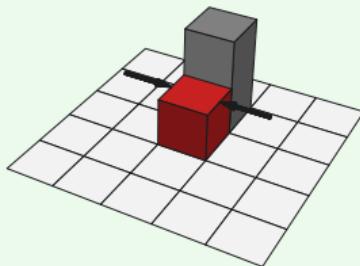
Ejemplos de cómo se pueden restringir las formas de sujeción en función de la vecindad de un objeto.



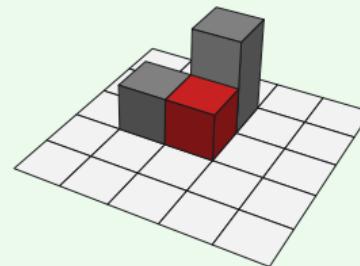
*V*



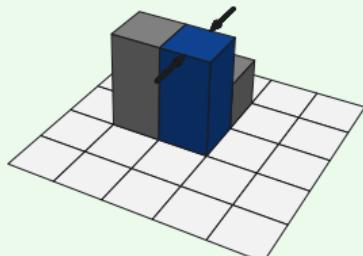
*H*



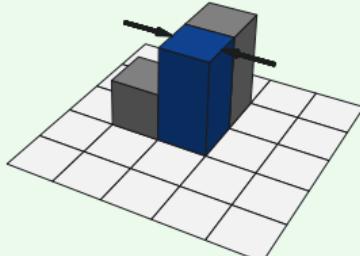
*V*



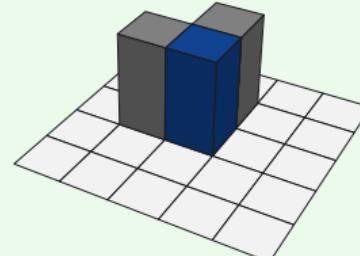
*V*



*H*



*V*



# Costos de acceso a los objetos

## Definición

Sea  $t_{ij}$  la variable discreta que cuantifica el número de acciones necesarias para tomar el objeto en  $e_{ij}$ , lo cual se denomina como el costo de tomar un objeto.

Así,  $t_{ij} - 1$  corresponderá al número de objetos-obstáculo que hay que retirar antes de tomar un objeto de interés.

## Definición

Asimismo, se define el costo global  $T$  de un acomodo de objetos como la suma de los costos individuales  $t_{ij}$ :

$$T = \sum_{i,j} t_{ij}$$

1 Introducción

2 Hipótesis y objetivos

3 Marco teórico

4 Metodología propuesta

5 Resultados

# Funciones de puntuación

Función:  $puntuacionPar(e_{i_1j_1}, e_{i_2j_2})$

**Datos de entrada:** Celdas de los elementos vecinos.

**Resultado:** Puntuación del par de vecinos.

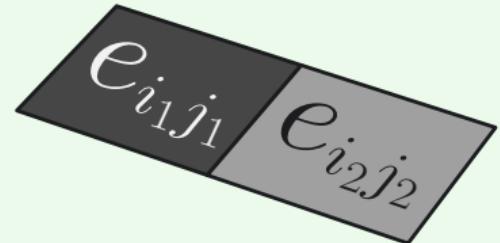
**Definiciones:**  $C_{ij}$ : clase del elemento en  $e_{ij}$ .  
 $C_0$ : celda vacía.

1 **function**  $puntuacionPar(e_{i_1j_1}, e_{i_2j_2})$ :

```

2   switch  $C_{i_1j_1}, C_{i_2j_2}$  do
3     case  $C_1, C_0$  do return 3
4     case  $C_1, C_2$  do return 1
5     case  $C_2, C_1$  do return 2
6     case  $C_2, C_0$  do return 1
7     case  $C_0, C_1$  do return 2
8     case  $C_0, C_2$  do return 1
9     case  $C_{k_C}, C_{k_C}$  do return 0
10   end
11 end
```

Función para obtener la puntuación de un par de elementos vecinos en la malla.



# Funciones de puntuación

## Función: *puntuacionElemento*( $e_{ij}$ )

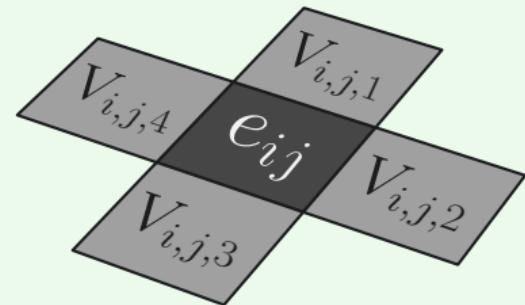
**Datos de entrada:** Celda del elemento a puntuar  $e_{ij}$ .

**Resultado:** Puntuación  $p$  del elemento.

```

1 function puntuacionElemento( $e_{ij}$ ):
2    $p_H = \text{puntPar}(e_{ij}, V_{i,j,2}) + \text{puntPar}(e_{ij}, V_{i,j,4})$ 
3    $p_V = \text{puntPar}(e_{ij}, V_{i,j,1}) + \text{puntPar}(e_{ij}, V_{i,j,3})$ 
4    $p = p_H + p_V$ 
5   if  $C_{ij} \neq C_0$  then
6     if sujetable( $e_{ij}$ ) then  $p = \max(p_H, p_V) + 3$ 
7     if imposible( $e_{ij}$ ) then  $p -= 2$ 
8   end
9   return  $p$ 
10 end
```

Función para calcular la puntuación de un elemento de la malla.



# Funciones de puntuación

Función: *puntuacionGlobal()*

$$puntuacionGlobal() = \sum_{i,j} puntuacionElemento(e_{ij})$$

# Algoritmo principal

## Algoritmo: *Recocido simulado*

**Datos de entrada:** Temperatura inicial  $T$ , constantes  $\alpha$ ,  $a$ ,  $b$ , malla con objetos.

**Resultado:** Malla con objetos acomodados (representada por una matriz de enteros).

Q Parámetros del algoritmo.

1  $T = 50,000,000$

2  $\alpha = 0.0000001$

3  $a = 0.99991$

4  $b = 1$

Q Puntajes actual y de intercambio.

5  $p_A = \text{puntuacionGlobal}()$

6  $p_I = 0$

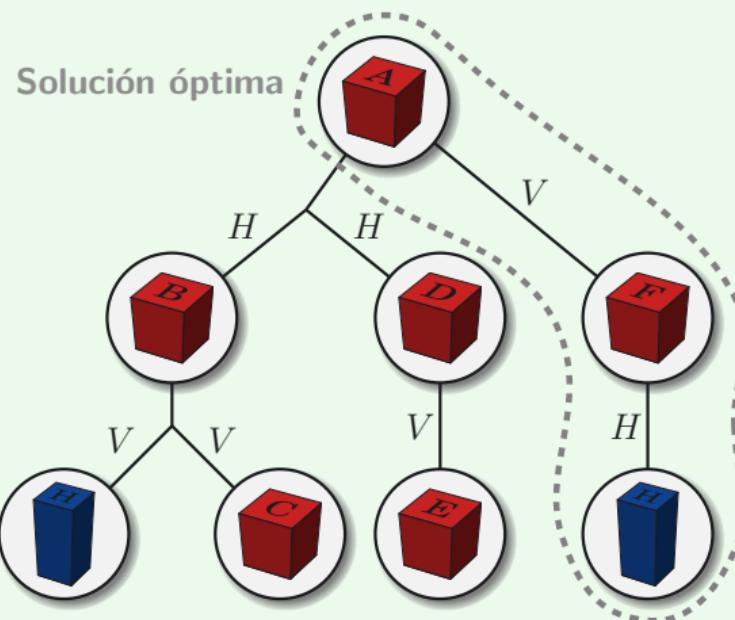
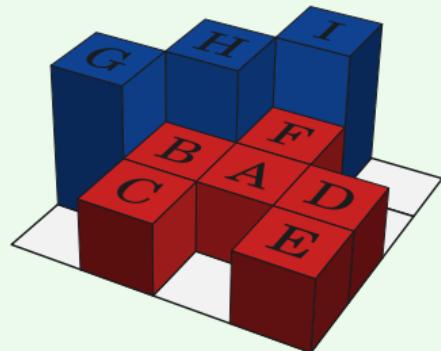
```

7 while  $T > \frac{b}{1-a}$  do
8   Intercambiar aleatoriamente 2 elementos
      diferentes en la malla.
9    $p_I = \text{puntuacionGlobal}()$ 
10  if  $p_A \leq p_I$  or  $\text{random}([0, 1]) < e^{-\frac{p_A - p_I}{\alpha T}}$  :
11    Se acepta el intercambio.
12     $p_A = p_I$ 
13  else
14    No se acepta el intercambio.
15  end
16   $T = aT + b$ 
17 end

```

# Funciones de costo

La función de costo utilizada está basada en la que se presenta en el artículo *Manipulation planning among movable obstacles* [Stilman et al., 2007].



# Funciones de costo

## Función: $\text{secuencia}(e_{ij}, l)$

**Datos de entrada:** Celda  $e_{ij}$  del objeto deseado y lista vacía  $l$ .

**Resultado:** Lista  $l$  con la secuencia de elementos a retirar.

**Definiciones:**  $h_{ij}$ ,  $C_{ij}$ ,  $o_{ij}$ : altura, clase y datos relevantes del elemento en  $e_{ij}$ .

```

1 function secuencia( $e_{ij}$ ,  $l$ ):
2   if  $C_{ij} = C_0$  then Finalizar con éxito.
3   if  $o_{ij} \notin l$  then
4     append( $o_{ij}$ ,  $l$ )
5     if sujetable( $e_{ij}$ ) then
6       Finalizar recursión con éxito.
7     else
8       if  $h_{i+1,j} \geq h_{ij}$ : secuencia( $e_{i+1,j}$ ,  $l$ )
9       if  $h_{i-1,j} \geq h_{ij}$ : secuencia( $e_{i-1,j}$ ,  $l$ )
10      else if Par 2:
11        if  $h_{i,j+1} \geq h_{ij}$ : secuencia( $e_{i,j+1}$ ,  $l$ )
12        if  $h_{i,j-1} \geq h_{ij}$ : secuencia( $e_{i,j-1}$ ,  $l$ )
13      else
14        Si ambos pares fallaron:
15        Finalizar recursión con fallo.
16      end
17    else
18      Finalizar recursión con fallo.
19    end
20  end

```

*Si no, ejecutar un par:*

**else if Par 1:**

**if**  $h_{i+1,j} \geq h_{ij}$ :  $\text{secuencia}(e_{i+1,j}, l)$

**if**  $h_{i-1,j} \geq h_{ij}$ :  $\text{secuencia}(e_{i-1,j}, l)$

**else if Par 2:**

**if**  $h_{i,j+1} \geq h_{ij}$ :  $\text{secuencia}(e_{i,j+1}, l)$

**if**  $h_{i,j-1} \geq h_{ij}$ :  $\text{secuencia}(e_{i,j-1}, l)$

**else**

*Si ambos pares fallaron:*

        Finalizar recursión con fallo.

**end**

**else**

        Finalizar recursión con fallo.

**end**

Finalizar recursión con éxito.

# Funciones de costo

## Algoritmo: *Costo $t_{ij}$ de tomar un objeto*

**Datos de entrada:** Celda  $e_{ij}$  del objeto deseado.

**Resultado:** Costo  $t_{ij}$  de tomar el objeto en  $e_{ij}$ .

**Definiciones:** *encontrarTodas()*: Función de backtracking que retorna todas las soluciones posibles de *secuencia()*.

Q Lista para almacenar todas las secuencias posibles para tomar un objeto.

- 1  $L = \{\}$
- 2  $L = \text{encontrarTodas}(\text{secuencia}(e_{ij}, l))$
- 3  $t_{ij} = \min_i |L_i|$

Como se mencionó, el costo global  $T$  es simplemente la suma de los costos individuales  $t_{ij}$ :

$$T = \sum_{i,j} t_{ij}$$

1 Introducción

2 Hipótesis y objetivos

3 Marco teórico

4 Metodología propuesta

5 Resultados

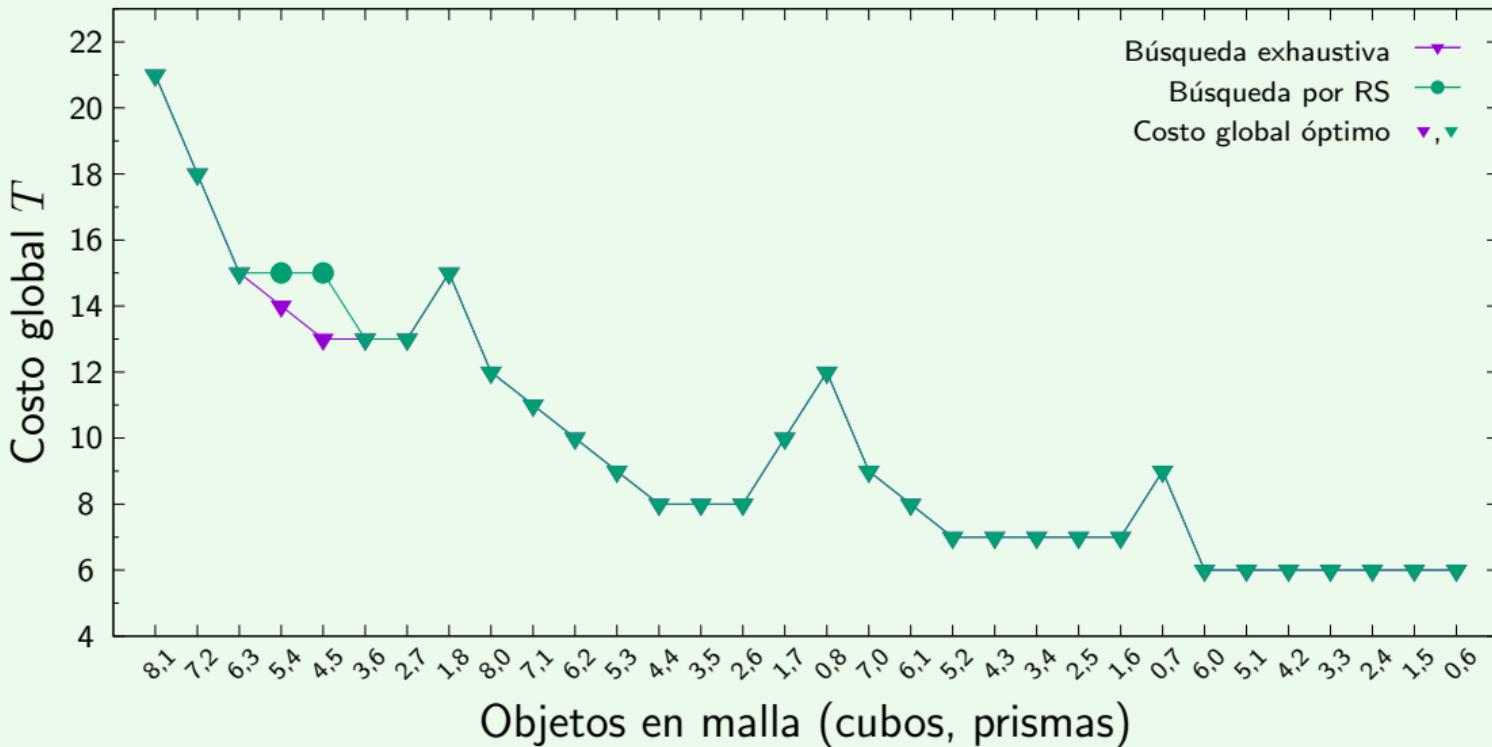
# Tamaños y casos probados

Tamaños de malla y número de combinaciones de cubos y prismas (casos) probados en estas.

Tamaño malla	# Casos probados
$3 \times 3$	32
$3 \times 5$	85
$4 \times 4$	88
$5 \times 5$	226
$6 \times 6$	423
$5 \times 8$	558
$7 \times 7$	834
$8 \times 8$	1312

# Mallas de $3 \times 3$

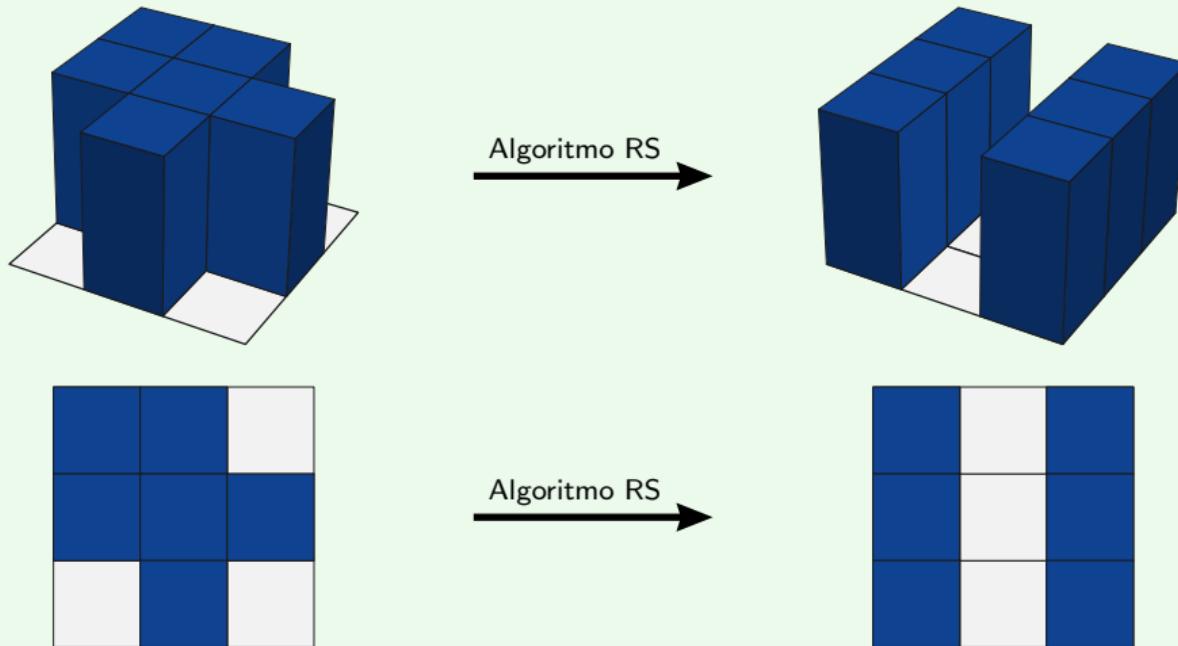
Resultados en mallas de  $3 \times 3$



Objetos en malla (cubos, prismas)

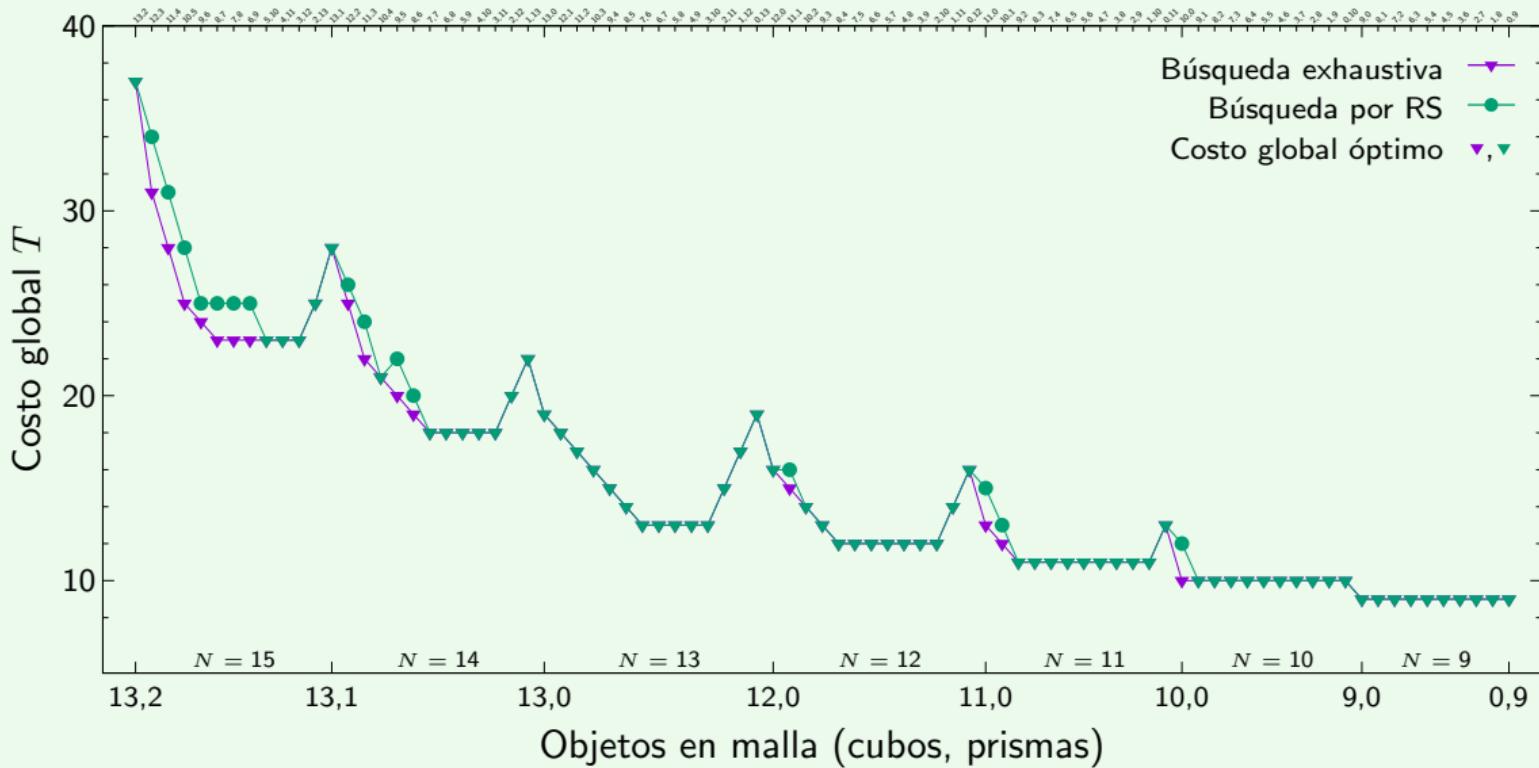
# Ejemplo ilustrativo en malla de $3 \times 3$

Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



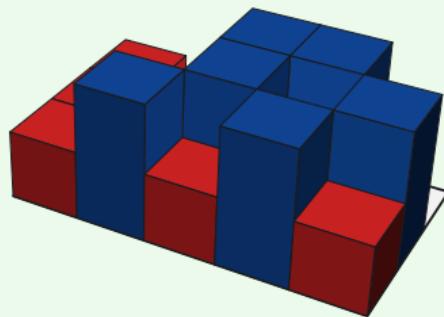
# Mallas de $3 \times 5$

Resultados en mallas de  $3 \times 5$

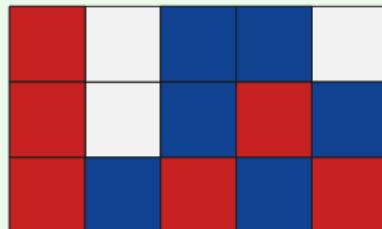
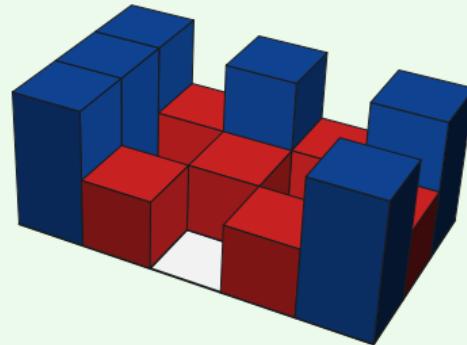


# Ejemplo ilustrativo en malla de $3 \times 5$

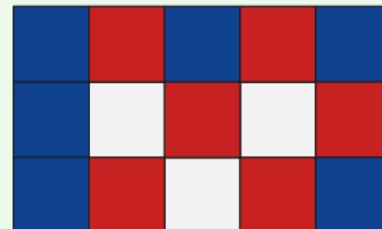
Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



Algoritmo RS  
→

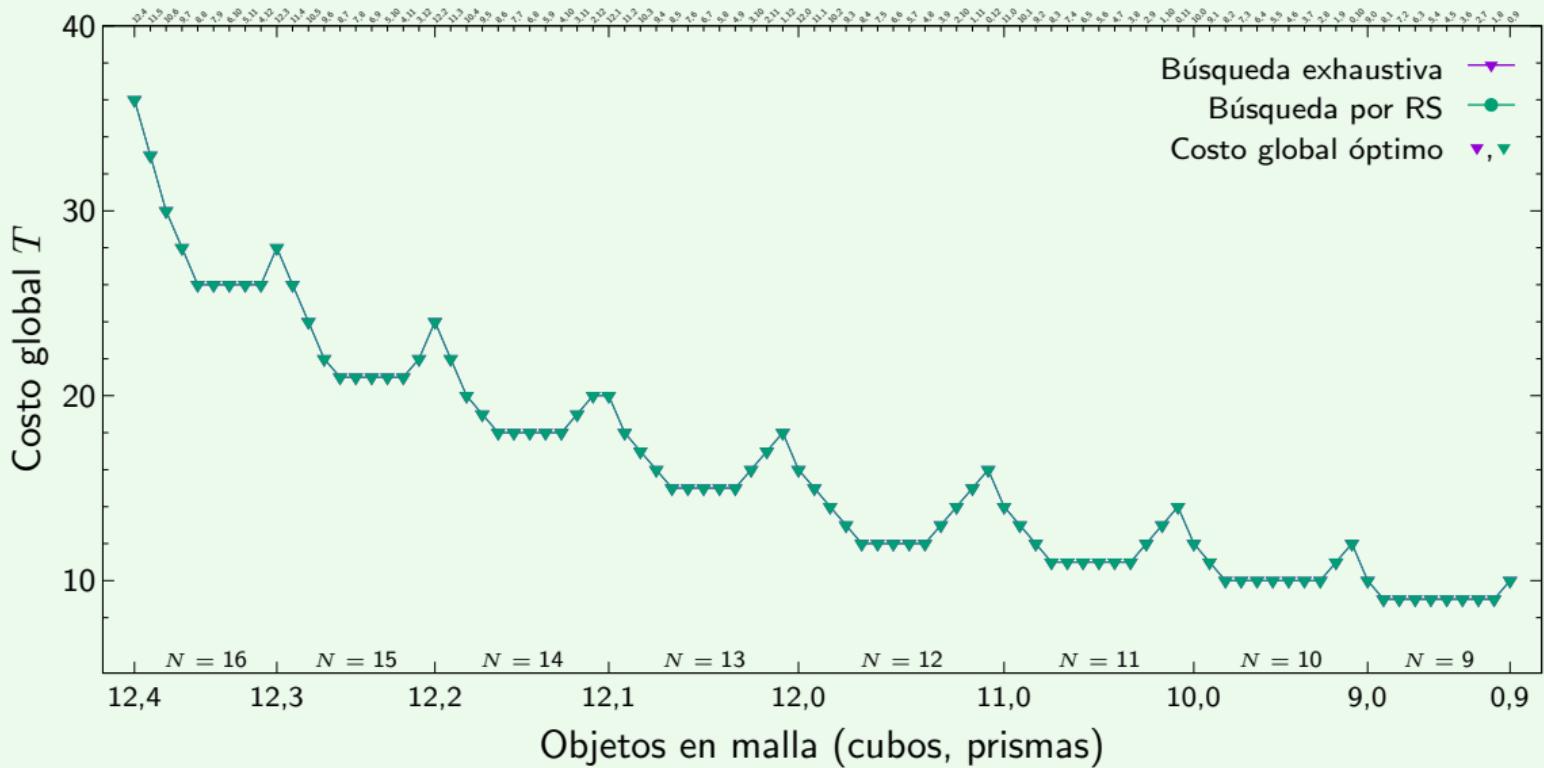


Algoritmo RS  
→



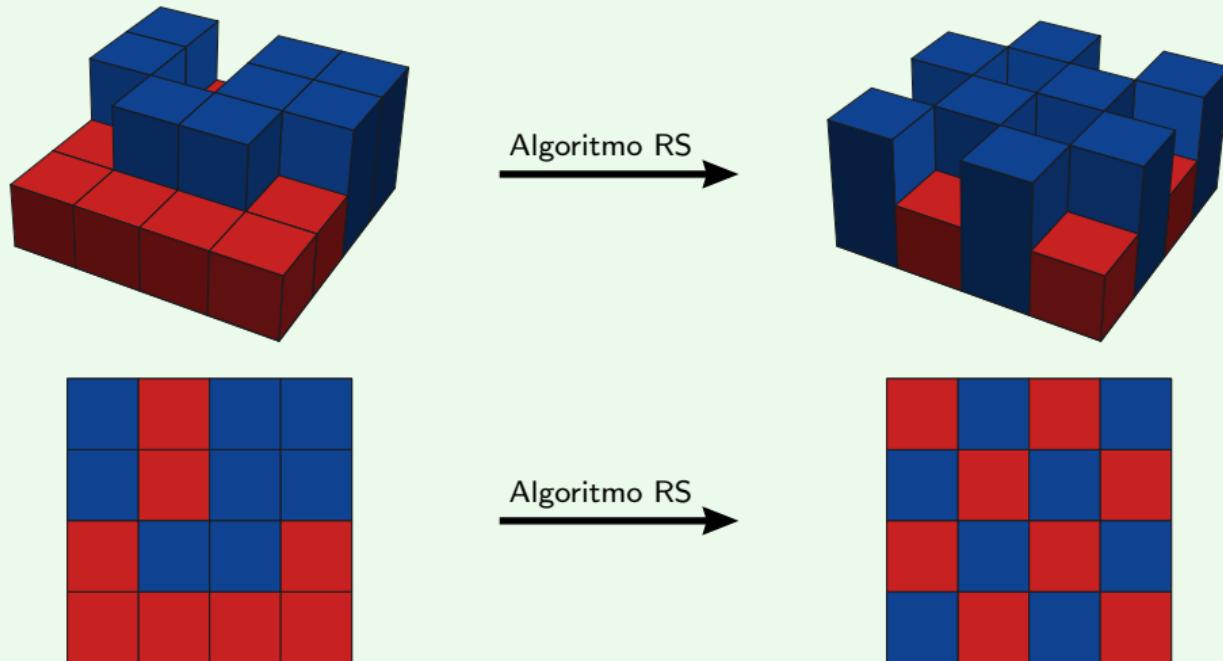
# Mallas de $4 \times 4$

Resultados en mallas de  $4 \times 4$



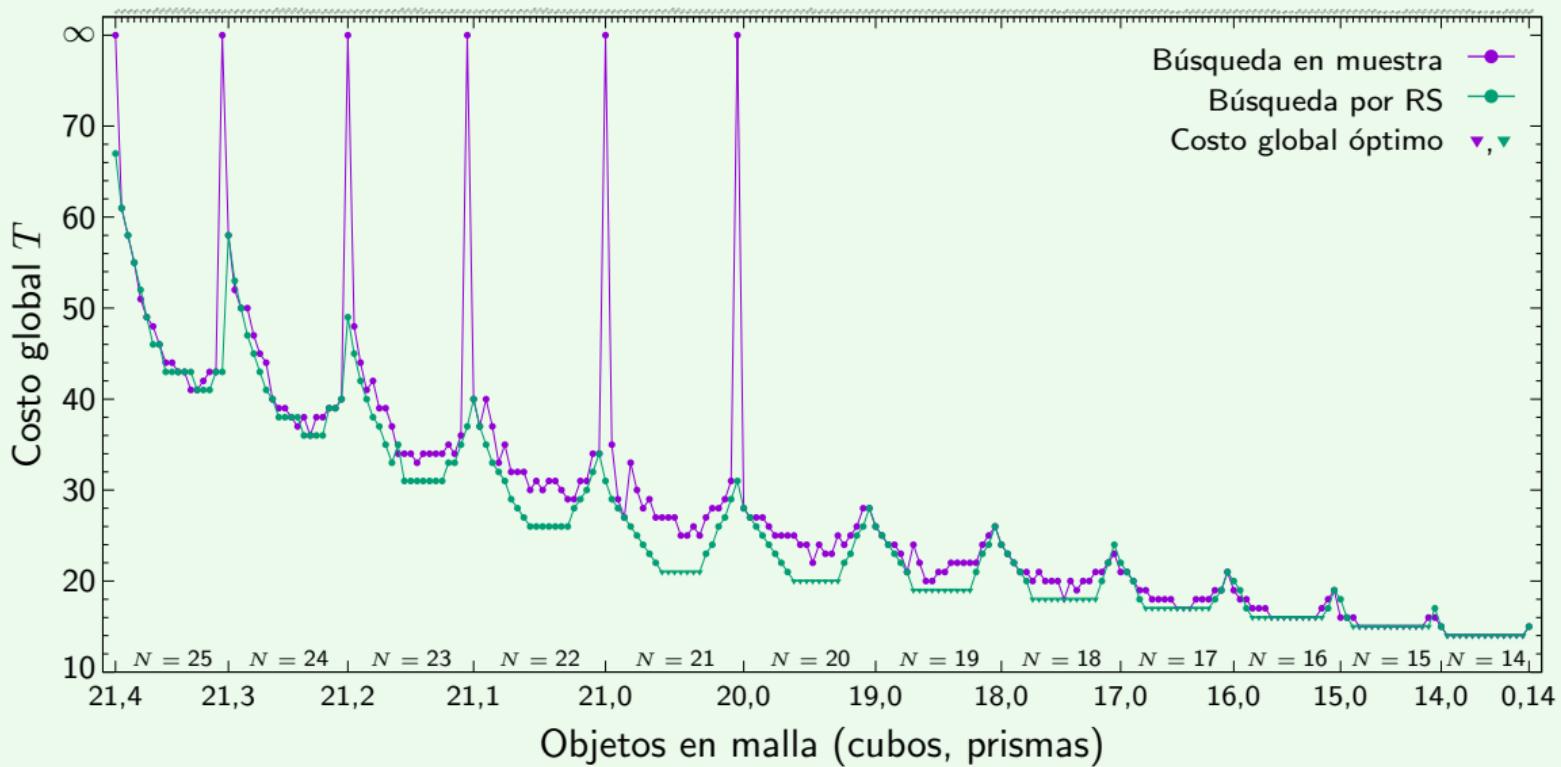
# Ejemplo ilustrativo en malla de $4 \times 4$

Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



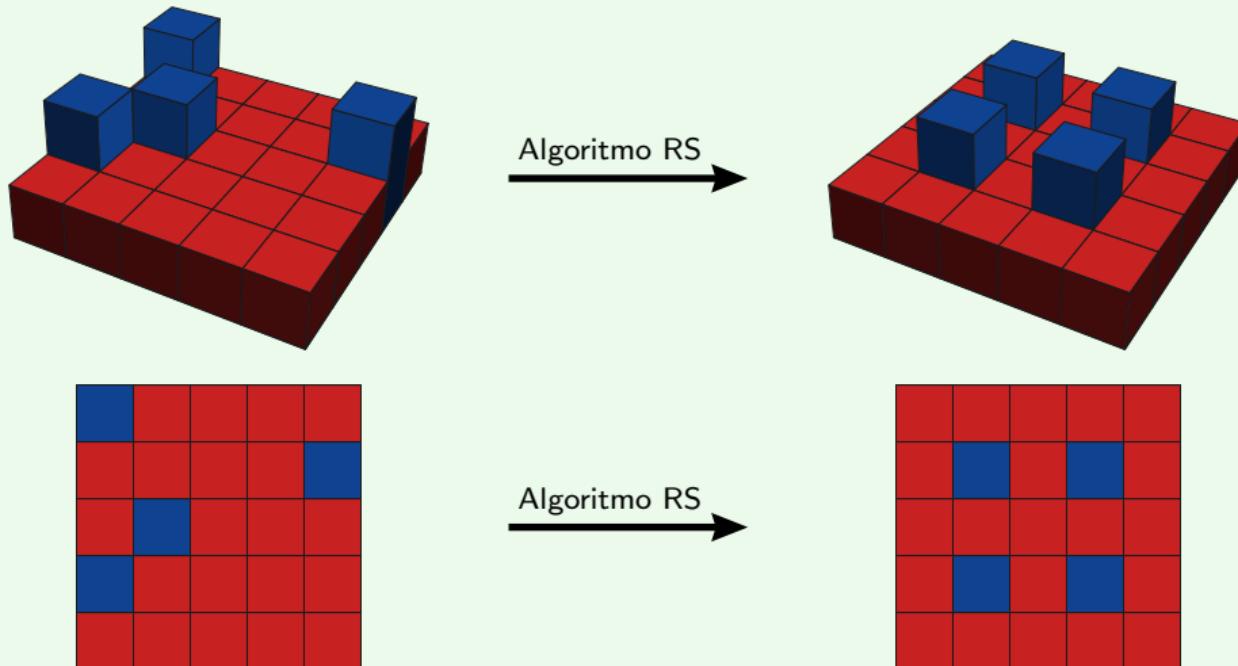
# Mallas de $5 \times 5$

## Resultados en mallas de $5 \times 5$



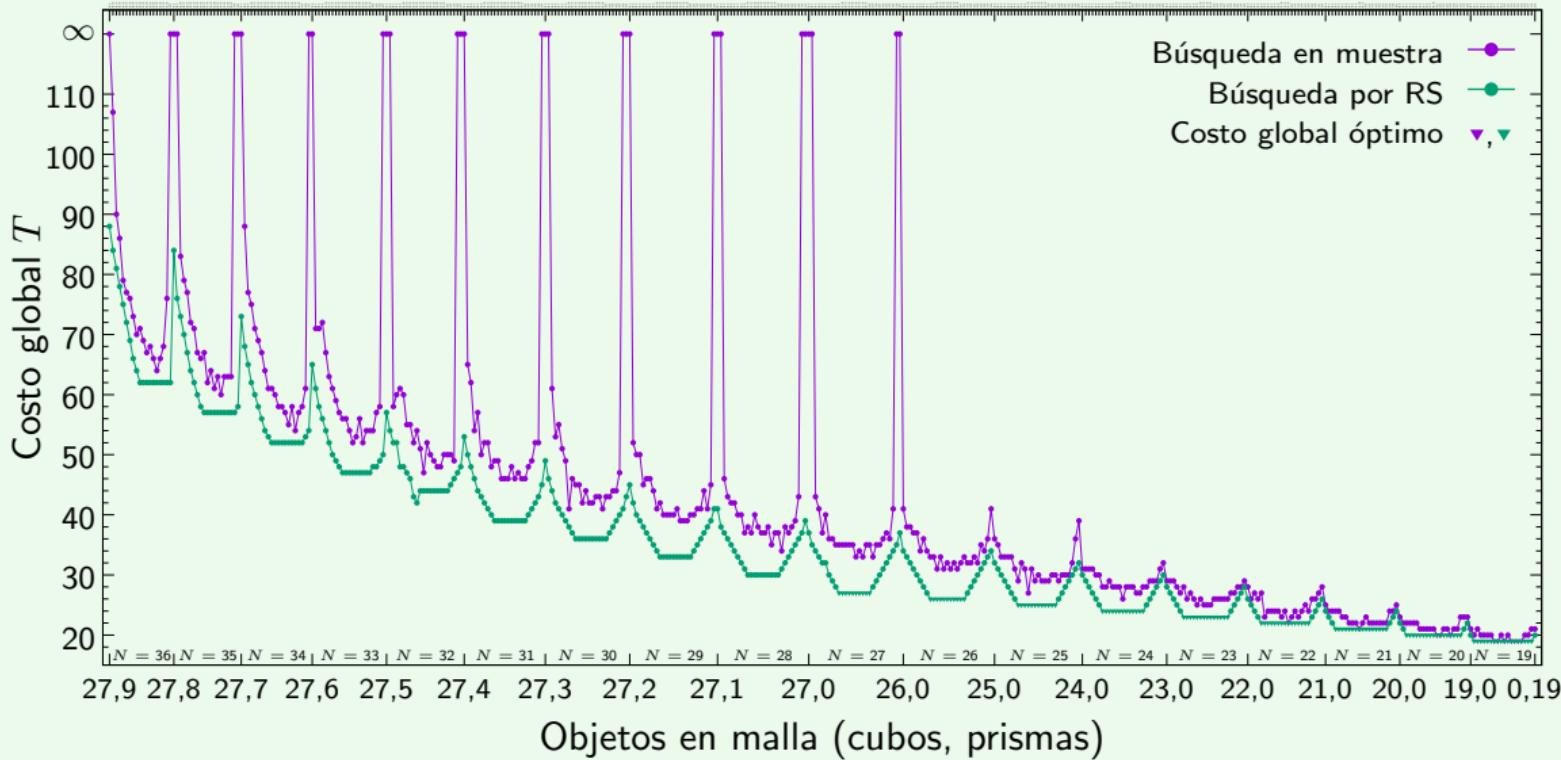
# Ejemplo ilustrativo en malla de $5 \times 5$

Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



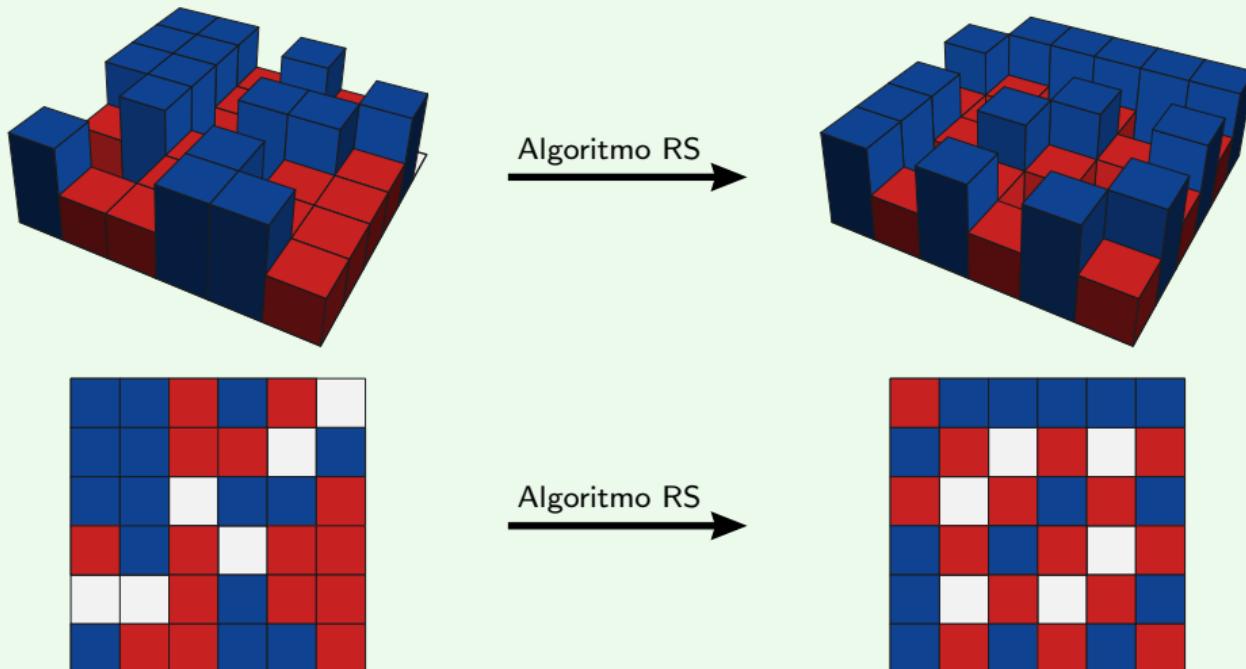
# Mallas de $6 \times 6$

## Resultados en mallas de $6 \times 6$



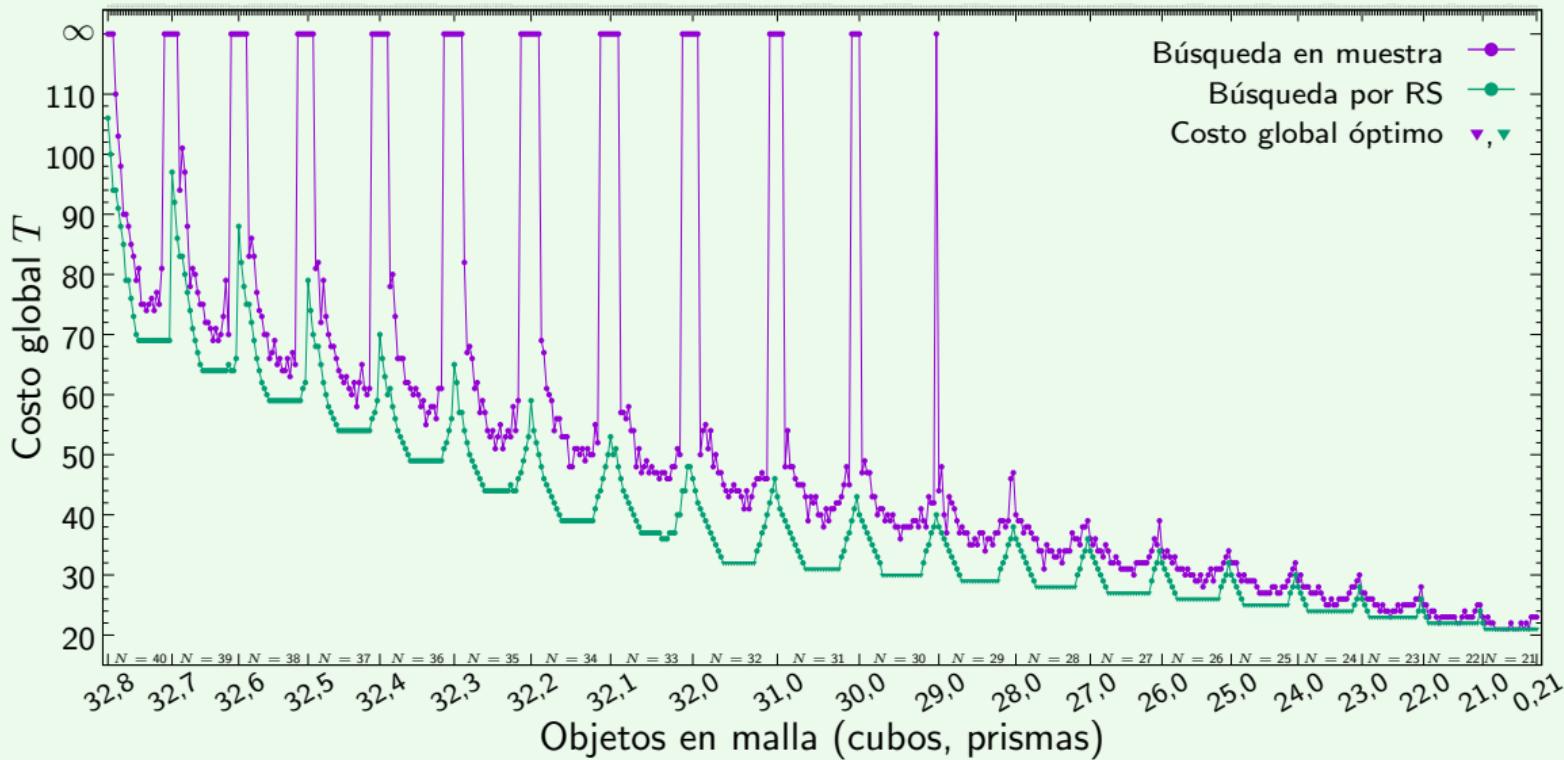
# Ejemplo ilustrativo en malla de $6 \times 6$

Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



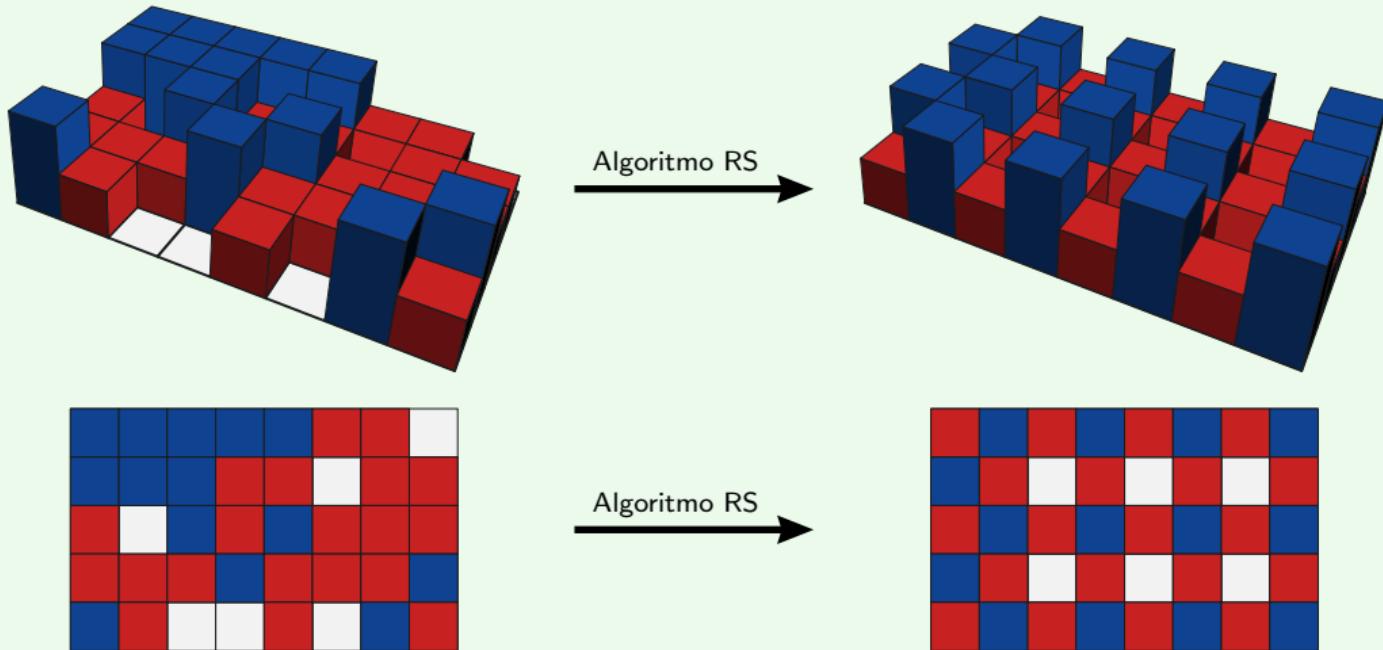
# Mallas de $5 \times 8$

## Resultados en mallas de $5 \times 8$



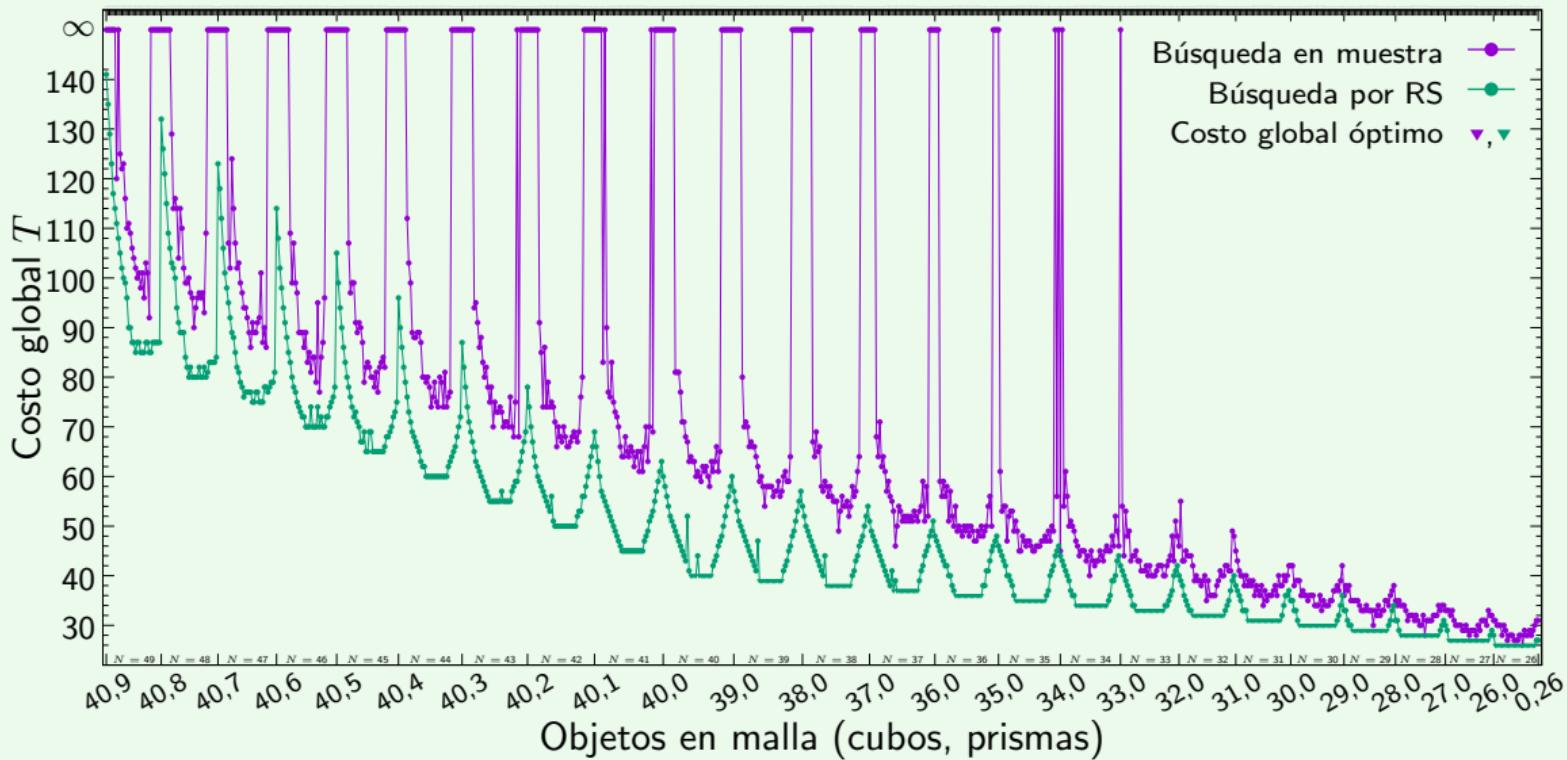
# Ejemplo ilustrativo en malla de $5 \times 8$

Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



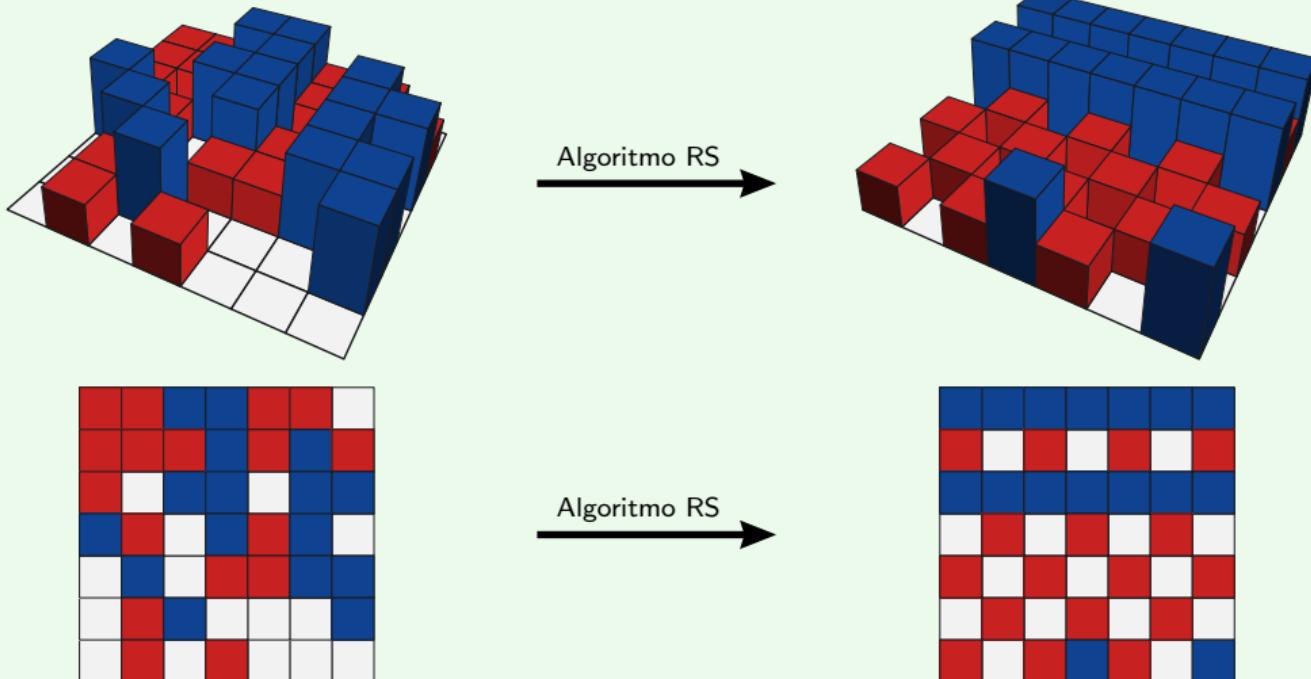
# Mallas de $7 \times 7$

## Resultados en mallas de $7 \times 7$



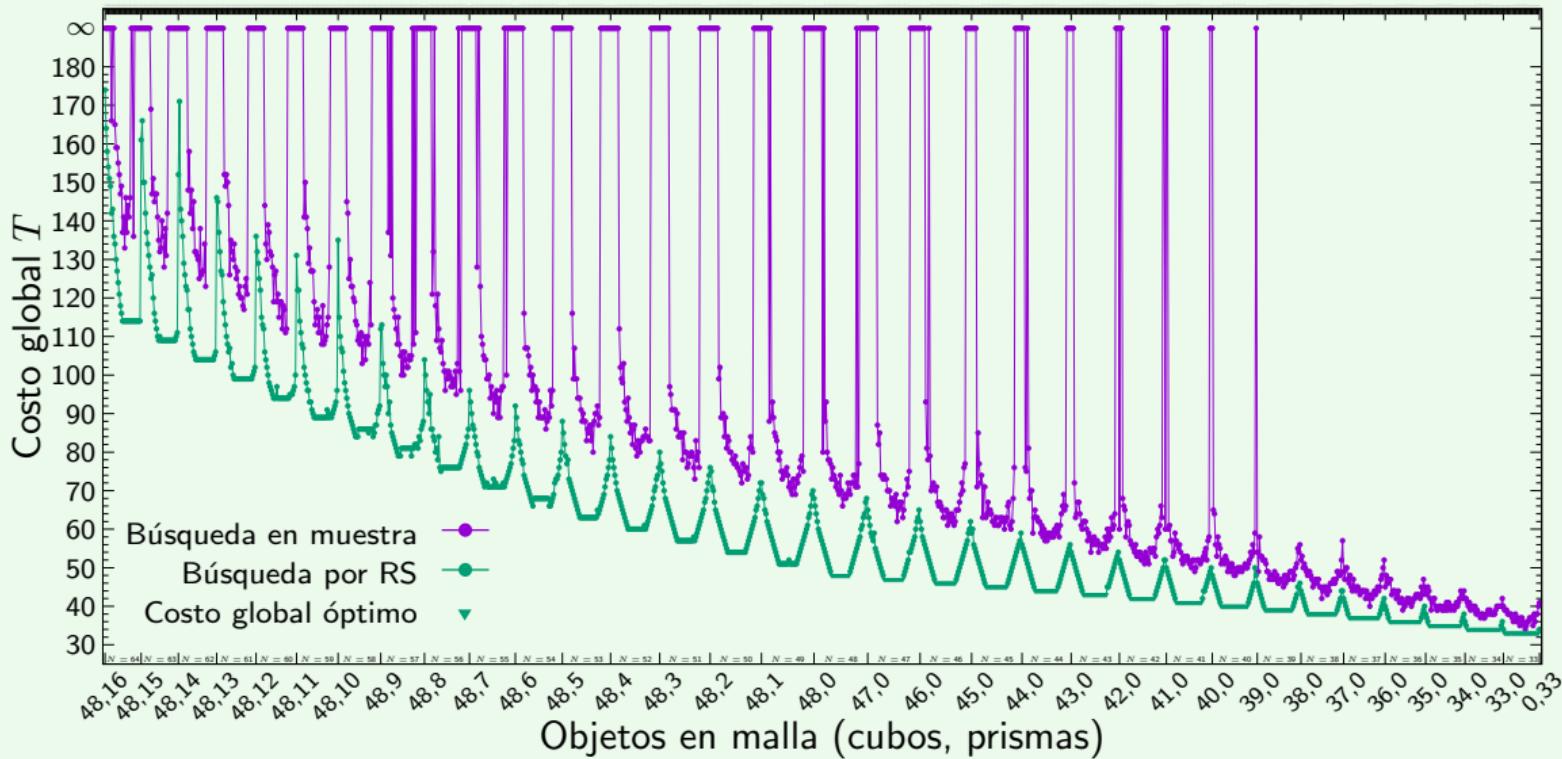
# Ejemplo ilustrativo en malla de $7 \times 7$

Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



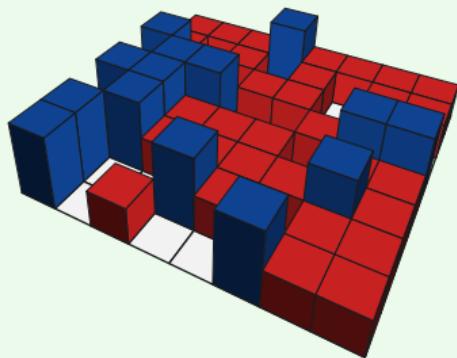
## Mallas de $8 \times 8$

## Resultados en mallas de $8 \times 8$

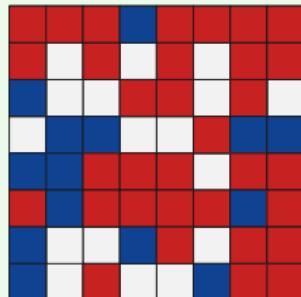
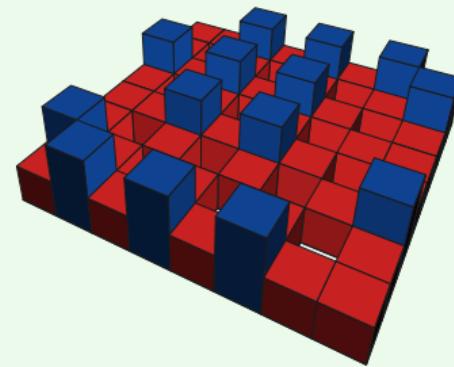


# Ejemplo ilustrativo en malla de $8 \times 8$

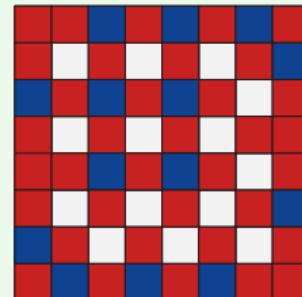
Acomodo aleatorio inicial (izquierda) y acomodo encontrado por el algoritmo (derecha), mostrados en diferentes vistas.



Algoritmo RS



Algoritmo RS



# Tabla de resultados

Resultados del algoritmo propuesto para mallas de tamaño igual o menor a  $4 \times 4$ .

Malla	# Total casos	# Casos óptimos	% Óptimos	Promedio de acciones extra*
<b><math>3 \times 3</math></b>	32	30	93.75 %	0.09
<b><math>3 \times 5</math></b>	85	70	82.35 %	0.33
<b><math>4 \times 4</math></b>	88	88	100 %	0

\* Promedio de acciones (o costo global  $T$ ) extra por caso. Este valor se calcula mediante la expresión  $(\sum T - \sum T_{\text{opt}}) / \# \text{Total casos}$ , donde  $\sum T_{\text{opt}}$  es la suma de costos óptimos para todos los casos probados.

# Referencias

- [Akbari et al., 2019] Akbari, A., Lagriffoul, F., and Rosell, J. (2019).  
*Combined heuristic task and motion planning for bi-manual robots.*  
*Autonomous Robots*, 43(6):1575–1590.
- [Han et al., 2017] Han, S., Stiffler, N., Krontiris, A., Bekris, K., and Yu, J. (2017).  
*High-quality tabletop rearrangement with overhand grasps: Hardness results and fast methods.*  
In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts.
- [Pinto et al., 2018] Pinto, L., Mandalika, A., Hou, B., and Srinivasa, S. (2018).  
*Sample-efficient learning of nonprehensile manipulation policies via physics-based informed state distributions.*  
*arXiv e-prints*, page arXiv:1810.10654.
- [Stilman et al., 2007] Stilman, M., Schamburek, J.-U., Kuffner, J., and Asfour, T. (2007).  
*Manipulation planning among movable obstacles.*  
In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3327–3332.

GRACIAS